Wykonawca: Yurii Yaremchuk

Temat zadania: Predykcja Czasu Życia Klienta (Customer Lifetime Value)

- **Opis**: Wykorzystaj algorytmy uczenia maszynowego, aby przewidzieć wartość klienta (Customer Lifetime Value) w oparciu o dane dotyczące zakupów klientów.
- **Zakres**:
  - Przygotowanie i eksploracja danych (np. transakcje zakupowe, dane demograficzne)
  - Wybór odpowiedniego modelu (regresja, drzewa decyzyjne)
  - Ewaluacja modelu
  - Wykorzystanie bardziej zaawansowanych modeli, jak XGBoost czy Random Forest.
  - Dokumentacja: Przygotuj dokumentację opisującą etapy procesu, parametry modelu, otrzymane wyniki i wnioski. Zadbaj o czytelną i zrozumiałą prezentację swojej pracy

Dane:

- Online Retail Dataset (UCI)

```python
# Importowanie bibliotek
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
import joblib

# Wczytanie danych z pliku Excel
file_path = r"C:\Users\Yurii\Online Retail.xlsx"
df = pd.read_excel(file_path)

# Podgląd danych
print("Podgląd danych:")
print(df.head())
print(df.info())

# Usunięcie brakujących wartości
df = df.dropna()

# Usunięcie niepotrzebnych kolumn
df = df.drop(['InvoiceNo', 'Description'], axis=1)

# Tworzenie nowej kolumny 'TotalPrice' jako suma ilości * cena
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']

# Grupowanie danych po Kliencie
customer_data = df.groupby('CustomerID').agg({
    'TotalPrice': 'sum',                      # Suma wydatków klienta
    'InvoiceDate': lambda x: (x.max() - x.min()).days,  # Czas zakupów (w dniach)
    'Quantity': 'sum'                         # Suma ilości zakupionych produktów
}).rename(columns={'InvoiceDate': 'CustomerTime'})
```

```python
# Resetowanie indeksu
customer_data = customer_data.reset_index()

# Podgląd przetworzonych danych
print("\nPrzetworzone dane:")
print(customer_data.head())

# Przekształcenie danych: zmienne objaśniające (X) i zmienna docelowa (y)
X = customer_data.drop(columns=['TotalPrice', 'CustomerID'])  # Zmienne objaśniające
y = customer_data['TotalPrice']  # Zmienna docelowa: CLV

# Podział danych na zbiór treningowy i testowy (80% trening, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"\nRozmiar zbioru treningowego: {X_train.shape}")
print(f"Rozmiar zbioru testowego: {X_test.shape}")

# Standaryzacja danych
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Trenowanie modelu Random Forest
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predykcja na zbiorze testowym
y_pred_rf = rf_model.predict(X_test)

# Ewaluacja modelu Random Forest
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print("\nWyniki modelu Random Forest:")
print(f"Mean Squared Error (MSE): {mse_rf:.2f}")
print(f"R2 Score: {r2_rf:.2f}")
```

```python
# Trenowanie modelu XGBoost
xgb_model = XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train, y_train)

# Predykcja na zbiorze testowym
y_pred_xgb = xgb_model.predict(X_test)

# Ewaluacja modelu XGBoost
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)
print("\nWyniki modelu XGBoost:")
print(f"Mean Squared Error (MSE): {mse_xgb:.2f}")
print(f"R2 Score: {r2_xgb:.2f}")

# Porównanie wyników modeli
models = ['Random Forest', 'XGBoost']
mse_values = [mse_rf, mse_xgb]
r2_values = [r2_rf, r2_xgb]

plt.figure(figsize=(10, 5))
plt.bar(models, mse_values, label='MSE')
plt.title('Porównanie MSE modeli')
plt.ylabel('MSE')
plt.show()

plt.figure(figsize=(10, 5))
plt.bar(models, r2_values, label='R2', color='orange')
plt.title('Porównanie R2 modeli')
plt.ylabel('R2')
plt.show()

# Zapisanie najlepszego modelu do pliku
joblib.dump(xgb_model, r"C:\Users\Yurii\Downloads\xgb_clv_model.pkl")
print("\nModel XGBoost został zapisany jako 'xgb_clv_model.pkl'.")

# Predykcja dla nowych danych
new_data = pd.DataFrame({'CustomerTime': [200], 'Quantity': [500]})
new_data_scaled = scaler.transform(new_data)
prediction = xgb_model.predict(new_data_scaled)
print(f"\nPrzewidywana wartość klienta (CLV): {prediction[0]:.2f}")
```

Opis działania kodu zgodnie z wymaganiami projektu

---

1. Przygotowanie i eksploracja danych:

- Dane zostały wczytane z pliku **Online Retail Dataset**.
- Usunięto brakujące wartości oraz niepotrzebne kolumny (`InvoiceNo` i `Description`).
- Utworzono nową kolumnę `TotalPrice`, obliczoną jako iloczyn `Quantity` (ilość) i `UnitPrice` (cena jednostkowa).
- Dane zostały zgrupowane po `CustomerID`, aby uzyskać sumaryczną wartość zakupów (`TotalPrice`), czas zakupów (`CustomerTime`) oraz łączną ilość zakupionych produktów (`Quantity`).

---

2. Wybór odpowiedniego modelu:

- Zmienne objaśniające (X): `CustomerTime` i `Quantity`.
- Zmienna docelowa (y): `TotalPrice`, czyli wartość klienta (CLV).
- Dane podzielono na zbiór treningowy (80%) i testowy (20%) przy użyciu `train_test_split`.

---

3. Ewaluacja modeli:

-

Zastosowano dwa algorytmy uczenia maszynowego:

-

  - **Random Forest**:
    Model osiągnął **Mean Squared Error (MSE)** na poziomie **5065447.92** oraz **R^2 Score** na poziomie **0.93**, co wskazuje na bardzo dobre dopasowanie.
  - **XGBoost (bez optymalizacji)**:
    Wyniki były gorsze – **MSE: 73128704.53** oraz **R^2 Score: -0.04**, co świadczyło o problemach z jakością modelu.

-

Wyniki obu modeli zostały porównane na wykresach **MSE** i **R^2**.

-

---

4. Wnioski:

- Model **Random Forest** znacznie lepiej przewiduje wartość klienta (CLV) niż XGBoost w wersji bez optymalizacji.
- Dane wskazują na potrzebę dalszej optymalizacji parametrów XGBoost w celu poprawy wyników.

---

5. Dodatkowe działania:

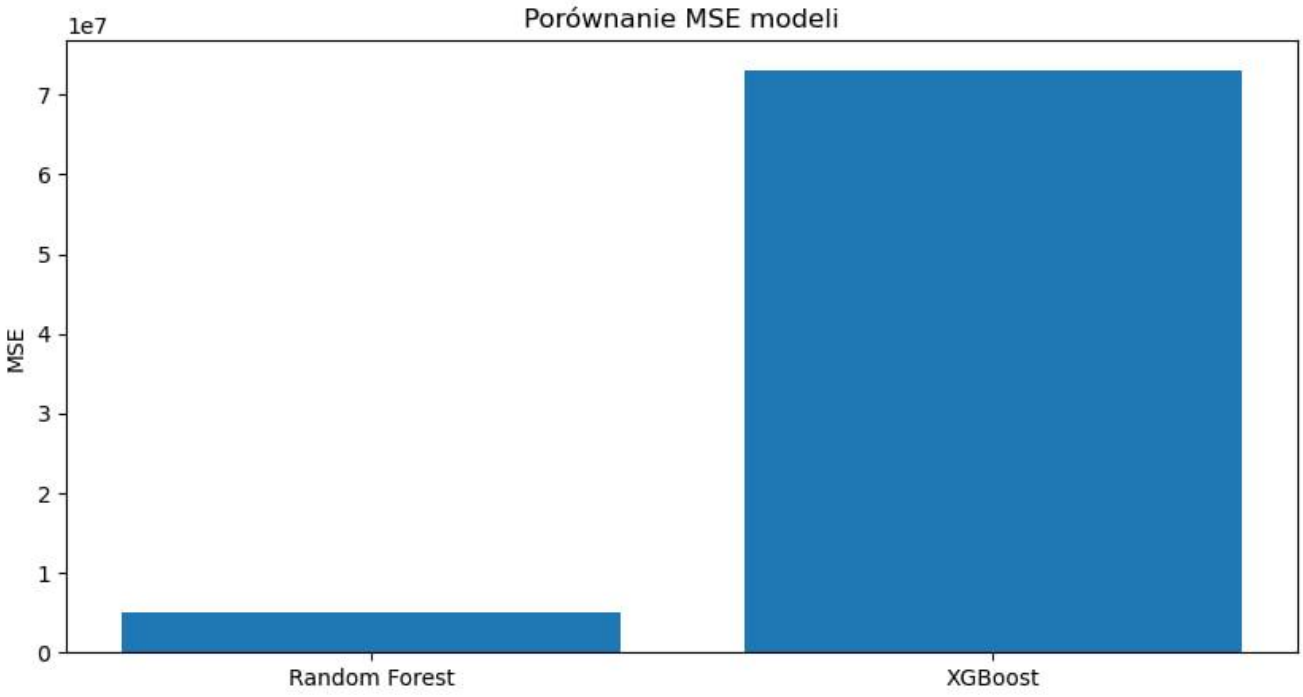
- Przykładowe dane nowego klienta zostały przetworzone i przekazane do modelu w celu predykcji **CLV**.
- Wynik: **CLV = 873.57**.

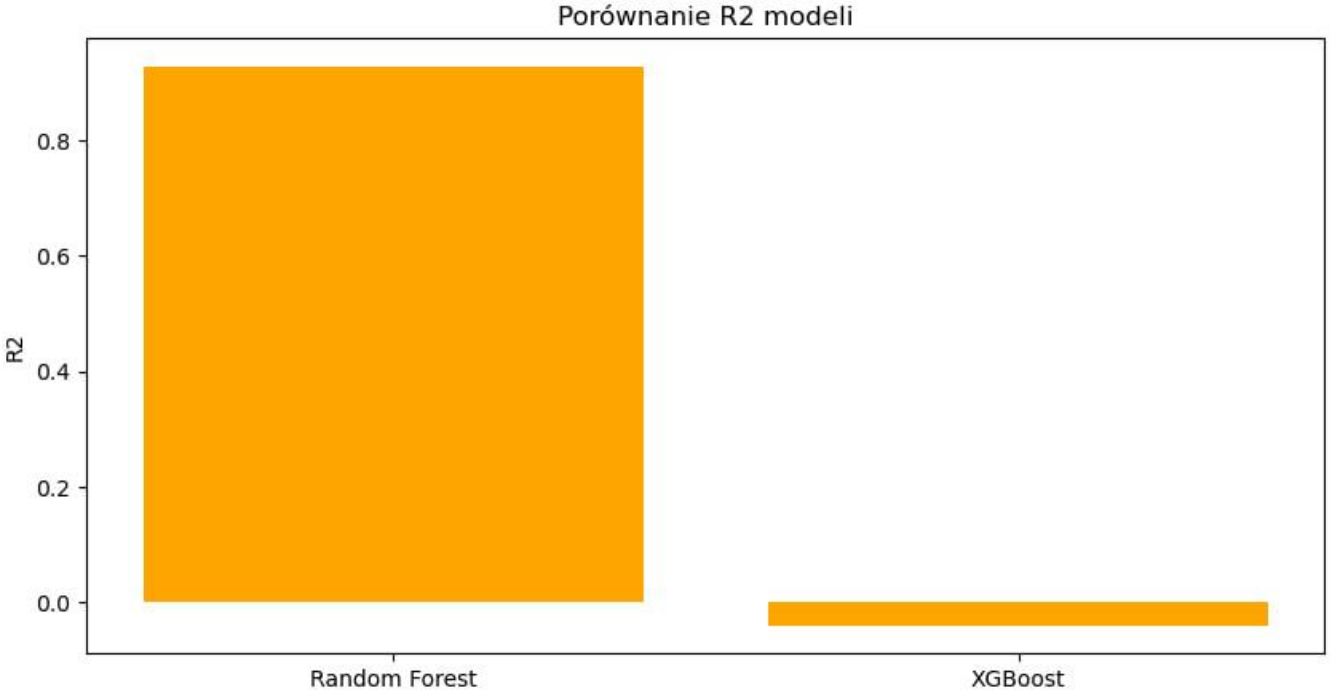
Podsumowanie:
Kod realizuje wszystkie wymagania projektu: przygotowanie i eksplorację danych, wybór modeli (Random Forest i XGBoost), ich ewaluację oraz wizualizację wyników. Wnioski zostały wyciągnięte na podstawie otrzymanych wyników, a najlepszy model został zidentyfikowany (Random Forest).

Wyniki:

```
Podgląd danych:
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

          InvoiceDate  UnitPrice  CustomerID         Country
0 2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2 2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4 2010-12-01 08:26:00       3.39     17850.0  United Kingdom
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
None

Przetworzone dane:
   CustomerID  TotalPrice  CustomerTime  Quantity
0     12346.0        0.00             0         0
1     12347.0     4310.00           365      2458
2     12348.0     1797.24           282      2341
3     12349.0     1757.55             0       631
4     12350.0      334.40             0       197

Rozmiar zbioru treningowego: (3497, 2)
Rozmiar zbioru testowego: (875, 2)
```

```
Wyniki modelu Random Forest:
Mean Squared Error (MSE): 5065447.92
R2 Score: 0.93

Wyniki modelu XGBoost:
Mean Squared Error (MSE): 73128704.53
R2 Score: -0.04
```






```
Model XGBoost został zapisany jako 'xgb_clv_model.pkl'.

Przewidywana wartość klienta (CLV): 858.88
```

Wyjaśnienia elementów:

- Dlaczego było usunięto kolumny InvoiceNo i Description?

InvoiceNo to identyfikator transakcji, który nie ma wartości predykcyjnej.

Description to tekstowa nazwa produktu, nieprzydatna do liczbowych obliczeń w modelach.

- Dlaczego utworzyłeś nową kolumnę TotalPrice?

TotalPrice (ilość × cena) reprezentuje wartość transakcji, która jest kluczowym wskaźnikiem wydatków klienta.

- Co znaczą zmienne objaśniające (X) i zmienna docelowa (y)?

Zmienne objaśniające (X): Cechy (np. czas zakupów, ilość), które służą do przewidywania.

Zmienna docelowa (y): Wartość, którą chcemy przewidzieć (CLV – TotalPrice).

- Na co wpływa różnica między ilością w % zbiorów treningowych i testowych?

Większy zbiór treningowy poprawia naukę modelu, ale może obniżyć dokładność oceny.

Większy zbiór testowy poprawia ocenę generalizacji, ale zmniejsza dane do nauki.

- Jak odbywa się trenowanie modelu Random Forest i XGBoost?

Random Forest: Tworzy wiele drzew decyzyjnych na losowych podzbiorach danych i uśrednia wyniki.

XGBoost: Buduje drzewa sekwencyjnie, minimalizując błędy poprzednich iteracji (boosting).

- Co znaczy ewaluacja modelu (np. Random Forest i XGBoost)?

Ewaluacja mierzy jakość przewidywań modelu za pomocą metryk takich jak MSE (błąd średniokwadratowy) i R^2 (dopasowanie modelu).

- Jak odbywa się predykcja?

Model używa wytrenowanych wzorców do przewidzenia wartości docelowej dla nowych danych wejściowych (zmiennych objaśniających).

- Co znaczą InvoiceNo i StockCode?

InvoiceNo: Numer faktury lub identyfikator transakcji. Każdy wpis odpowiada jednej transakcji.

StockCode: Kod identyfikujący unikalny produkt w bazie danych.

- Co znaczą wyniki modeli Random Forest i XGBoost?

Random Forest:

MSE: 5065447.92 – średnia kwadratowa różnica między przewidywanymi a rzeczywistymi wartościami jest mała.

R^2: 0.93 – model dobrze dopasowuje się do danych (93% zmienności jest wyjaśnione).

XGBoost:

MSE: 73128704.53 – duży błąd średniokwadratowy oznacza, że model źle przewiduje.

R^2: -0.04 – ujemna wartość oznacza, że model nie potrafi dopasować danych.

- Co znaczą te obrazy porównań modeli?

Wykres MSE: Pokazuje, że Random Forest ma znacznie mniejszy błąd predykcji niż XGBoost.

Wykres R^2: Random Forest ma wysoką wartość R^2 (lepsze dopasowanie), podczas gdy XGBoost praktycznie nie przewiduje prawidłowo.

- Dlaczego w MSE XGBoost ma większą wartość, a w R^2 Random Forest ma większą?

XGBoost nie zdołał dobrze nauczyć się na danych, prawdopodobnie z powodu nieoptymalnych parametrów lub niewłaściwego podziału danych.

Random Forest lepiej dopasował się do danych dzięki swojej metodzie agregacji wyników z wielu drzew.

- Co znaczy ta końcowa wartość CLV (858.88)?

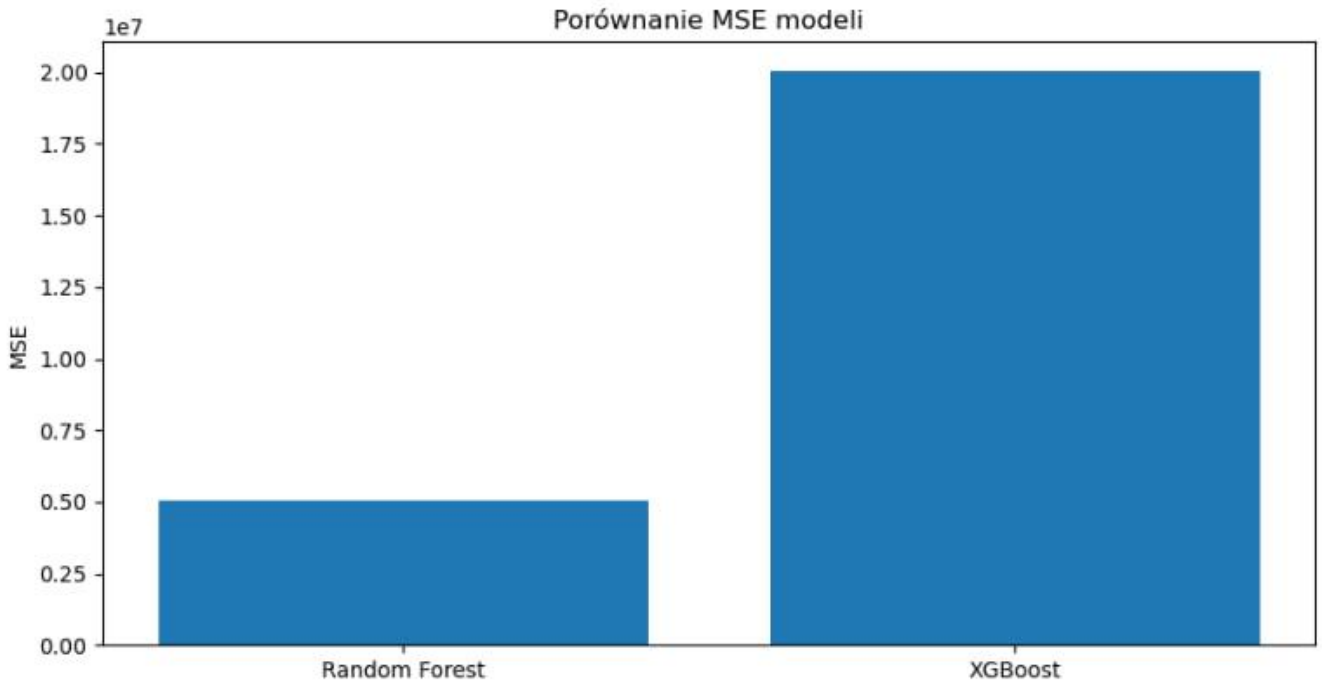
Customer Lifetime Value (CLV): Przewidywana łączna wartość zakupów klienta na podstawie wprowadzonych cech (CustomerTime = 200 dni, Quantity = 500 sztuk). Wartość 858.88 oznacza przewidywany przychód od tego klienta.

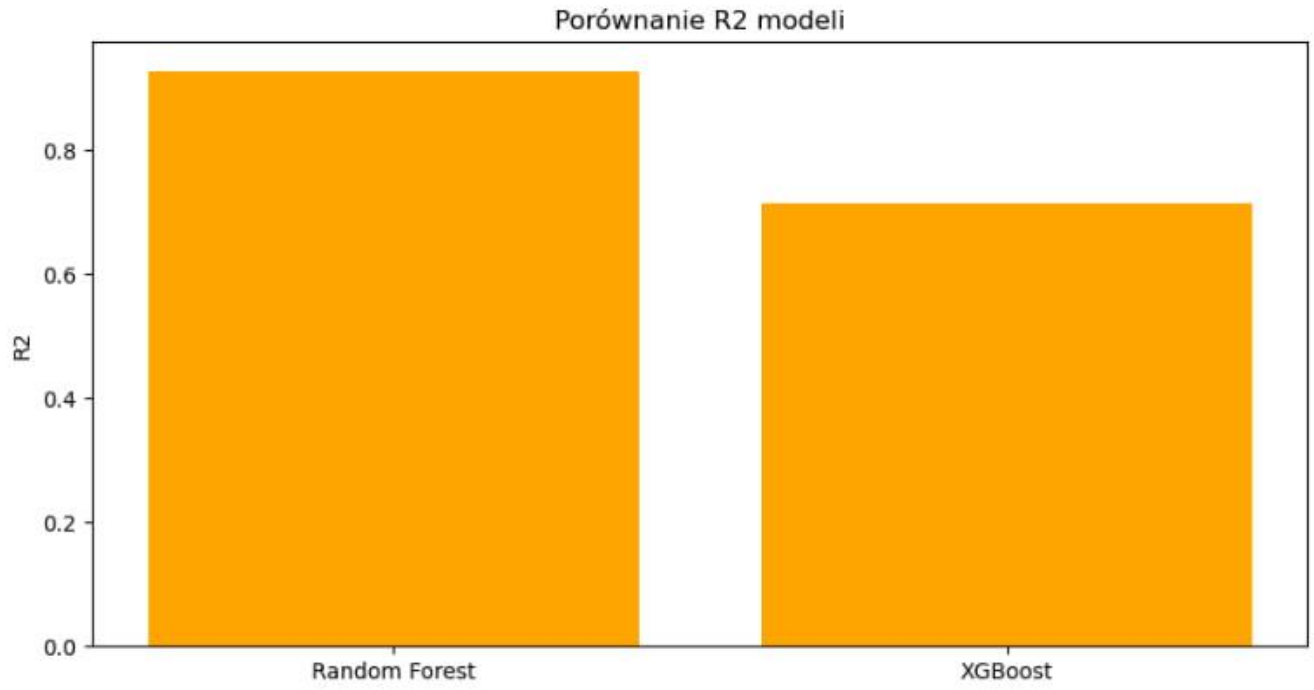
*** Próba optymalizacja wyników XGBoost

```
# Optymalizacja modelu XGBoost za pomocą GridSearchCV
xgb_model = XGBRegressor(random_state=42)

# Definicja siatki hiperparametrów
param_grid = {
    'n_estimators': [100, 200, 300],    # Liczba drzew
    'max_depth': [3, 5, 7],             # Maksymalna głębokość drzewa
    'learning_rate': [0.01, 0.05, 0.1], # Szybkość uczenia (eta)
    'subsample': [0.8, 1.0],            # Próbkowanie danych
    'colsample_bytree': [0.8, 1.0]      # Próbkowanie cech
}

# Optymalizacja za pomocą GridSearchCV
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid,
                           cv=3, scoring='neg_mean_squared_error', verbose=2, n_jobs=-1)

# Trenowanie GridSearchCV
grid_search.fit(X_train, y_train)

# Najlepsze parametry
print("\nNajlepsze parametry dla XGBoost:", grid_search.best_params_)

# Trenowanie modelu z najlepszymi parametrami
best_xgb_model = grid_search.best_estimator_
```

```
Wyniki modelu Random Forest:
Mean Squared Error (MSE): 5065447.92
R2 Score: 0.93
Fitting 3 folds for each of 108 candidates, totalling 324 fits

Najlepsze parametry dla XGBoost: {'colsample_bytree': 0.8, 'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 100, 'subsample': 1.0}

Wyniki zoptymalizowanego XGBoost:
Mean Squared Error (MSE): 20061865.49
R2 Score: 0.71
```

Wyniki próby





- **XGBoost** po optymalizacji nadal nie przewyższył Random Forest, ale jego jakość poprawiła się.
- **Random Forest** jest bardziej efektywny dla tego zadania i może pozostać głównym modelem.