Credit Card Fraud Detection - dokumentacja końcowa projektu (K10)

Zaawansowane uczenie maszynowe

Rębacz Gabriel, Belniak Michał

25 stycznia 2021

Spis treści

1. Szczegółowa interpretacja tematu projektu

Wybranym zadaniem jest analiza problemu klasyfikacji na zbiorze danych *Credit Card Fraud Detection*[2] dostępnym w serwisie Kaggle.com. Zbiór zawiera informacje o transakcjach kartami kredytowymi, w tym informacje o kwocie transakcji, czasie transakcji oraz 28 parametrach o nieznanym znaczeniu, które zostały poddane transformacji poprzez analizę głównych składowych.

Zakres projektu obejmuje 4 podstawowe sfery:

- wnikliwą analizę zbioru danych,
- przetworzenie zbioru danych do postaci odpowiednich dla poszczególnych modeli,
- przygotowanie, uczenie i strojenie modeli wybranych spośród dostępnych w języku R,
- ocenę i porównanie modeli wraz z adekwatnymi wnioskami.

Zbiór danych należało przeanalizować pod kątem czynników które mają lub mogą mieć wpływ na sposób ich przygotowania oraz dobór parametrów modeli. Należało także określić sposób porównywania jakości modeli.

2. Opis wykorzystywanych algorytmów

W ramach projektu zostały wykorzystane trzy modele przedstawione poniżej. Do specyfikacji, trenowania i ewaluacji modeli wykorzystana została biblioteka *Tidymodels*.

2.1. Drzewo klasyfikacji

Drzewo klasyfikacji [4] to rodzaj drzewa decyzyjnego - narzędzia wspierającego decyzje, które wykorzystuje strukturę drzewiastą jako reprezentację. W węzłach znajdują się warunki dla wartości atrybutów, które kierują do jednego z możliwych wyników - rozgałęzień. Liście natomiast reprezentują klasę lub prawdopodobieństwo klas. Ścieżka od korzenia do liścia stanowi proces predykcji. O warunku w węźle drzewa decyduje wartość entropii wzajemnej bądź indeksu Giniego dla podziału. Wysokie wartości tych wskaźników oznaczają bardziej równomierny, a więc i korzystny, podział zbioru treningowego między węzły potomne. Zakończenie tworzenia kolejnych rozgałęzień i umieszczenie liścia zachodzi w momencie osiągnięcia kryterium stopu, zarówno wynikającego 'naturalnie' (na przykład brak próbek treningowych lub wszystkie przykłady z jednej klasy) jak i określonego przez implementującego.

W naszym projekcie R wykorzystaliśmy implementację drzewa decyzyjnego z pakietu *rpart*.

2.2. Regresja logistyczna

Regresja logistyczna [3] jest jedną z metod regresji która głównie znajduje zastosowanie jeśli przewidywana klasa może przyjąć tylko dwie wartości. Zgodnie z nazwą, model wykorzystuje funkcję logistyczną jako funkcję określającą wartość predykcji, a klasyfikacja odbywa się dzięki ustalonej wartości progowej, po przekroczeniu której wynikową klasą jest 1, a w przeciwnym przypadku 0.

$$h\Theta(x) = \sigma(Z) = \frac{1}{1 + e^{-Z}}; \tag{1}$$

Argumentem funkcji jest równanie liniowe wartości atrybutów próby postaci:

$$Z = \sum_{i=1}^{n} w_i a_i(x) + w_{n+1} \tag{2}$$

gdzie wartości $w_i, w_2 ... w_{n+1}$ są modyfikowane w procesie uczenia stanowiąc parametry modelu. W celu optymalizacji tych parametrów definiuje się funkcję kosztu.

$$L = Cost(h\Theta(x), c) = -c \log(h\Theta(x)) - (1 - c) \log(1 - h\Theta(x)), \tag{3}$$

Funkcja kosztu jest wykorzystywana do obliczenia jej gradientu w celu zastosowania metody gradientu prostego. Przy metodzie tej parametry zmienia się według poniższej reguły:

$$w_i = w_i - \alpha \frac{\delta L}{\delta w_i} \tag{4}$$

Gdzie α stanowi parametr kroku, który może decydować o szybkości zbiegania algorytmu do wartości optymalnych lub nawet utrudniać lub uniemożliwiać jej osiągnięcie.

W naszym projekcie R wykorzystana została implementacja modelu z pakietu *glm*.

2.3. Sieć neuronowa

Siecią neuronową [1] nazywany jest aproksymator w postaci modelu grafowego kierunkowego, w którym węzeł, nazywany neuronem, przetwarza wartości przekazywane poprzez wchodzące do niego krawędzie i przekazuje obliczoną wartość do krawędzi wyjściowych. Neurony podzielone są na warstwy, w których żaden neuron nie jest ze sobą połączony, natomiast każdy neuron posiada tyle krawędzi wejściowych, ile jest neuronów w warstwie poprzedzającej i tyle krawędzie wyjściowych ile jest neuronów w warstwie następnej. Krawędzie sieci posiadają wagi, które stanowią parametry w procesie uczenia.

Sposób przetwarzania wartości w neuronie polega na obliczeniu sumy iloczynów wartości na krawędzi wejściowej i odpowiadających im wag, a następnie na obliczeniu wartości funkcji aktywacji, która ustalana jest dla wszystkich neuronów w warstwie. Wyjściem ostatniej warstwy jest aproksymowana wartość

przybliżanej funkcji docelowej. Algorytm uczenia sieci opiera się na obliczaniu jakości aproksymacji sieci i wykorzystaniu algorytmu wstecznej propagacji gradientu do zaktualizowania wag sieci.

W naszym projekcie R wykorzystaliśmy implementację z pakietu *keras*.

3. Plan badań

3.1. Cel poszczególnych eksperymentów

Głównym celem była analiza porównawcza trzech modeli w problemie klasyfikacji pod kątem jakości predykcji klas i stopnia ich dopasowania do zbioru treningowego. Dla każdego modelu przeprowadzone zostały cztery eksperymenty. Uczenie modeli odbywało się na czterech różnych zbiorach treningowych, które różniły się sposobem przekształcenia zbioru. Każdy z modeli był następnie ewaluowany na tym samym zbiorze testowym, który został wyznaczony przed wszelkimi przekształceniami zbioru oraz który nie był uwzględniany w trakcie strojenia modelu. Poszczególne eksperymenty będą miały początkowo na celu strojenie parametrów algorytmów, tak aby porównanie modeli było jak najmniej obarczone niedostosowaniem ich parametrów. Strojenie nie odbywało się jednak dla sieci neuronowej ze względu na brak możliwości w *Tidymodels* strojenia parametrów, które uznaliśmy za warte strojenia (na przykład stopa uczenia). Strojenie wykorzystywało metodę optymalizacji Bayesowskiej wraz z 10-krotną walidacją krzyżową, a za ocenę jakości danych hiperparametrów służyło średnie pole pod krzywą ROC dla podzbiorów wyznaczonych dla walidacji krzyżowej. Po znalezieniu odpowiednich parametrów porównane zostały wyniki procesów treningowych na czterech różnych zbiorach treningowych:

- zbiór poddany podpróbkowaniu,
- zbiór poddany podpróbkowaniu oraz selekcji atrybutów,
- zbiór poddany nadpróbkowaniu,
- zbiór poddany nadpróbkowaniu oraz selekcji atrybutów.

Selekcja atrybutów odbyła się na zasadzie stworzenia lasu losowego złożonego z 50 drzew, a następnie wyborze tych atrybutów, dla których wartość *variable importance* była większa od jej średniej wartości dla wszystkich atrybutów. Selekcja atrybutów odbyła się przed operacjami podpróbkowania oraz nadpróbkowania.

3.2. Charakterystyka zbioru danych

Dane są w postaci tabelarycznej i nie zawierają one pustych wartości. Liczba wszystkich rekordów to 284,807, więc można wnioskować, że zbiór jest odpowiednio liczny. Wszystkie kolumny zawierają wartości numeryczne (parametry V1-V28 oraz czas, wartość transakcji i klasę). Czas i klasa są liczbami całkowitymi,

a pozostałe parametry liczbami rzeczywistymi. Zbiór klas jest dwuelementowy - 1 w przypadku oszustwa, a 0 w przypadku poprawnej transakcji. Parametry V1-V28 zostały poddane transformacji poprzez analizę głównych składowych, a ich interpretacja jest nieznana ze względu na poufność danych.

Zbiór jest mocno niezbalansowany - oszustwa stanowią jedynie 0.172% wszystkich transakcji. Konieczne było przetworzenie zbioru w celu jego zbalansowania. Zostały zastosowane dwie przeciwstawne metody - nadpróbkowanie(*oversampling*) metodą SMOTE, czyli zwiększenie liczby rekordów z klasą 1 za pomocą tworzenia nowych rekordów klasy mniejszościowej na podstawie istniejących rekordów oraz podpróbkowanie(*undersampling*), czyli wybranie losowo tylko części rekordów z klasą 0. Obydwie metody mają na celu stworzenie zestawu danych treningowych z równą liczbą rekordów z klasą 0 oraz 1. Dodatkowo, przeprowadziliśmy standaryzację wartości atrybutów czasu i kwoty.

3.3. Parametry algorytmów

Wpływ parametrów na jakość modeli nie został zbadany zgodnie ze wstępnymi założeniami. Ograniczenia wynikały z możliwości biblioteki *Tidymodels*. Część parametrów została wyznaczona za pomocą strojenia metodą optymalizacji Bayesowskiej. Badany był natomiast wpływ sposobu przetwarzania danych treningowych.

3.3.1. Drzewo klasyfikacji

W przypadku drzewa klasyfikacji strojone były parametry:

- koszt złożoności drzewa,
- maksymalna głębokość drzewa,
- minimalna liczba punktów danych w węźle wymagana do podziału.

3.3.2. Regresja logistyczna

W przypadku regresji logistycznej strojono współczynnik kary. Na podstawie eksperymentów dobrano regularyzację L2.

3.3.3. Gęsta sieć neuronowa

Nie odbyło się strojenie hiperparametrów sieci. Biblioteka *Tidymodels* umożliwia wyłącznie strojenie tak zwanego *dropout*, służącego do eliminacji nadmiernego dopasowania modelu do danych oraz strojenie liczby neuronów w warstwie ukrytej i współczynnika kary dla regularyzacji L2, z której, być może niesłusznie, zrezygnowaliśmy. Uczono sieci o 64 neuronach w warstwie ukrytej z funkcją aktywacji ReLU.

3.4. Miary jakości i procedury oceny modeli

Głównym sposobem oceny jakości było pole pod krzywą ROC, jako jeden z najpopularniejszych sposobów oceny jakości predykcji binarnej. Uwzględnione zostały także miary jakości: dokładność oraz współczynnik F.

4. Uzyskane wyniki

W trakcie badań uzyskano wyniki przedstawione poniżej. Natomiast dokładna analiza danych znajduje oraz wyniki się w pliku *Credit-card-fraud-detecion-–analysis.html*.

4.1. Regresja logistyczna

Wykorzystano model regresji logistycznej z metodą regularyzacji L2. Poniżej przedstawiono zestawienie wartości miar jakości modelu dla regresji logistycznej:

Zbiór	Dokładność	f-score	Pole pod krzywą ROC
Undersampling	0.9682308	0.9838306	0.9821616
Undersampling z selekcją atrybutów	0.9826969	0.9912581	0.9699946
Oversampling	0.9795228	0.9896374	0.9801168
Oversampling z selekcją atrybutów	0.9778795	0.9887982	0.9502381

4.2. Drzewo klasyfikacji

Poniżej przedstawiono zestawienie wartości miar jakości modelu dla drzewa klasyfikacji:

Zbiór	Dokładność	f-score	Pole pod krzywą ROC
Undersampling	0.9121922	0.9539996	0.9558970
Undersampling z selekcją atrybutów	0.9197062	0.9580991	0.9654289
Oversampling	0.9638207	0.9815478	0.9611257
Oversampling z selekcją atrybutów	0.9651128	0.9822178	0.9378991

W przypadku drzewa wyraźnie widać poprawę działania w przypadku zbiorów poddanych oversampling w stosunku do modeli, które przetwarzały zbiory poddane działaniu undersampling. Wskazują na to dokładność oraz f-score. Jednakże, różnice zanikają w przypadku pola pod krzywą ROC. Oznacza to, że model odpowiadający zbiorowi z nadpróbkowaniem wykrywa oszustwa skuteczniej, lecz tylko dla określonego progu użytego do obliczenia dokładności i f-score (0.5). Dla innych progów odcięcia jakość modeli w obydwu przypadkach może być bardziej zbliżona. Warto także zaznaczyć, że dla każdego rodzaju zbioru,

wartości zarówno pola pod krzywą ROC jak i pozostałych miar są niższe niż dla modelu regresji logistycznej, co czyni regresję logistyczną względnie lepszym modelem.

4.3. Sieć neuronowa

Wykorzystano perceptron dwuwarstwowy z 64 neuronami w warstwie ukrytej oraz optymalizator Adam. Funkcją aktywacji dla neuronów w warstwie ukrytej była funkcja ReLU. Poniżej przedstawiono wartości miar jakości modelu:

Zbiór	Dokładność	f-score	Pole pod krzywą ROC
Undersampling	0.9375009	0.9676885	0.9666849
Undersampling z selekcją atrybutów	0.9557590	0.9773349	0.9583862
Oversampling	0.9992135	0.9996061	0.8836914
Oversampling z selekcją atrybutów	0.9901687	0.9950512	0.9451059

W przypadku zbioru poddanemu nadpróbkowaniu uzyskano najlepsze spośród wszystkich modeli dokładność oraz f-score. Są one niezwykle bliskie wartości 1, zarówno w przypadku zbioru z selekcją atrybutów jak i bez. Natomiast, pole pod krzywą ROC ma najniższe do tej pory wartości w przypadku tych zbiorów, szczególnie w przypadku zbioru z wszystkimi atrybutami, co jest zaskakujące, gdyż w przypadku regresji logistycznej uzyskano odwrotny efekt. Lepiej pod tym względem wypadają zbiory poddane operacji undersampling. Jeśli chodzi o porównanie z pozostałymi modelami, sieć neuronowa radzi sobie z wykrywaniem oszustw najlepiej spośród wszystkich modeli, o ile wytrenowano je na zbiorze poddanym nadpróbkowaniu oraz próg odcięcia wyniesie 0.5. Natomiast, krzywa ROC jest najmniej korzystna spośród wszystkich modeli.

5. Podsumowanie

Dokładne wyniki zostały przedstawione w pliku: *Credit-card-fraud-detecion–analysis.html*, który znajduje się w załączonym źródłach.

Biorąc pod uwagę najważniejsze kryterium, to znaczy pole pod krzywą ROC, najlepiej z zadaniem klasyfikacji poradził sobie model klasyfikacji logistycznej. Natomiast w przypadku drzewa klasyfikacji oraz sieci neuronowej werdykt nie jest jednoznaczny, gdyż wynik zależy od wykorzystywanego zbioru treningowego. Uzyskano zbliżone wyniki dla wszystkich zbiorów z wyłączeniem zbioru poddanemu nadpróbkowaniu - tam sieć neuronowa uzyskała dużo gorsze pole pod krzywą ROC.

Jeśli chodzi o selekcję atrybutów, nie wpływa ona znacząco na jakość modeli. Jedynie dla regresji logistycznej widać nieznaczną poprawę wartości pola pod krzywą ROC. Natomiast, z pewnością zaletą takiego rozwiązania jest krótszy czas trenowania modelu.

Ciekawe wyniki uzyskano w przypadku sieci neuronowej. Wyraźne różnice występują dla dokładności i wartości współczynnika F między zbiorami poddanymi nadpróbkowaniu oraz podpróbkowaniu. W przypadku nadpróbkowania wartości wskazanych miar są niezwykle wysokie, niemal sięgające wartości 1, więc wykrywanie oszustw jest w tym przypadku bardzo czułe.

Bibliografia

[1] missinglink.ai Contributors. *The Complete Guide to Artificial Neural Networks: Concepts and Models*. [Online; accessed 16-November-2020]. URL: `https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/`.

[2] *Credit Card Fraud Detection*. Dostęp zdalny (15.11.2020): `https://www.kaggle.com/mlg-ulb/creditcardfraud`. 2016.

[3] Saishruthi Swaminathan. *Logistic Regression — Detailed Overview*. [Online; accessed 16-November-2020]. 2018. URL: `https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc`.

[4] Wikipedia contributors. *Decision tree — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2020]. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Decision_tree&oldid=983253586`.