# Pipeline Machine Learningowy do przewidywania cen mieszkań w Polsce

Twój kod to kompletny pipeline ML obejmujący: eksplorację danych, inżynierię cech, preprocessing, uczenie modeli, tuning hiperparametrów, analizę błędów, interpretację (SHAP) oraz predykcję na nowych danych. Poniżej znajduje się szczegółowy opis kroków oraz ich ocena.

Krok po kroku: Co się dzieje?

0. Importy

Wszystkie potrzebne biblioteki do pracy z danymi, uczenia maszynowego, wizualizacji, optymalizacji i interpretacji modelu.

1. Wczytanie danych

Szukasz plików CSV w katalogu data/ lub wskazanym przez użytkownika.

Wczytujesz i łączysz wszystkie tabelki w jedną ramkę danych df.

Ocena: ✔️ Dobrze. Automatyczne łączenie kilku plików ułatwia rozbudowę zbioru.

2. Eksploracyjna analiza danych (EDA)

Wyświetlanie pierwszych wierszy, informacji o kolumnach, statystyk, unikalności.

Analiza braków danych.

Wizualizacje: histogram cen, boxploty cen ogólnie i w miastach, macierz korelacji.

Ocena: ✔️ Dobrze. Klasyczne EDA z przyjaznymi wykresami.

3. Usuwanie najwyższych wartości cenowych w każdym mieście

Funkcja usuwa najwyższe X% mieszkań (tu: 0.5%) w każdej grupie-miasto.

Ocena: ✔️ Dobre podejście do outlierów. ⚠️ Możesz rozważyć inne metody identyfikacji outlierów, np. IQR lub Z-Score.

4. Uzupełnianie braków danych (KNNImputer), osobno dla każdego miasta

Każde miasto przetwarzane oddzielnie dla wybranych cech liczbowych.

Ocena: ✔️ Plus za grupowanie po mieście! ✔️ KNNImputer lepszy niż proste średnie.

5. Leave-One-Out Encoding dla miasta

Zamiast one-hot kodowania – miasto zamieniane na średnią cenę, uwzględniając wyłączenie danego rekordu.

Ocena: ✔️ Bardzo dobre podejście redukujące wymiar i minimalizujące overfitting.

⚠️ Uważaj na wyciek targetu – tutaj dobrze rozwiązane przez leave-one-out.

6. Dwa różne pipeline preprocessingu

Random Forest: one-hot encoding dla miasta i cech binarnych.

XGB/SHAP: LOO encoding dla miasta, reszta binarna jako dummies.

Usuwanie zbędnych kolumn, uzupełnianie braków, konwersja boolean na 'yes'/'no'.

Ocena: ✔️ Dobrze, że masz dwa niezależne pipeline dla różnych modeli.

⚠️ Konwersja 'yes'/'no' zamiast 0/1 może być zbędna – warto przemyśleć.

7. Podział na zbiory treningowe/walidacyjne/testowe

0.6 train / 0.2 val / 0.2 test.

Pipeline dla RF: bez skalowania.

Pipeline dla XGB/SHAP: ze skalowaniem (StandardScaler).

Ocena: ✔️ Klasyczny split, odpowiednie podejście do skalowania.

8. Trenowanie modeli i ich ocena

Trenowanie XGB, RF, LinearRegression na 5-krotnej walidacji krzyżowej.

Ewaluacja RMSE/testset i R².

Ocena: ✔️ Bardzo dobrze!

⚠️ Przydałoby się raportowanie wyników walidacji, nie tylko testu.

9. Optymalizacja hiperparametrów XGB (Optuna)

Walidacja na zbiorze walidacyjnym, 50 prób wyszukiwania.

Ocena: ✔️ Optuna to nowoczesne rozwiązanie, dobrze dobrana metoda.

⚠️ Można rozważyć early stopping lub większy zakres prób.

10. Zapis modeli na dysku (‘joblib’)

Zapisywanie modeli wraz z metadanymi i listą cech.

Ocena: ✔️ Super praktyka, łatwo odtworzyć środowisko później.

11. Analiza błędów

Mierzone metryki: MAE, RMSE, MedAE, MAPE, % błędów poniżej konkretnej kwoty.

Wizualizacje błędów i rozkładu przewidywań.

Ocena: ✔️ Bardzo dobry krok, pomaga wykryć skrajne przypadki.

12. Interpretacja modelu (SHAP)

Analiza najważniejszych cech, wykresy summary oraz dependence plots.

Ocena: ✔️ To jest wartościowy etap! Dobrze, że go uwzględniasz.

13. Predykcje na nowych danych

Sprawdzenie wymagalności cech i typów danych.

Automatyczne kodowanie one-hot/LOO.

Logowanie predykcji.

Ocena: ✔️ Dobre praktyki w zakresie input validation.

⚠️ Można rozszerzyć walidację granic wartości liczbowych.

Podsumowanie i rekomendacje

Co robisz dobrze?

✔️ Modularny, czytelny kod. ✔️ Dwa pipeline pod różne modele. ✔️ Sensowne przetwarzanie braków i kodowanie cech. ✔️ Pełna analiza błędów oraz interpretacja modeli. ✔️ Logowanie predykcji.

Co można poprawić?

Outlier removal – spróbuj metod dynamicznych (IQR/Z-score).

Binarne cechy – zamiast ‘yes’/‘no’, używaj bezpośrednio 0/1.

Uwzględnienie lokalizacji – wykorzystaj clustering geograficzny.

Cross-validation – zastosuj GroupKFold, by testować generalizację na nowe miasta.

Automatyzacja – zamknij wszystko w obiekt Pipeline w sklearn dla łatwej produkcji.

Ocena ogólna:

🔥 Bardzo dobry pipeline, świetnie nadający się zarówno do prototypowania, jak i produkcji!