Projekt WUM

Bartosz Szymański

1 Wstępna eksploracja i przygotowanie danych

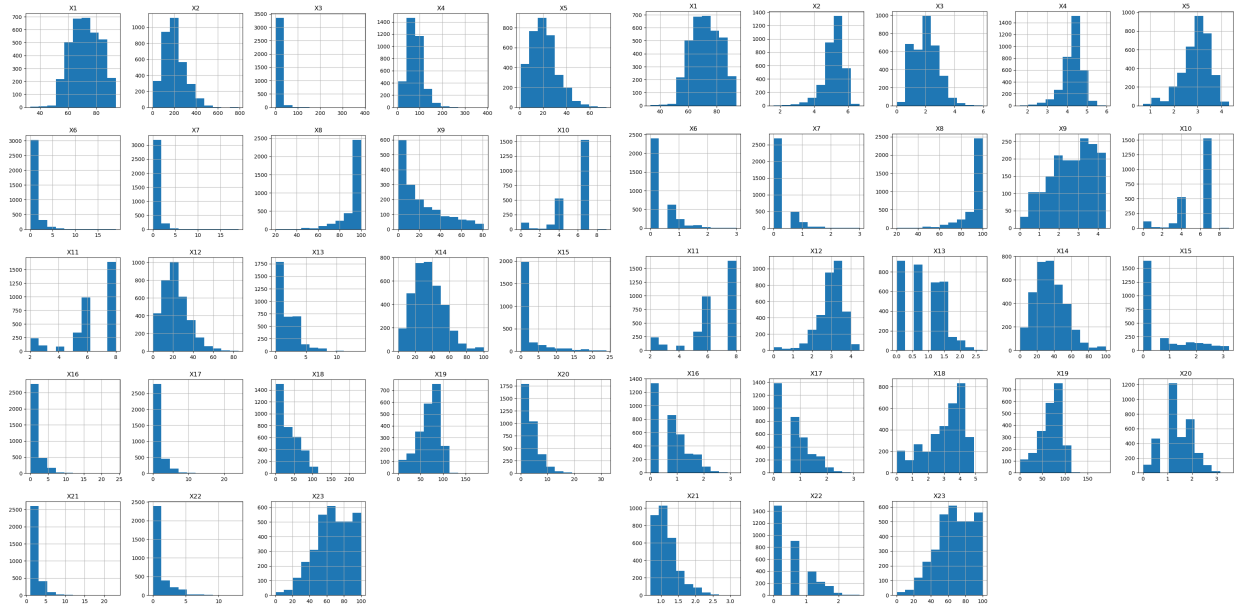
Pracę nad problemem rozpocząłem od dokładnego przyjrzenia się danym oraz przygotowaniem ich do tworzenia modeli. Zadanie polegało na przewidywaniu szans na niewypłacalność (scoring ryzyka kredytowego) na podstawie informacji dotyczących hisotrii bankowej danego klienta. Do dyspozycji mamy 23 zmienne objaśniające, z czego dwie z nich są zmiennymi kategorycznymi, a pozostałe 21 to zmienne przedziałowe. W danych mamy doczynienia z brakami danych. Zmienna celu przyjmuje dwie wartości, zatem nasz problem to klasyfikacja binarna. Rozkład zmiennnej objaśnianej jest względnie równomierny - zmienna Bad pojawia sie w około 52 procentach wierszy, a Good w pozostałych 48 (po odfiltorwaniu pustych wierszy, co będzie opisane w następnym podrozdziale).

1.1 Braki w danych

Pierwszym napotkanym problemem były braki w danych. W części z wierszy pojawiły się wartości ujemne, które zgodnie z poleceniem traktujemy jako braki w danych. Z tego powodu zastąpiłem wartości -9, -8 oraz -7 w zbiorze Nan'ami, żeby mieć pełen pogląd na ilość brakujących wartości. Ponadto wartościami Nan zastąpiłem również piątki oraz szóstki w kolumnie X10 oraz siódemki w kolumnie X11, ponieważ z meta danych wiemy, że są to nieznane wartości. Okazało się, że w naszym zbiorze X występują obserwacjie, dla których każda z zmiennej przyjmuje wartość Nan. Postanowiłem usunąć wszystkie takie wiersze, ponieważ nie wnosiły one żadnej informacji do naszego modelu. Kolejną istotną obserwacją było duża ilość braków danych w kolumnach X9, X10, X15, X19 - odpowiednio 50, 34, 23 oraz 36 procent. Zdecydowałem się jednak pozostawić wszystkie z nich, ponieważ zdaje się, że mogą mieć istotny wpływ na wypłacalność klienta (przede wszystkim X9, X10 oraz X19) oraz dodatkowo dla każdej ze zmiennych mamy wpisane wartości dla conajmniej połowy klientów, zatem usunięcie dowolnej z nich spowodowałoby utratę dużej liczby informacji. Dodatkowo w poźniejszej obróbce danych dodamy zmienną z indykatorem dla zmiennych, które zostały uzupełnione, bo być może istnieje konkretny powód tych braków i istnieją jakieś powiązania między tymi klientami.

1.2 Rozkłady zmiennych

Kolejnym istotnem elementem mojej wstępnej ekspolaracji danych było przyjrzenie się rozkładom poszczególnych zmiennych. Analizując histogramy poszczególnych zmiennych możemy dostrzec, że wiele zmiennych ma rozkład lewoskośny. By poradzić sobie z tym problemem wykorzystałem przekształcenie zmiennych X2, X3, X4, X5, X6, X7, X9, X12, X13, X15, X16, X17, X18, X20, X21, X22 funkcją logarytmiczną, która pozwoliła na zwiększenie wariancji i zbliżenie rozkładu do normalnego. Z racji występowania wartości zerowych wykorzystałem funkcję log(X+1). Na wykresach nie zaobserwowałem też poderzjanych, ani specjalnie odstających wartości, którymi powinniśmy się zająć.



Rysunek 1: Po lewej rozkłady przed przekształceniem logarytmem, po prawej po

1.3 Korelacje pomiędzy zmiennymi

Następnym krokiem analizowania danych było zbadanie korelacji między poszczególnymi zmiennymi. Największy współczynniki korealcji Persona (0.99) okazały sie mieć zmienne X16 oraz X17, co nie dziwi z racji ich definicji - zmienna X17 to zmienna X16 z usniętymi zapytaniami z ostatnich 7 dni. Z racji tak wysokiej wartości współczynnika i silnego oraz oczywistego powiązanie zmiennych postanowiłem usunąć z danych kolumnę X17, żeby uniknąć wykrywania przez model zbędnych lub nieprawdziwch powiązań. Innymi zmiennymi, dla których wartość współczynnika korelacji Persona wyniosła powyżej 0.8 były pary X5 i X12, X10 i X11, X6 i X7, jednak w tych przypadkach każda z kolumn wnosiła dodatkowe informacje, zatem żadnej z nich nie usunłąłem. Do rozwiązania problemu istnień tych zależności w dalszych etapach wykorzystałem modele, które dobrze sobie radzą z tym problemem lub metodę PCA.

1.4 Przygotowanie danych do budowy modeli

Dodatkowymi czynnościami w pracy z danymi było jeszcze zastąpenie wartości Bad jedynkami oraz wartości Good zerami w zmiennej celu. Dodatkowo w kolumnie X10 wartości 8 zastąpiłem wartością 7 z racji, iż oznaczają dokładnie to samo. Ponadto przed rozpoczęciem budowy modeli dane podzieliłem na zbiór treningowy oraz testowy w propocjach 8:2. Kolejnym krokiem było utworzenie pipeline'u, który przygotowywał dane przed ostatecznym urchuchomieniem na nich modeli. W tym celu dane zostały podzielone na kategoryczne (zmienne X10 oraz X11) oraz przedziałowe (wszystkie pozostałe poza X17, które całkowicie usuneliśmy). Dla zmiennych przedziałowych zastosowaliśmy uzupełnienie wartości brakujących za pomocą metody SimpleImputer z parametrem add indicator = True, który zachowuje informację o tym, które dane są brakujące (metoda uzupełniania danych pozostała domyślna, ponieważ w kolejnych krokach będziemy wybierać najbardziej optymalną). Następnie zmienne kategoryczne są kodowany z wykorzystaniem metody OneHotEncoding z paramterem drop = "frist", co zmniejszy nam nieco liczbę kolumn, nie tracąc przy tym żadnej informacji. Dla zmiennych przedziałowych w identyczny sposób zostało zastosowane uzupełnianie brakujących danych, a dodatkowo zostały one przeskalowane przy pomocy obiektu StandardScaler, co powinno poprawić działanie modeli. Następnie przy pomocy funkcji ColumnTransforem obróbka obu typów zmiennych została połączona w jedną całośc.

2 Tworzenie modeli

2.1 Regresja logistyczna

2.1.1 Regularyzacja l1

Pierwszym model, który postanowiłem wykorzsytać do klasyfikacji klientów była regresja logistyczna. Zacząłem od modelu z regularyzacją l1, by móc sprawdzić, które ze zmiennych okażą się nieprzydatne (otrzymają zerowy współczynnik). Solver, który wykorzystałem, to 'liblinear'. Następnie przy pomocy funkcji RandomizedSearchCV przeprowadziłem wyszukiwanie najbardziej optymalneych parametrów z scoringiem balanced accurac i 5-krotną kroswalidacją. Parametry były optymalizowane spośród następujących zmiennych i wartości: współczynnik C modelu: [0.25,0.5,0.75,1,2,3,5,10,25,40,50,100,150,500,1000,2000,10000], maksymalna liczba iteracji modelu: [100,1000,5000,10000,20000,50000], strategia uzupełniania braków danych dla zmiennych numerycznych: ['constant', 'most frequent'], strategia uzupełniania braków danych dla zmiennych kategorycznych ['mean', 'median', 'most frequent']. Optymalnymi parametrami okazał się zestaw o wartościach odpowiednio: 0.5, 10000, mean i constant. Uzyskana wartość blanced accuracy w 10-krotnej kroswalidacji to 0.7515.

2.1.2 Elasticnet

Wykorzystując model regresji logistycznej postanowiłem zbudować jeszcze jeden model, tym razem z wykorzystując karę ElasticNet. Przeprowadzając identyczną jak dla regularyzacij procedurę zbudowałem model. Jedyną różnicą było wybranie solvera saga oraz dodanie do optymalizowanych hiperparametrów l1 ratio z wartościami [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]. Optymalny zestawem okazaly się wartości 5000 iteracji, l1 ratio = 0.9, C = 0.25, mean do imputowania zmiennych numerycznych i moxt frequent do imputowania zmiennych kategorycznych. Sprawdzając wynik ponownie w ten sam sposób otrzymany rezultat balanced score wyniósł 0.7544 zatem odrobinę, więcej niż z regularyzacją l1.

2.2 Las losowy

Kolejnym modelm, który zdecydowałem się sprawdzić został las losowy. W tym celu ponownie utworzyłem pipepline składający się z wcześniej utworzonego preprocessingu oraz modelu lasu losowego, a nastepinie przeprowadziłem poszukiwania najbarzdiej optymalnego zestawu hiperparametrów. Wybrane wartości oraz parametry, to: liczba drzew [100,375,380,400,425,450,475,500], maksymalna głębokość [6,7,8,9], minimalna liczba osberwacij do podziału [2,5,7,10,11,12,14], minimalna liczba obserwacij w liściu [2,3,4,5,6,7,8], maksymalna liczba cech [None, 'sqrt', 'log2'], bootstrap True/False oraz sposób imputacij dla danych kategorycznych i numerycznych: ['mean', 'median', 'most frequent']. Uzyskany wynik balanced score w 10-krotnej kroswalidacji to 0.7340.

2.3 Gradeint Boosting

Następnym modelem, który został przeze mnie przetesowant to Gradient Boosting. Procedruowa budowy modelu przebiegała identycznie. Testowany zestaw hiperparametrów, to: strategia wypełniania braków danych numerycznych i kategorycznych mean/median/most frequent, liczba drzew [50,100,200,300,400,500], współczynnik uczenia [0.01,0.05,0.1,0.15,0.2,0.25,0.3], maksymalna głębokość [3,4,5,6,7], minimalna liczba obserwacji w podziale [8,9,10,11,12,13,14,15,16], minimalna liczba obserwacji w liściu [1,2,3,4], maksymalna

liczba cech [None, 'sqrt', 'log2']. A optymalnymi wartościami okazały się odpowiednio median, constant, 200, 0.01, 5, 13, 3, log2. Uzyskany wynik w 10-krotnej kroswalidacji to 0.7471.

2.4 SVM

Ostatnim przetestowanym modelm został SVM. Z racji, że model nie jest oparty na drzewach, ani nie nadaje wagi zmiennym postanowiłem połączyć ten algorytm z metodą PCA, żeby zapobiec problemom z zależnościami między zmiennymi. Następnie bując pipeline składający się z preprocessingu danych, PCA oraz samego modelu za pomocą funkcji RandomizedSearchCV przetestowaliśmy kombinacje z następującego zestawu parametrów: uzupełnianie braków danych numerycznych/ kategorycznych mean/median/most frequent, C [0.25,0.75,3,5,10,25,50,100,150,200,500,1000], jądro ['linear', 'poly', 'rbf', 'sigmoid'], stopnie [2,3,4,5,6,7], gamma ['scale', 'auto'] liczba komponetów pca [12,13,14,15,16,17,18,19,20]. Najlepszy zestaw parametrow to liniowe jądra, gamma 'scale' 7 stopni, C=1000 oraz stretegia mean i 17 komponentów PCA. Uzyskany wynik w 10-krotnej kroswalidacji to 0.7442.

3 Wnioski i ostateczny wybór

Wszsytkie spośród zbudowanych modeli osiągneły podobne wyniki. Model, który postanoiwłem wybrać została regresja logistyczna z karą elasticnet. Model ten osiągnął najlepszy wynik na zbiorze testowym, ponadto działa też najszybiciej i jest stosunkowo prosty w porównaniu do pozostałych. Pozwala też na interpretacje istotności dancyh oraz ma niewiele parametrów, co ułatwia dostrajanie modelu. Po wybraniu najlepszego modelu jeszcze raz przeprowadziłem uczenie, tym razem na całym dostępnym zbiorze X i (pliki z końcowką train) i przeprowadziłem przeszukiwanie najlepszego zestawu hiperparametrów, ponieważ model będzie mógł lepiej się dopasować na większym zbiorze danych. Ostateczny model prezentuje się następująco: maksymalna liczba iteracij: 5000, l1 ratio: 0.9, C: 0.1, uzupełnianie braków zmiennych numerycznych: mean, uzupełnianie brakóww zmiennych kategorycznych: constant. Model z takimi parametrami w aplikacji uzyskał wynik 0.7759, zatem wyższy niż podczas testów w kodzie.

```
{'reg__max_iter': 5000,
 'reg__l1_ratio': 0.9,
 'reg__C': 0.1,
 'pre__numerical__imputer__strategy': 'mean',
 'pre__categorical__imputer__strategy': 'constant'}
```

Rysunek 2: Ostateczny model