Praca domowa 2: Walidacja KM2b BezNazwy

deepRession

10.04.2022

1 Treść zadania:

Celem było przygotowanie code review grupy o nazwie: "BezNazwy".

2 Czy ten kod osiąga cel, który postawiono?

Kod osiąga cel postawiony w zadaniu, tj. przetwarza dane i jest w stanie wytrenować model z domyślnymi parametrami, jednakże część preprocessing'u działa w niedokładny i niesatysfakcjonujący sposób przez co wymaga poprawy.

3 Czy w kodzie są jakieś oczywiste błędy logiczne?

1. Preprocessing generuje wartości NA.
2. Przeprowadzenie de facto skategoryzowania zmiennych ciągłych (wliczając zmienną celu) w dodatku do jedynie trzech kubełków.
3. Brak możliwości wyborów rozmiaru zbiorów treningowego i testowego.
4. Kodowanie zmiennych kategorycznych za pomocą liczb - zmiennych ciągłych.

2. Pomysł takiej kategoryzacji jest bardzo naiwny. Rozważmy przypadek gdzie przewidujemy ceny posesji i jedną ze zmiennych jest jego powierzchnia. Większość mieszkań (pewnie koło 70 %) znajduje się w zakresie metrażowym 40-100 m2. Ale w zbiorze występować będą też takie mające powierzchnię rzędu 200-300 m2, a nie będą to wartości odstające ze względu na znaczną liczebność. W takim przypadku wszystkie mieszkania o metrażu 40-100 wylądują w jednym kubełku i ich powierzchnia nie będzie rozróżnialna pomimo tego że stanowią większość rynku! Taka kategoryzacja jest bardzo naiwnie przeprowadzona i zmienne ciągłe powinny zostać wyskalowane za pomocą np. standard scaler'a.
3. Jest to problematyczne ze względu na różne rozmiary zbiorów z którymi użytkownik może mieć do czynienia. Dla małych zbiorów stosunek 7:3 jest prawidłowy, jednakże w przypadku gdy operujemy na zbiorach o kilkudziesięciu tysiącach obserwacji, zbiór treningowy będzie nam maleć np. do 50% i chcielibyśmy mieć możliwość dobrania takiego właśnie podziału.
4. Wydaję się to co najmniej dziwne. Nie robią tego w taki sposób jak opisany na prezentacji. W ten sposób dodajemy relację większości zmiennym, które niekoniecznie muszą ją posiadać. Sam catboost jest przeznaczony właśnie by obsługiwać zmienne kategoryczne, więc nie powinniśmy ingerować w jego działanie.

4 Czy patrząc na wymagania zawarte podczas prezentacji są one w pełni zaimplementowane?

W większej mierze wymagania te zostały zaimplementowane, ale nie w pełni. Głównym brakującym elementem, o którym była mowa podczas prezentacji, jest brak zaimplementowanych hiperparametrów albo chociaż ich opisów (w celu ułatwienia ich zastosowania).

5 Czy kod jest zgodny z istniejącymi wytycznymi stylistycznymi?

Kod spełnia istniejące wytyczne stylistyczne. Autorzy nie mają zbyt długich linijek kodu, są one schludnie napisane, zmienne mają informatywne nazwy. Jedynym problemem jest sposób dokumentacji parametrów zaczerpnięty z Pythona i brak wykorzystania pakietów roxygen i devtools do wygenerowania profesjonalnej dokumentacji funkcji.

6 Czy są jakieś obszary, w których kod mógłby zostać poprawiony? (skrócić, przyspieszyć, itp.)

Odradzilibyśmy używanie funkcji **funs**. Jest ona przestarzała i dokumentacja R proponuje używanie **list()**. Dodalibyśmy też możliwość decydowania o rozmiarze testowego zbioru danych. Warto by też wyciągnąć tę funkcję poza preprocessing. Swoją drogą w miejscu podziału zbioru Zespół Budujący "zostawił" funkcję wypisującą wektor zmiennych objaśnianych zbioru treningowego, jest to niepotrzebne. Jeżeli kubełkujemy zmienne ciągłe w taki sposób to powinniśmy też dać chociaż możliwość decydowania o ilości kubełków. Czasami podział na 3 może być niewystarczający i wręcz zaburzać działanie modelu. Należałby tak poprawić to kubełkowanie, tak aby obsługiwało też zmienne ujemne, np. zamiast 0 użyć najmniejszej wartości z kolumny.

7 Czy dokumentacja i komentarze są wystarczające?

Dokumentacja istnieje, ale w wielu kwestiach jest niewystarczająca. Pomijając nieprofesjonalny sposób zapisania dokumentacji inspirowanym dokumentacją w Pythonie, a nie w R, rażącymi problemami są bardzo skrótowe opisy działania funkcji. Prominentnym przykładem jest brak szczegółowego opisu co dzieje się w ramach preprocessingu (szczątkowe komentarze niestety nie informują o dokładnym działaniu funkcji) i w efekcie jest on dla użytkownika czarną skrzynką (dopóki sam nie zbada jego działania). Ponadto w niektórych przypadkach brakuje opisu typu zmiennej (np. dla target nie wiemy czy mamy podać character, czy kolumnę) oraz co zwraca dana funkcja. Istotną wadą jest brak dokumentacji parametrów dla funkcji tworzącej model catboost, gdyż użytkownik pakietu musi przejrzeć dokumentacje innej paczki, aby dowiedzieć się jak działa opisywana funkcja.

8 Czy udało się odtworzyć zamieszone przykłady w kodzie?

Udało się, jednakże warto wspomnieć, że ramka nie została nam udostępniona w ramach folderu na GitHubie, więc dostęp był utrudniony. Dodatkowo jako test funkcji zastosowano tylko jedną ramkę danych, co okazało się niewystarczające aby sprawdzić poprawność kodu. W kwestii odtwarzania przykładów dość sporym błędem było pokubełkowanie zmiennej celu. Funkcja wykonała regresję dla tych numerycznych kategorii po czym autorzy zaokrąglili wyniki i porównali z tymi nadanymi. W ten sposób tracimy bezpowrotnie informację o realnej cenie posesji i zmieniamy regresję na coś co przypomina dziwną klasyfikację- otrzymane wyniki mogą być poza zbiorem kubełków.

9 Czy udało się użyć przygotowanych kodów na zbiorze danych z projektu?

Tak, jednakże nie w sposób jaki autorzy chcieli osiągnąć. Zmienną oznaczającą długość geograficzną, będącą zmienną ciągłą, musieliśmy zostawić bez ich domyślnego preprocessingu lub zmienić jej znak na dodatni. W przeciwnym przypadku wartości cześć wartości zmienia się na **NA** z powodu wartości ujemnych. Warto tutaj zaznaczyć, że całość przetestowaliśmy także na innych zbiorach i problem z wartościami NA powtórzył się, i zostały one wyprodukowane w ramach preprocessingu, Ponadto dla zbioru german credit data, model działał zaskakująco słabo, gdyż dla średniej wartości kredytu rzędu 2300 MAE wynosiło ponad 900.