Jakub Maracewicz
Michał Orzołek

Imputacja braków danych i predykcja cen wynajmu mieszkań w Poznaniu

Wprowadzenie

Celem projektu jest kompleksowa analiza braków danych w zbiorach pzn-rent-train i pzn-rent-test, imputacja braków danych, wykorzystanie przygotowanych danych do budowy modelu predykcyjnego (XGBoost) przewidującego cenę wynajmu w zbiorze testowym.

Wczytanie danych i konfiguracja środowiska

W pierwszym kroku zaimportowane zostały biblioteki do manipulacji danymi, modelowania, imputacji oraz wizualizacji. Ustawiono ziarno losowości RANDOM_STATE = 123, a następnie wczytano dane wejściowe. Zbiór treningowy ma wymiar (11297, 36), zbiór testowy (4842, 35), a przykładowy plik submission (4842, 2).

Raport sentyneli

W zbiorze treningowym wartość -999 występuje wyłącznie w kolumnie flat_area i pojawia się w 1069 przypadkach. W zbiorze testowym analogicznie flat_area zawiera 477 wystąpień wartości -999. Oznacza to, że część braków nie została oznaczona jako NaN, lecz zakodowana wartością sentynelową, co wymaga jawnej konwersji do braków przed dalszym przetwarzaniem.

Analiza braków danych

Wykonano zestawienie liczby i odsetka braków dla cech w zbiorze treningowym oraz wizualizacje struktury braków (missingno). Największe braki w danych treningowych występują w: flat_internet 3569 (31.59%), flat_for_students 1804 (15.97%), flat_closed_area 1804 (15.97%), flat_balcony 1368 (12.11%), quarter 1368 (12.11%), flat_garage 1317 (11.66%), flat_furnished 1317 (11.66%), flat_garden 1317 (11.66%), flat_dishwasher 1317 (11.66%), flat_rooms 1227 (10.86%). Taka struktura braków uzasadnia zastosowanie imputacji w części numerycznej oraz imputacji modalnej w zmiennych kategorycznych i binarnych.

Konwersja dat i sanity check na datach oraz relacjach cenowych

Wszystkie zmienne datowe date_activ, date_modif, date_expire zostały przekonwertowane do typu datetime z odpornością na błędne formaty (errors="coerce"). W zbiorze treningowym wykryto 1 przypadek niepoprawnej kolejności dat (date_activ > date_modif lub date_modif > date_expire). Następnie

wyznaczono listing_duration jako różnicę date_expire minus date_activ. W zbiorze treningowym średnia listing_duration wynosi 131.0486 dnia, mediana 30 dni, maksimum 2184 dni, a liczba obserwacji z listing_duration ≥ 365 wynosi 1096. W zbiorze testowym średnia wynosi 136.6411 dnia, mediana 30 dni, maksimum 2039 dni, a listing_duration ≥ 365 występuje w 474 przypadkach. Dla obserwacji z listing_duration ≥ 365 ustawiono listing_duration jako brak (NaN) oraz utworzono znacznik duration_anomaly.
Dodatkowo przeprowadzono kontrolę wartości cen i opłat: w zbiorze treningowym nie stwierdzono price ≤ 0, flat_rent < 0 ani flat_deposit < 0, a w zbiorze testowym (bez price) nie stwierdzono flat_rent < 0 ani flat_deposit < 0. Zidentyfikowano 18 przypadków anomalii flat_rent > price w zbiorze treningowym; utworzono zmienną rent_over_price_anomaly i w tych obserwacjach wyzerowano informację po stronie cechy ustawiając flat_rent = NaN, pozostawiając price jako zmienną docelową. W zbiorze testowym ustawiono rent_over_price_anomaly = 0, aby zachować spójność schematu cech między train i test.

Czyszczenie flat_area

Zmiennej flat_area nadano zestaw reguł sanity: wartości -999 potraktowano jako sentynel, wartości ≤ 5 jako zbyt małe, a wartości > 200 jako zbyt duże dla typowych metraży mieszkań. W zbiorze treningowym przed czyszczeniem flat_area miało minimum -999, średnią -61.9304 i maksimum 1000; liczba wartości ≤ 5 wynosiła 1069, a wartości > 200 wystąpiły 3 razy. W zbiorze testowym analogicznie minimum -999, średnia -66.1137, maksimum 2200; wartości ≤ 5 było 477, a > 200 wystąpiło 1 raz. Po czyszczeniu w zbiorze treningowym flat_area przyjmuje wartości od 8 do 196, ma średnią 49.0802 i odnotowano 2289 braków, natomiast w zbiorze testowym zakres to 8–168, średnia 48.6699 i 985 braków. Równolegle utworzono flagi flat_area_sentinel, flat_area_too_small i flat_area_too_big, aby model mógł odróżnić brak wynikający z czyszczenia od braków pierwotnych.

Czyszczenie flat_rooms i konstrukcja area_per_room

Dla flat_rooms zastosowano reguły: wartości ≤ 0 oraz > 10 uznano za anomalne i ustawiono jako brak. Przed czyszczeniem w zbiorze treningowym odnotowano minimum -9 i maksimum 11, a liczba wartości ≤ 0 wynosiła 1111 oraz > 10 wynosiła 4; w zbiorze testowym minimum -9, maksimum 8, a liczba wartości ≤ 0 wynosiła 454. Następnie zbudowano cechę area_per_room = flat_area / flat_rooms. Dla zbioru treningowego przed czyszczeniem area_per_room miało minimum 1.125, średnią 24.2375 i maksimum 80; wykryto 21 przypadków area_per_room < 5. Po czyszczeniu ustawiono area_per_room < 5 lub > 80 jako braki; w zbiorze treningowym po tej operacji area_per_room ma minimum 5, średnią 24.2913, maksimum 80, a liczba braków w area_per_room wynosi 3213 (co obejmuje zarówno braki wynikające z filtrów, jak i braki propagowane z flat_area lub flat_rooms).

Spójność zmiennej flat_furnished względem wyposażenia

Sprawdzono niespójności typu flat_furnished = False przy jednoczesnej obecności wyposażenia (flat_fridge, flat_cooker, flat_oven, flat_washmachine, flat_dishwasher, flat_television). W zbiorze treningowym wykryto 1844 takie przypadki; utworzono wskaźnik furnished_inconsistency oraz ustawiono flat_furnished jako brak w obserwacjach niespójnych. Po tej korekcie liczba braków w flat_furnished w zbiorze treningowym wynosi 3161.

Czyszczenie quarter

Zmienna quarter zawiera zarówno braki, jak i niespójności zapisu. W zbiorze treningowym liczba braków wynosi 1368, a w testowym 562. Zastosowano standaryzację (lowercase, strip), a następnie utworzono quarter_clean, w którym braki mapowane są na wartość "unknown", natomiast rzadkie kategorie z częstością < 20 (wyznaczone na podstawie TRAIN) mapowane są do "other". W zbiorze treningowym najczęstsze wartości quarter_clean to: grunwald 1473, unknown 1368, centrum 944, jeżyce 855, rataje 771, wilda 767, piątkowo 699, stare miasto 675, winogrady 643, łazarz 464, nowe miasto 335, naramowice 334, other 191.

Feature engineering i przygotowanie danych do modelowania

Na kopiach zbiorów zbudowano cechy pochodne z dat (time_to_modif, activ_year, activ_month, activ_dow), cechy z tytułu ogłoszenia (ad_title_len, ad_title_words) oraz markery braków dla kluczowych zmiennych numerycznych (flat_area_missing, flat_rooms_missing, flat_rent_missing, flat_deposit_missing, building_floor_num_missing). Zmiennych surowych datowych oraz surowych tekstów (w tym ad_title i ad_title_filled) nie wykorzystano bezpośrednio w modelu. W dalszej części zdefiniowano podział na zmienne numeryczne i kategoryczne, uzyskując 26 zmiennych numerycznych i 28 kategorycznych.

Imputacja i kodowanie zmiennych

Dla zmiennych numerycznych zastosowano IterativeImputer (schemat MICE) z estymatorem ExtraTreesRegressor (n_estimators = 50) oraz max_iter = 10, a jako strategię startową przyjęto medianę. Dla zmiennych kategorycznych użyto imputacji modalnej (most_frequent) oraz OneHotEncoder z handle_unknown = "ignore". Kluczowa zmiana względem wcześniejszych wersji polega na tym, że dopasowanie imputera i kodowania wykonywane jest w sposób kontrolujący wyciek informacji, czyli transformacje są uczone na danych treningowych (lub foldach treningowych w CV), a dopiero potem stosowane do walidacji i testu; dopasowanie na pełnym zbiorze wykonywane jest dopiero na etapie modelu finalnego do predykcji testu.

Wyniki modeli

Jako punkt odniesienia zbudowano model liniowy (LinearRegression) uczony na

logarytmie zmiennej docelowej (log1p(price)), a następnie przetransformowany wstecz na skalę ceny. Dla podziału train valid (test_size = 0.2, random_state = 123) uzyskano wyniki: RMSE = 377.47 oraz MAE = 264.65.
Następnie zastosowano XGBoost (XGBRegressor) z losowym przeszukiwaniem hiperparametrów (RandomizedSearchCV). Hiperparametry losowane były z uprzednio zdefiniowanych zbiorów wartości, a procedura obejmowała n_iter = 40 oraz walidację krzyżową cv = 3, przy funkcji celu neg_mean_squared_error liczonej na skali log(price). Najlepszy zestaw parametrów to: subsample = 0.8, reg_lambda = 0.1, reg_alpha = 0.1, n_estimators = 1000, min_child_weight = 5, max_depth = 8, learning_rate = 0.03, gamma = 0.0, colsample_bytree = 0.6. Odpowiadający temu wynik CV RMSE na skali log(price) wyniósł 0.1606.
Model XGBoost z najlepszymi parametrami, bez early stopping, osiągnął na zbiorze walidacyjnym (po transformacji expm1 do skali cen) RMSE = 303.25 oraz MAE = 207.94.
Dodatkowo zastosowano early stopping (early_stopping_rounds = 50) z metryką rmse, uzyskując best_iteration = 695. Dla wersji z early stopping wyniki na walidacji wyniosły RMSE = 303.32704606579347 oraz MAE = 208.30503845214844. Widać, że early stopping nie poprawił jakości w tej konfiguracji, ponieważ RMSE pozostał praktycznie na tym samym poziomie, natomiast MAE wzrosło względem wariantu bez early stopping, co sugeruje, że skrócenie liczby drzew w tym przypadku nie przełożyło się na lepszą generalizację w metryce błędu bezwzględnego.
Ostatecznie wytrenowano model finalny i wygenerowano predykcje dla zbioru testowego, zapisując plik submission_xgb_log_es.csv o wymiarze (4842, 2). W pliku wynikowym nie występują braki danych w kolumnach ID oraz TARGET.