Prognozowanie sprzedaży przy użyciu modelu uczenia maszynowego w Pythonie

Anna Wojtczak

BIG DATA, INŻYNIERIA I ANALIZY DANYCH Z WYKORZYSTANIEM JĘZYKA PYTHON

rok akademicki 2024/2025
Uniwersytet WSB Merito

1. Cel projektu

Budowa modelu regresyjnego do prognozowania liczby refundowanych czujników glukozy na podstawie danych historycznych. Model mógłby zostać wykorzystywany do prognozowania przyszłej sprzedaży refundowanych glukometrów, wspierania decyzji w zakresie planowania budżetu NFZ w zależności od lokalizacji i sezonu.

2. Opis danych

Źródło danych – dane otwarte: dane.gov.pl

https://dane.gov.pl/pl/dataset/4661,sprzedaz-czujnikow-do-monitorowania-stezenia-gluko
(udostępnione 7 stycznia 2025, zaktualizowane 22 maja 2025)

Zestawienie obejmuje dane za okres od stycznia do grudnia 2024 roku i dotyczy liczby refundowanych czujników do monitorowania stężenia glukozy w podziale na poszczególne placówki sprzedaży. Podano także informacje o wartości refundacji. Dane są pogrupowane ze względu na miesiąc sprawozdania ze sprzedaży danego wyrobu medycznego. Województwo odnosi się do oddziału wojewódzkiego NFZ świadczeniodawcy. Zestawienie przygotowano na podstawie danych Centrali NFZ według stanu na 20.05.2025 roku.

W zestawieniu uwzględniono następujące kody produktów:

- R.05.01.00 (hiperinsulinizm lub glikogenoza - dopłata 20%),
- R.05.01.01 (dzieci z cukrzycą - dopłata 20%),
- R.05.01.02 (kobiety z cukrzycą w ciąży - dopłata 30%),
- R.05.01.03 (osoby niewidome dopłata - 20%),
- R.05.02 (dorośli z cukrzycą dopłata - 30%).

3. Przygotowanie danych do analizy

Wczytanie i wstępna analiza danych:

```
import pandas as pd
df = pd.read_excel('czujniki_glukozy.xlsx', sheet_name='Czujniki')
print(df.head(5))
```

```
  Kod produktu  Miesiąc  ... Liczba wyrobów Kwota refundacji
0   R.05.01.00  styczeń  ...            749         133696.5
1   R.05.01.00  styczeń  ...             <5              NaN
2   R.05.01.00  styczeń  ...             <5              NaN
3   R.05.01.00  styczeń  ...             <5              NaN
4   R.05.01.01  styczeń  ...          23262        4745448.0
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 678 entries, 0 to 677
Data columns (total 8 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Kod produktu                  678 non-null    object
 1   Miesiąc                       678 non-null    object
 2   Województwo świadczeniodawcy  678 non-null    object
 3   Kod świadczeniodawcy          678 non-null    object
 4   Nazwa                         678 non-null    object
 5   REGON                         678 non-null    int64
 6   Liczba wyrobów                678 non-null    object
 7   Kwota refundacji              556 non-null    float64
dtypes: float64(1), int64(1), object(6)
memory usage: 42.5+ KB
```

- ilość badanych przypadków w zbiorze danych: 678, ilość kolumn:8
- dane jakościowe: ‘Kod produktu’, ‘Miesiąc’, ‘Województwo świadczeniodawcy’, ‘Kod świadczeniodawcy’, ‘Nazwa’
- dane ilościowe: ‘REGON’, ‘Liczba wyrobów’, ‘Kwota refundacji’
- unikalne wartości dla kolumn:

```
Unikalne wartości dla kolumn

Kod produktu                      5
Miesiąc                          12
Województwo świadczeniodawcy     16
Kod świadczeniodawcy             24
Nazwa                            12
REGON                            10
Liczba wyrobów                  263
Kwota refundacji                312
dtype: int64
```

- brak duplikatów w zbiorze danych
- braki w kolumnie ‘Kwota refundacji’ - 18%
- niepoprawny typ danych w kolumnie ‘Liczba wyrobów’

Czyszczenie danych:

- zmiana wartości "<5" w kolumnie "Liczba wyrobów" na 3 tj. średnią wartość z zakresu 1–4
- mapowanie miesięcy (styczeń = 1, ..., grudzień = 12)
- uzupełnienie braków w kolumnie ‘Kwota refundacji’

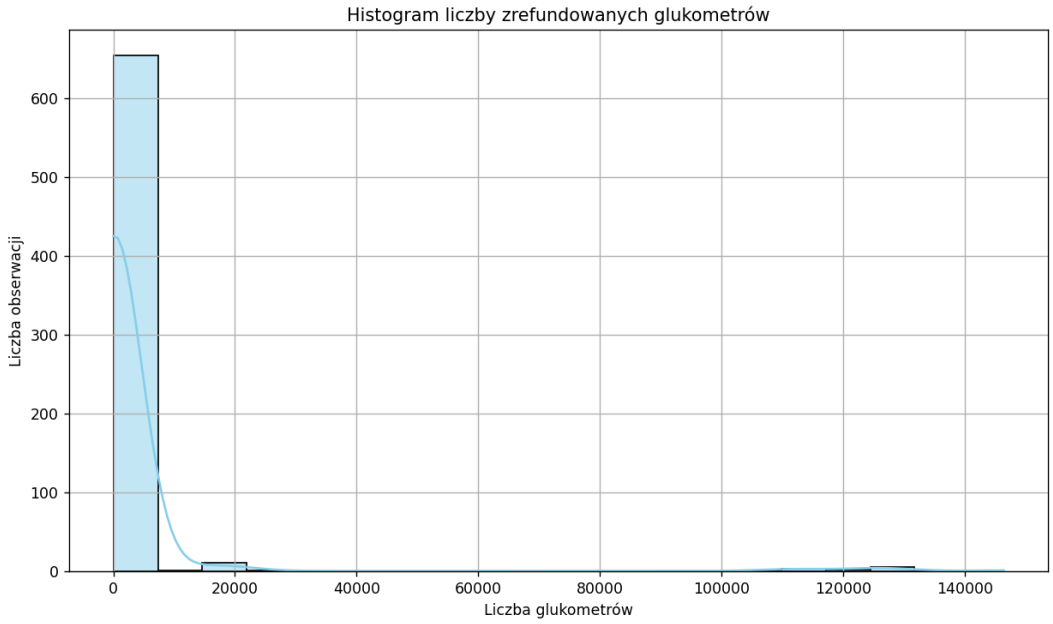
Zbiór danych zawierał braki w kolumnie „Kwota refundacji” przy liczbie czujników mniejszej niż 5. W celu uzupełnienia brakujących danych: przefiltrowano tabelę pod kątem niskiej liczby sprzedanych czujników, dodano nową kolumnę ‘cena_jednostkowa’, następnie ceny jednostkowe refundacji zostały uśrednione i pogrupowane w zależności od kodu produktu.

4. Eksploracyjna analiza danych (EDA)

Trend sprzedaży glukometrów w kolejnych miesiącach roku.

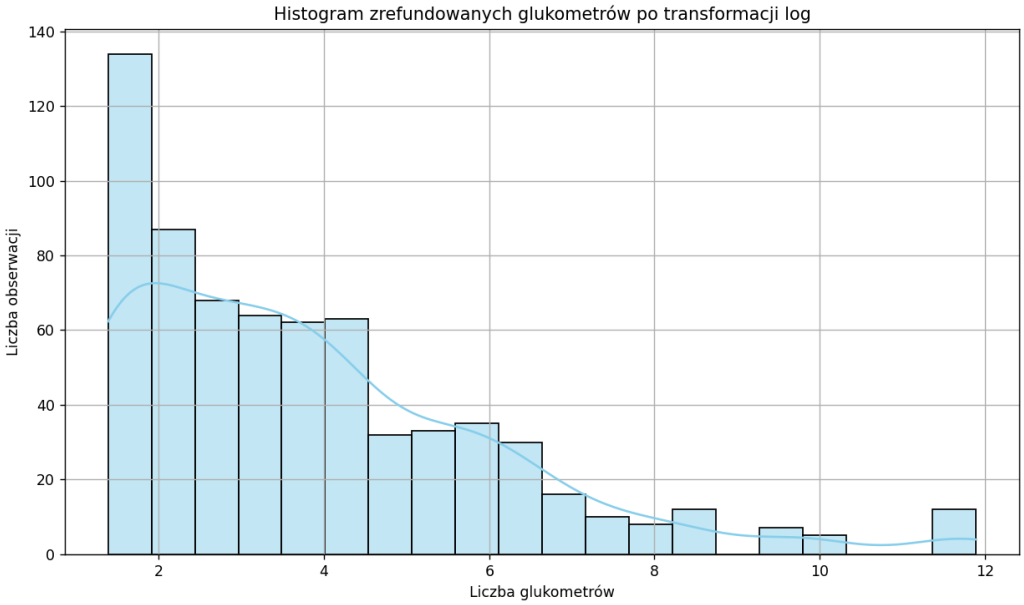
- Stabilny wzrost liczby refundacji od lutego do kwietnia może być efektem zwiększającej się dostępności glukometrów w placówkach. Natomiast widoczny wzrost w czerwcu może oznaczać zwiększone zapotrzebowanie przed sezonem wakacyjnym.
- Spadek w lipcu i sierpniu może sugerować mniej wizyt lekarskich i wiążące się z tym opóźnienia w realizacji świadczeń. Spadek w lutym może wskazywać na zakłócenia w dostawach.
- Widoczna jest sezonowość zachowań pacjentów z przewagą refundacji w drugiej połowie roku

Histogram liczby zrefundowanych gleukometrów.

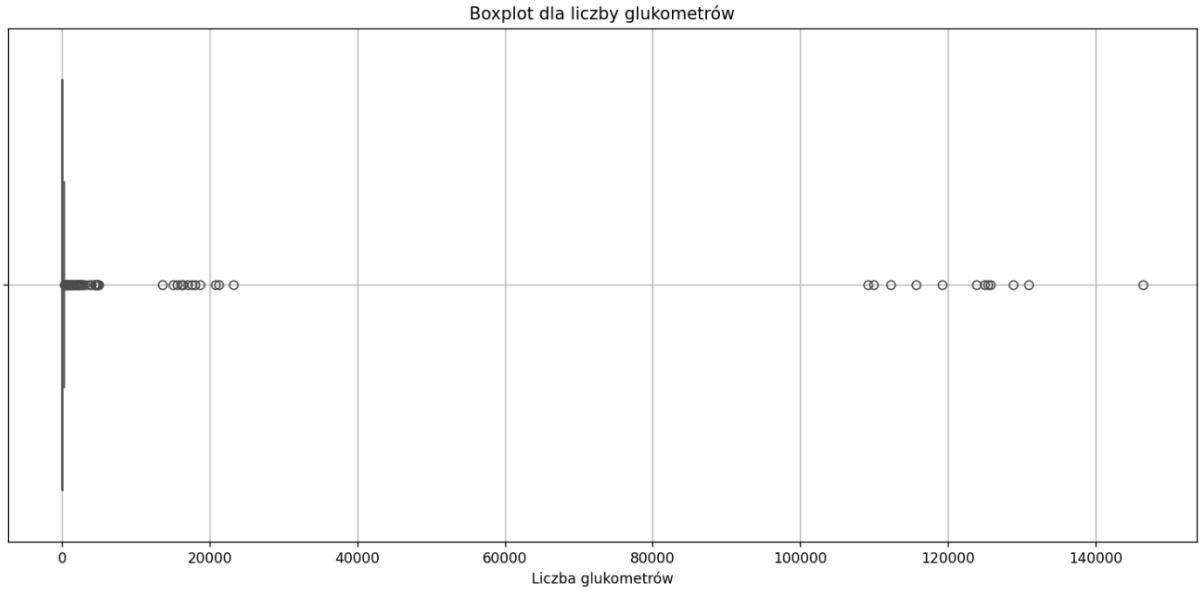


- Rozkład jest silnie prawoskośny, co oznacza, że większość liczby refundacji skupia się w zakresie poniżej 1000 sztuk, a tylko pojedyncze obserwacje mają znacznie większe wartości.
- Wiele placówek wykazało niewielką liczbę refundowanych glukometrów, co może oznaczać mniejsze punkty sprzedaży lub jednostki udzielające rzadko takich świadczeń.
- Pojedyncze placówki odpowiadają za bardzo wysoką sprzedaż glukometrów – mogą to być duże sieci.

Po zastosowaniu transformacji logarytmicznej (log1p) histogram jest bardziej zbliżony do rozkładu normalnego, co sugeruje lepsze przygotowanie zmiennej celu do modelowania regresyjnego.



Boxplot dla liczby sprzedanych glukometrów.

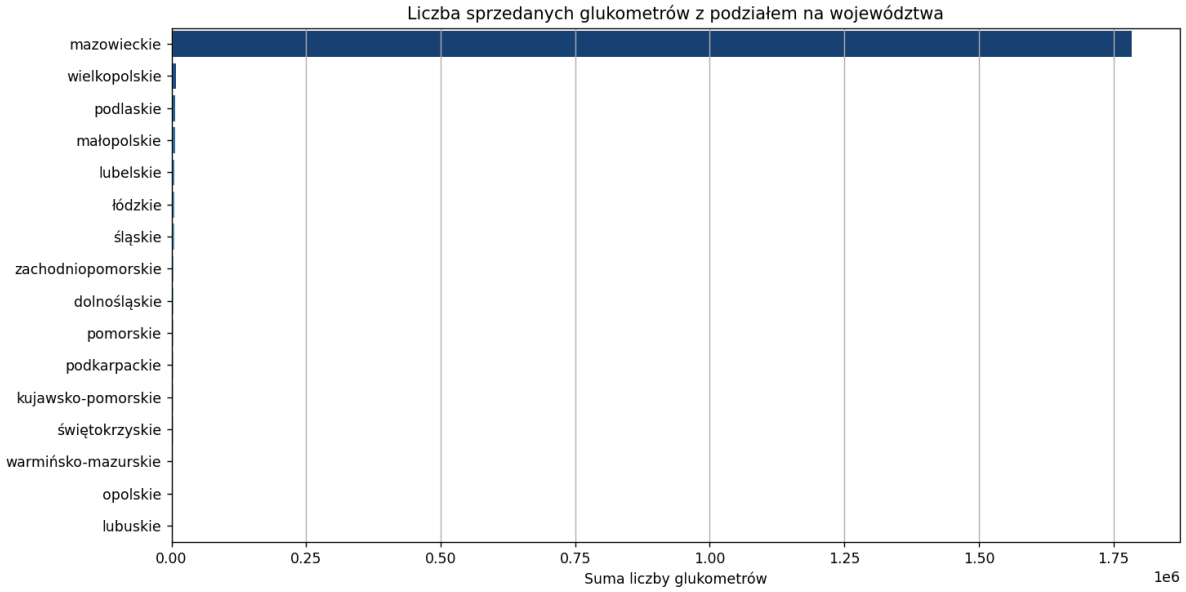


Zakres pomiędzy pierwszym a trzecim kwartylem (pudełko) znajduje się przy bardzo niskich wartościach, co oznacza, że 75% świadczeniodawców sprzedaje mało glukometrów (ok.150). Natomiast kilka placówek dominuje, ponieważ wykres pokazuje dużą ilość wartości odstających, są to placówki z bardzo wysoką sprzedażą, sięgającą ponad 140 000 glukometrów.

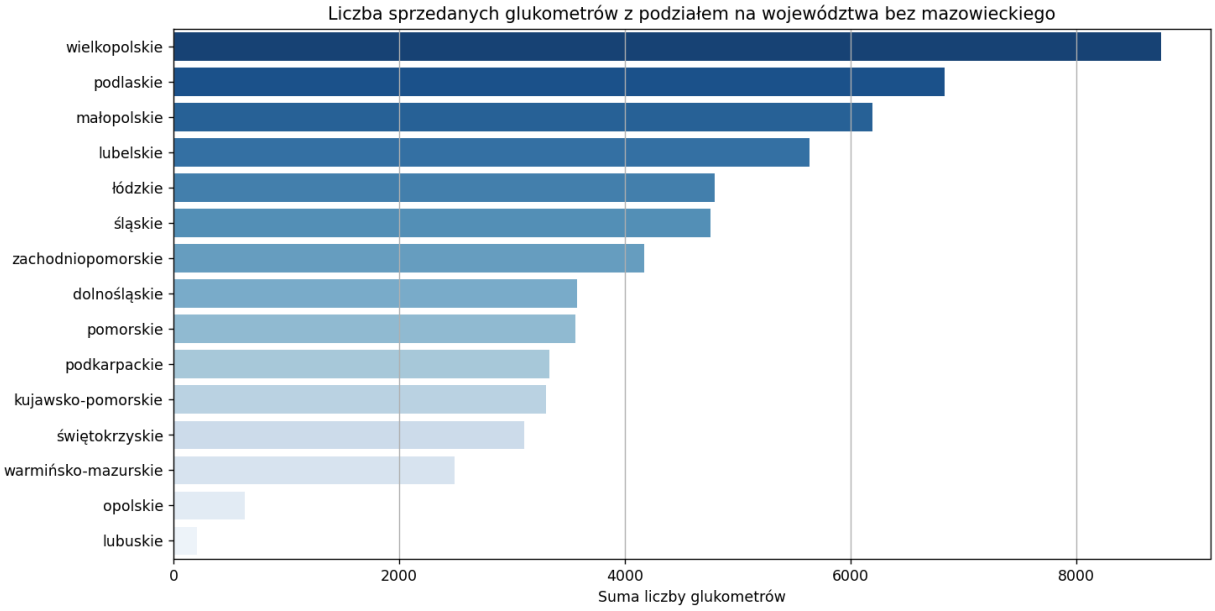
Dane są bardzo rozproszone, różnice w ilości sprzedanych glukometrów między placówkami są duże, co ponownie wskazuje na konieczność transformacji zmiennej celu.

Wykres liczby sprzedanych glukometrów z podziałem na województwa.

- Sprzedaż glukometrów w województwie mazowieckim przekracza 1,7 miliona sztuk z ogromną różnicą względem pozostałych, co świadczy o tym, że rynek refundowanych glukometrów w Polsce jest mocno scentralizowany.

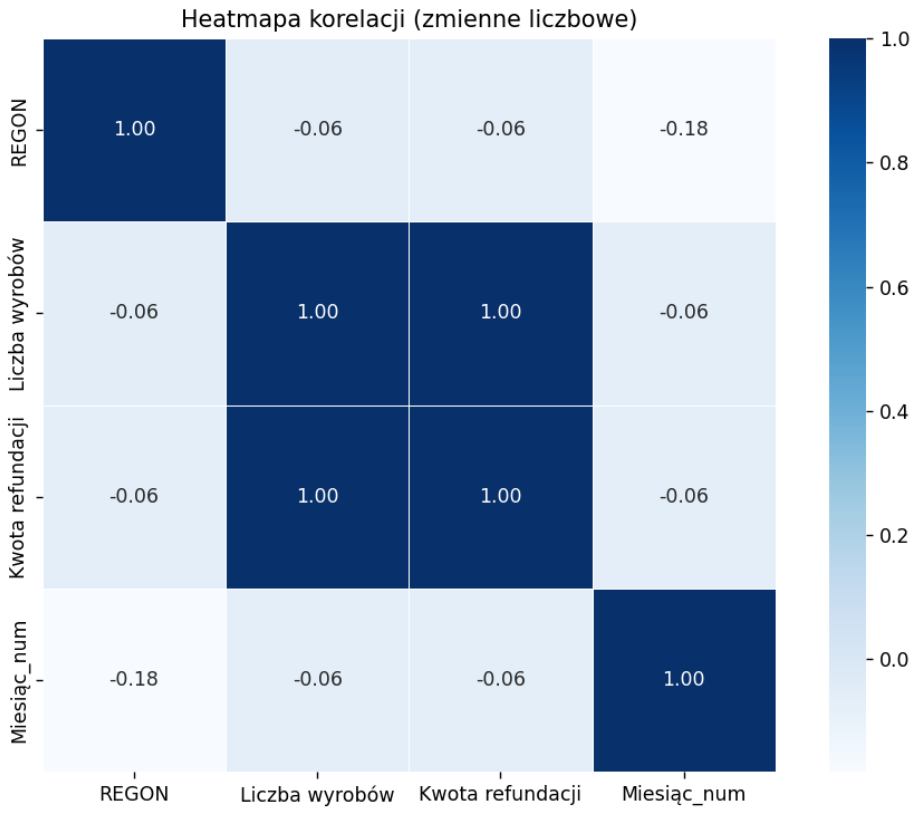


Wykres sprzedaży glukometrów w podziale na województwa, z wyłączeniem województwa mazowieckiego, na którym widać rzeczywistą strukturę regionalną w pozostałych częściach kraju:



- Największa sprzedaż glukometrów powyżej 8 000 sztuk występuje w woj. wielkopolskim.
- Wysoka sprzedaż w kilku regionach wschodnich i południowych: podlaskie, małopolskie, lubelskie — przekracza 5 000 sztuk.
- Najniższa sprzedaż: lubuskie, opolskie, warmińsko-mazurskie, co może wskazywać na słabszą dostępność, mniejszą liczbę świadczeniodawców lub mniejszą liczbę pacjentów z cukrzycą, ewentualne bariery lokalne (brak świadomości, słabsze finansowanie).
- Różnice te mogą posłużyć w planowaniu logistyki i alokacji zasobów, np. gdzie warto zwiększyć dystrybucję lub kampanię informacyjną.

Heatmapa korelacji na podstawie zmiennych liczbowych.



- Kwota refundacji bezpośrednio przekłada się na liczbę wyrobów.
- Zmienna ‘Miesiąc_num’ wnosi słabe, ale potencjalnie istotne informacje (np. sezonowość).
- Brak korelacji pomiędzy pozostałymi zmiennymi niezależnymi.

5. Przygotowanie danych do modelu

Analiza cech istotnych dla zmiennej celu, uwzględniająca eliminację nie informacyjnych zmiennych lub identyfikatorów:

- ‘Kwota refundacji’: wraz ze wzrostem refundacji, rośnie sprzedaż glukometrów
- ‘Miesiąc_num’: sezonowość
- ‘Województwo’: konkretne lokalizacje mają większą sprzedaż
- ‘REGON’ i ‘Kod produktu’, ‘Nazwa’: nie wnoszą informacji

One-hot encoding dla zmiennych kategorycznych:

```
df_encoded = pd.get_dummies(df, columns=['Województwo świadczeniodawcy'], drop_first=True)
```

Transformacja logarytmiczna zmiennej celu:

```
df_encoded['log_Liczba_wyrobow'] = np.log1p(df_encoded['Liczba wyrobów'])
```

Definicja zmiennych X i y:

```
X = df_encoded.drop(columns=['Liczba wyrobów', 'Kod produktu', 'Kod świadczeniodawcy', 'Nazwa','REGON', 'Miesiąc', 'log_Liczba_wyrobow'])
y = df_encoded['log_Liczba_wyrobow']
```

Podział danych:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. Trening modelu bazowego

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

7. Predykcja i odwrotna transformacja

Oszacowanie sprzedaży w skali logarytmicznej:

```
y_pred_log = model.predict(X_test)
```

Powrót do oryginalnej skali:

```
y_pred = np.expm1(y_pred_log)
```

Transformacja danych testowych do rzeczywistych wartości:

```
y_test_orig = np.expm1(y_test)
```

8. Ewaluacja i ocean modelu

```
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

rmse = mean_squared_error(y_test_orig, y_pred, squared=False)
print(f"RMSE: {rmse:.2f}")

mae = mean_absolute_error(y_test_orig, y_pred)
print(f"MAE: {mae:.2f}")

r2 = r2_score(y_test_orig, y_pred)
print(f"R²: {r2:.4f}")
```

```
RMSE: 585.69
MAE: 93.92
R²: 0.9993
```

- RMSE (Root Mean Squared Error) oznacza, że średni błąd predykcji modelu to około 586 sztuk sprzedanych glukometrów. RMSE jest czuły na wartości odstające — mimo to wynik jest niski w kontekście danych, gdzie niektóre wartości przekraczały 100 000 sztuk.
- MAE (Mean Absolute Error) pokazuje, że przeciętnie model myli się o około 94 glukometry.
- Współczynnik determinacji R^2 oznacza, że model jest niemal idealnie dopasowany do danych treningowych/testowych, wyjaśnia aż 99.93% zmienności.

9. Wizualizacja wyników

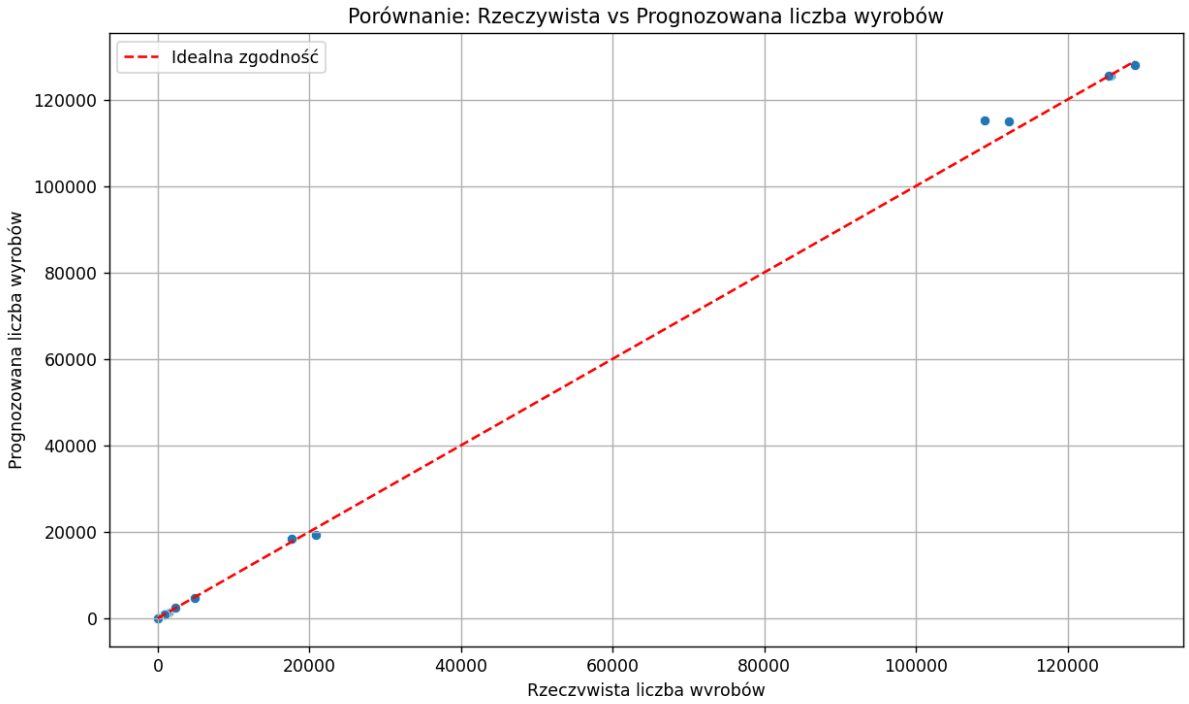
Wizualizacja rzeczywistej vs prognozowanej sprzedaży glukometrów:

```
comparison_df = pd.DataFrame({
    'Rzeczywista liczba wyrobów': y_test_orig,
    'Prognozowana liczba wyrobów': y_pred
}).reset_index(drop=True)
print(comparison_df.head(10))
```

	Rzeczywista liczba wyrobów	Prognozowana liczba wyrobów
0	251.0	268.414480
1	253.0	240.670439
2	19.0	16.925289
3	64.0	57.783387
4	4822.0	4755.796549
5	69.0	73.941296
6	360.0	349.601300
7	376.0	364.974495
8	59.0	59.415963
9	3.0	3.000000

Wykres porównujący rzeczywistą liczbę sprzedanych czujników glukozy (oś X) oraz prognozowaną liczbę czujników (oś Y).

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Rzeczywista liczba wyrobów', y='Prognozowana liczba
wyrobów', data=comparison_df)
plt.plot([0, max(y_test_orig.max(), y_pred.max())], [0,
max(y_test_orig.max(), y_pred.max())],
         color='red', linestyle='--', label='Idealna zgodność')
plt.title("Porównanie: Rzeczywista vs Prognozowana liczba wyrobów")
plt.xlabel("Rzeczywista liczba wyrobów")
plt.ylabel("Prognozowana liczba wyrobów")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Punkty rozmieszczone są blisko linii idealnej zgodności, co oznacza, że model dobrze przewiduje wartości. Małe odchylenia przy większych wartościach oznaczają, że model dobrze radzi sobie nawet z ekstremami. Nie występują punkty znacząco oddalone od linii, co świadczy o braku dużych błędów w predykcji.

10. Ulepszenie modelu: Strojenie hiperparametrów

Do optymalizacji parametrów modelu zastosowano klasę RandomizedSearchCV .
Definicja zbioru parametrów:

```
param_grid = {
    'n_estimators': [100, 200, 300, 400],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

from sklearn.model_selection import RandomizedSearchCV

random_search = RandomizedSearchCV(
    estimator=model,
    param_distributions=param_grid,
    n_iter=20,  # liczba kombinacji do przetestowania
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1,
    scoring='neg_mean_squared_error'
)
```

Dopasowanie do danych treningowych:

```
random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
print(f"Najlepsze parametry: {random_search.best_params_}")
```

```
Najlepsze parametry:
{'n_estimators': 400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_depth': 30, 'bootstrap': True}
```

Predykcja:

```
y_pred_best_log = best_model.predict(X_test)
y_pred_best = np.expm1(y_pred_best_log)  # odwrotna transformacja log
y_test_orig = np.expm1(y_test)
```

Ewaluacja:

```
rmse_best = mean_squared_error(y_test_orig, y_pred_best, squared=False)
mae_best = mean_absolute_error(y_test_orig, y_pred_best)
r2_best = r2_score(y_test_orig, y_pred_best)

print(f"\nStrojenie hiperparametrów - wyniki:")
print(f"RMSE: {rmse_best:.2f}")
```

```
print(f"MAE: {mae_best:.2f}")
print(f"R²: {r2_best:.4f}")
```

```
Strojenie hiperparametrów - wyniki:
RMSE: 441.70
MAE: 79.89
R²: 0.9996
```

Podejście to pozwoliło znaleźć jeszcze lepiej dopasowany model, co potwierdzają nowe metryki.

Tabela porównawcza dla modelu podstawowego i dostrojonego:

```
metrics_df = pd.DataFrame({
    'Metryka': ['MAE', 'RMSE', 'R²'],
    'Model bazowy': [round(rmse), round(mae), round(r2,4)],
    'Model strojony': [round(rmse_best), round(mae_best), round(r2_best,4)]
})
print(metrics_df)
```

	Metryka	Model bazowy	Model strojony
0	MAE	586.0000	442.0000
1	RMSE	94.0000	80.0000
2	R²	0.9993	0.9996

Wnioski:

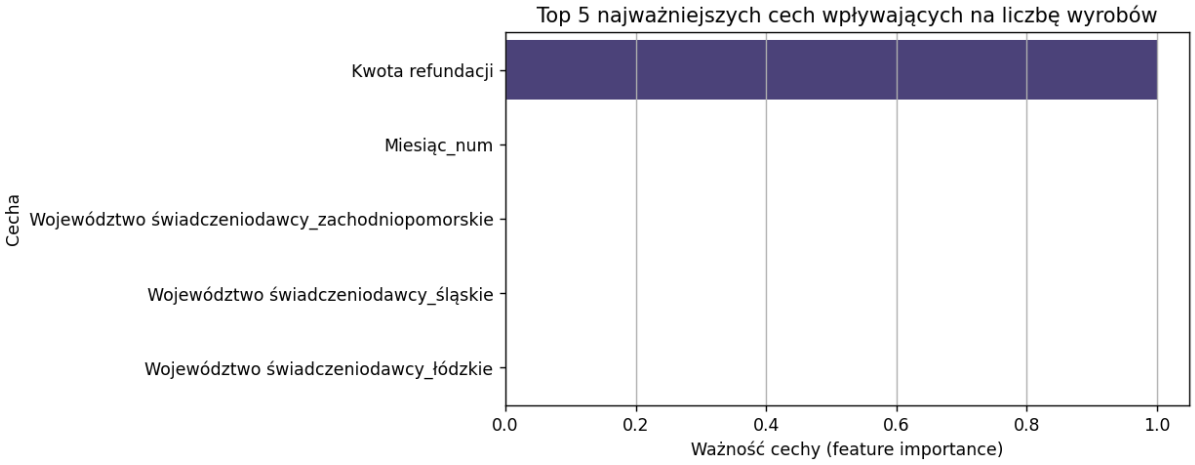
- MAE (586 → 442) - średni błąd prognozy zmniejszył się o 144 jednostki, co oznacza wyraźnie większą precyzję modelu strojonego.
- RMSE (94 → 80) - mniejsze odchylenia prognoz – model strojony lepiej radzi sobie z nietypowymi przypadkami i wartościami odstającymi.
- R^2 (0.9993 → 0.9996) - oba modele dobrze odwzorowują dane, ale strojony model wyjaśnia jeszcze więcej zmienności (99,96%).
- Strojenie hiperparametrów zoptymalizowało dokładność predykcji.

Zmienne, które znacząco wpływają na przewidywania:

```
feature_importances = pd.Series(best_model.feature_importances_,
index=X_train.columns)
feature_importances =
feature_importances.sort_values(ascending=False).head(5)

plt.figure(figsize=(10, 4))
sns.barplot(x=feature_importances.values, y=feature_importances.index,
```

```
palette='viridis')
plt.title("Top 5 najważniejszych cech wpływających na liczbę wyrobów")
plt.xlabel("Ważność cechy (feature importance)")
plt.ylabel("Cecha")
plt.grid(True, axis='x')
plt.tight_layout()
plt.show()
```



Po analizie ważności cech wpływających na liczbę refundowanych wyrobów, najistotniejsza okazała się ‘Kwota refundacji’, która jest bezpośrednio powiązana ze skalą sprzedaży. ‘Kwota refundacji’ zawiera część informacji o zmiennej celu i jest z nią silnie skorelowana.

Konsekwencje uwzględnienia tej zmiennej:

- Model osiąga nadzwyczaj wysokie metryki ($R^2 > 0.99$) – ale głównie dlatego, że ‘Kwota refundacji’ implikuje ‘Liczbę wyrobów’.
- W praktyce prognozujemy liczbę sprzedanych glukometrów, przy zasileniu modelu ‘Kwotą refundacji’ podajemy jednocześnie przybliżoną wartość celu, co prowadzi do data leakage (przecieku informacji) – model ma dostęp do informacji, której nie będzie miał w rzeczywistym zastosowaniu (podczas prognozowania sprzedaży).

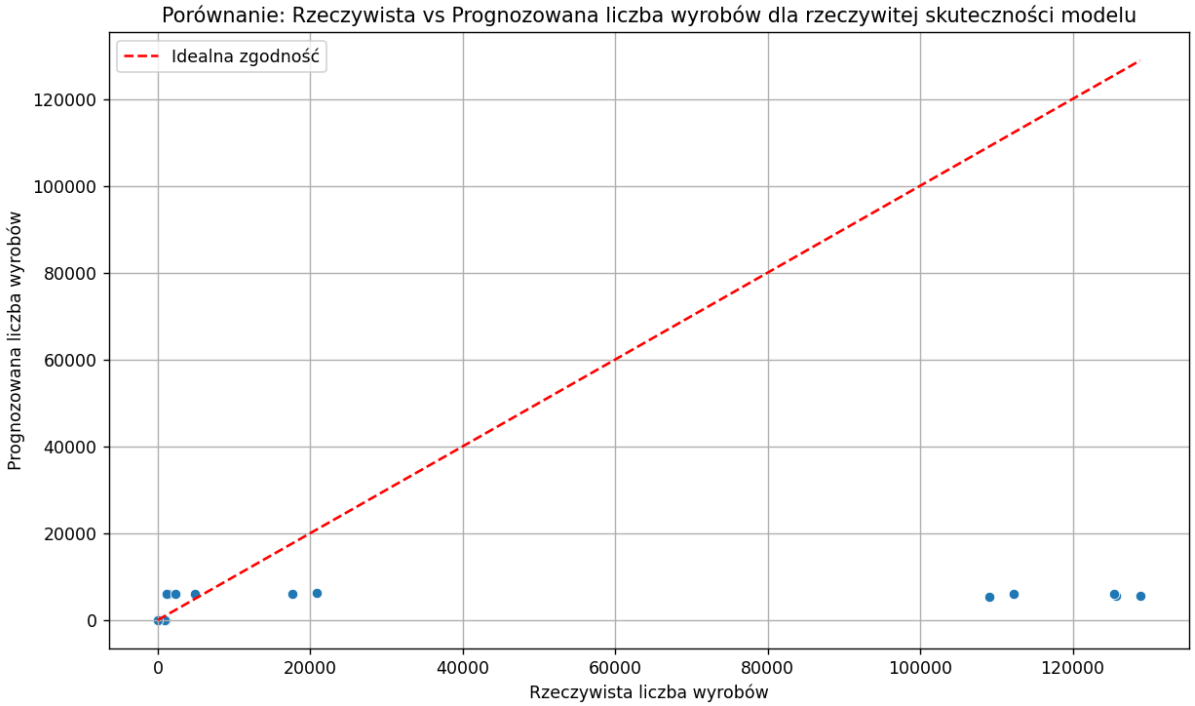
Wniosek: Model jest przesadnie dopasowany, metryki są zawyżone i nierealistyczne. Dlatego należy wykluczyć zmienną ‘Kwota refundacji’ z cech wejściowych, ponieważ prowadzi to do przecieku danych. Model korzysta z informacji, które w rzeczywistym scenariuszu predykcyjnym nie byłyby dostępne.

11. Podsumowanie

W początkowej wersji modelu wykorzystano zmienną ‘Kwota refundacji’, która jednak okazała się bezpośrednio powiązana ze zmienną celu (Liczba wyrobów). Jej obecność spowodowała data leakage, dlatego w finalnej wersji modelu zmienna ta została usunięta.

Metryki modelu z wykluczoną zmienną ‘Kwota refundacji’ pogorszyły się znacząco, R^2 spadł niemal do zera, ale daje obraz rzeczywistej skuteczności modelu.

```
RMSE dla rzeczywistej skutecznosci modelu: 22088.35
MAE dla rzeczywistej skutecznosci modelu: 4595.04
R² dla rzeczywistej skutecznosci modelu: 0.0550
```



Model nie potrafi skutecznie wyjaśnić zmienności i wykonać sensownej prognozy. Możliwe, że dane są zbyt zagregowane i zawierają zbyt małą ilość cech pośrednich (np. dodatkowe dane o pacjentach, szczegółowa data wystawienia/refundacji.)

12. Załączniki

- model_1.py z kodem źródłowym
- model_2.py z kodem źródłowym
- zbiór danych: czujniki_glukozy.xlsx