Lab3-Analizator wyników

Kamil Pietruchowski – s21147

1. Introdukcja

Na podstawie datasetu CollegeDistance.csv zbudowany został model predykcyjny, który przewiduje zmienną score.

2. Eksploracja i wstępna analiza danych

Podgląd danych:

	rownames	gender	ethnicity	score	fcollege	mcollege	home	urban	unemp	wage	distance	tuition	education	income	region
0	1	male	other	39.150002	yes	no	yes	yes	6.2	8.09	0.2	0.88915	12	high	other
1	2	female	other	48.869999	no	no	yes	yes	6.2	8.09	0.2	0.88915	12	low	other
2	3	male	other	48.740002	no	no	yes	yes	6.2	8.09	0.2	0.88915	12	low	other
3	4	male	afam	40.400002	no	no	yes	yes	6.2	8.09	0.2	0.88915	12	low	other
4	5	female	other	40.480000	no	no	no	yes	5.6	8.09	0.4	0.88915	13	low	other

Dane składają się z 4739 wpisów oraz 15 kolumn. 5 kolumn posiada typ danych Float64 (score, unemp, wage, distance, tuition), 2 kolumny posiadają typ danych Int64 (rownames, education) oraz 8 kolumn posiada typ danych Object (gender, ethnicity, fcollege, mcollege, home, urban, income, region).

```
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   rownames   4739 non-null   int64
 1   gender     4739 non-null   object
 2   ethnicity  4739 non-null   object
 3   score      4739 non-null   float64
 4   fcollege   4739 non-null   object
 5   mcollege   4739 non-null   object
 6   home       4739 non-null   object
 7   urban      4739 non-null   object
 8   unemp      4739 non-null   float64
 9   wage       4739 non-null   float64
 10  distance   4739 non-null   float64
 11  tuition    4739 non-null   float64
 12  education  4739 non-null   int64
 13  income     4739 non-null   object
 14  region     4739 non-null   object
dtypes: float64(5), int64(2), object(8)
```

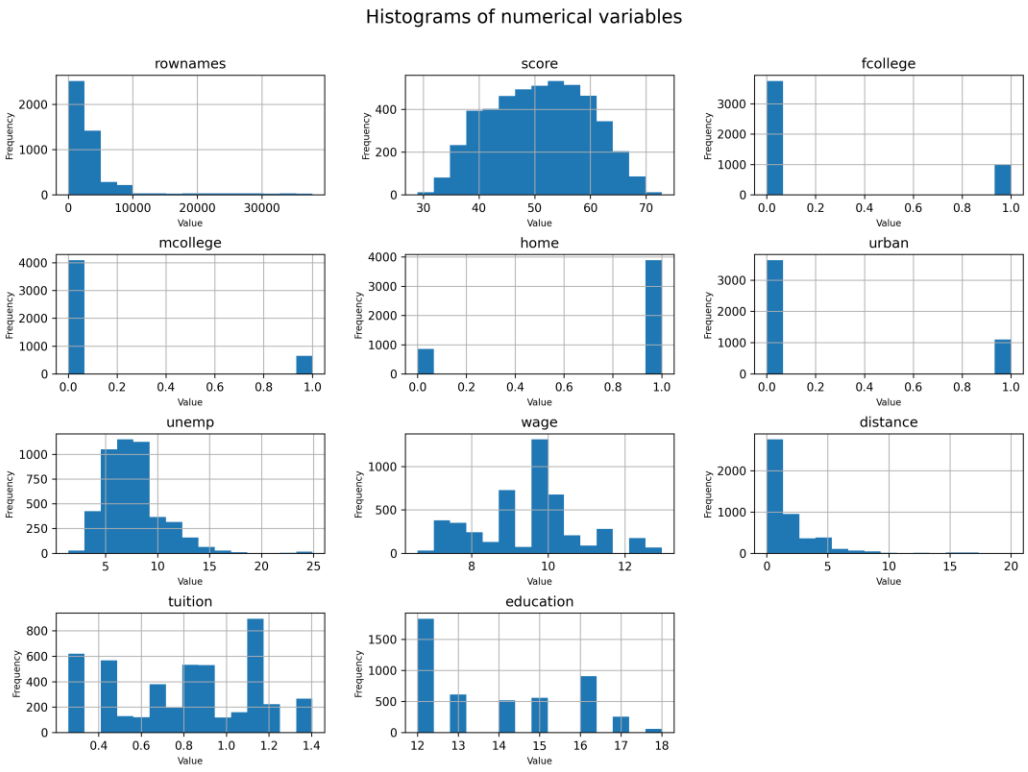
Dane nie posiadają żadnych brakujących wartości

```
Brakujące wartości w każdej kolumnie:
rownames     0
gender       0
ethnicity    0
score        0
fcollege     0
mcollege     0
home         0
urban        0
unemp        0
wage         0
distance     0
tuition      0
education    0
income       0
region       0
```

Statystyki opisowe danych numerycznych:

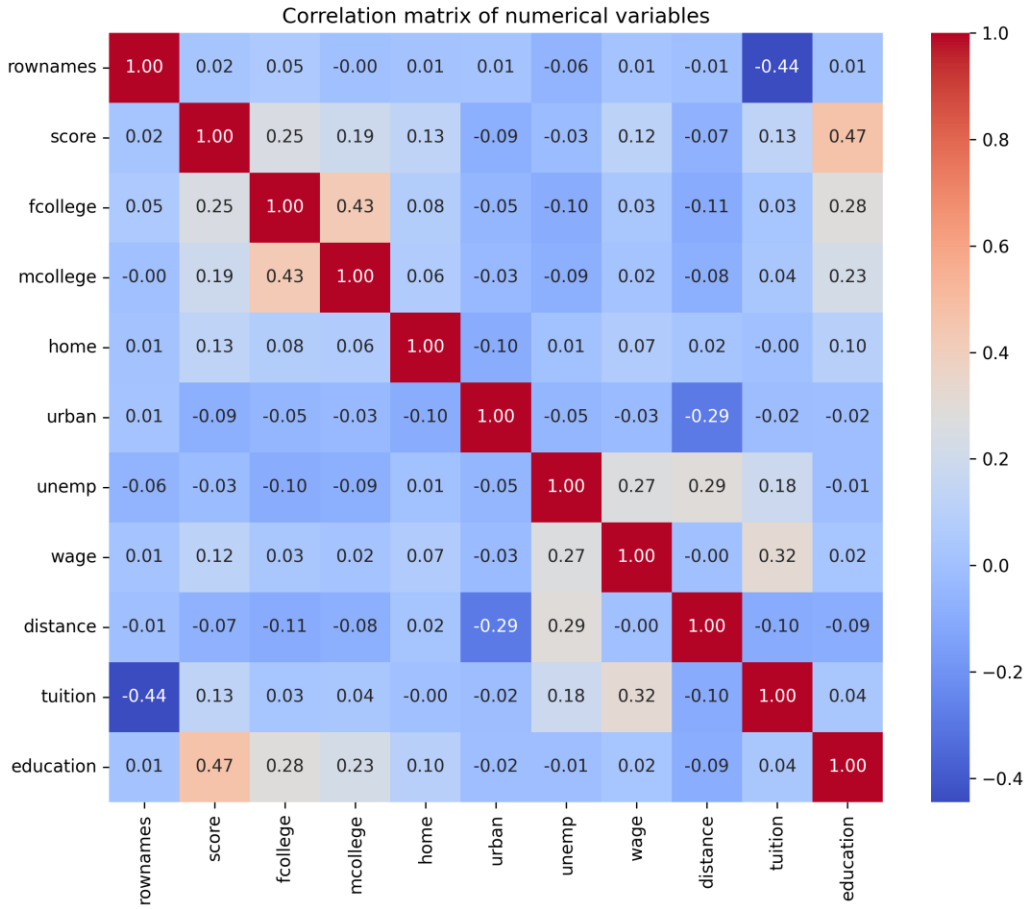
```
Statystyki opisowe:
       rownames    score    unemp     wage  distance  tuition  education
count   4739.00  4739.00  4739.00  4739.00    4739.0  4739.00    4739.00
mean    3954.64    50.89     7.60     9.50       1.8     0.81      13.81
std     5953.83     8.70     2.76     1.34       2.3     0.34       1.79
min        1.00    28.95     1.40     6.59       0.0     0.26      12.00
25%     1185.50    43.92     5.90     8.85       0.4     0.48      12.00
50%     2370.00    51.19     7.10     9.68       1.0     0.82      13.00
75%     3554.50    57.77     8.90    10.15       2.5     1.13      16.00
max    37810.00    72.81    24.90    12.96      20.0     1.40      18.00
```

Histogramy danych numerycznych(w tym dane yes/no zamienione na 1/0):



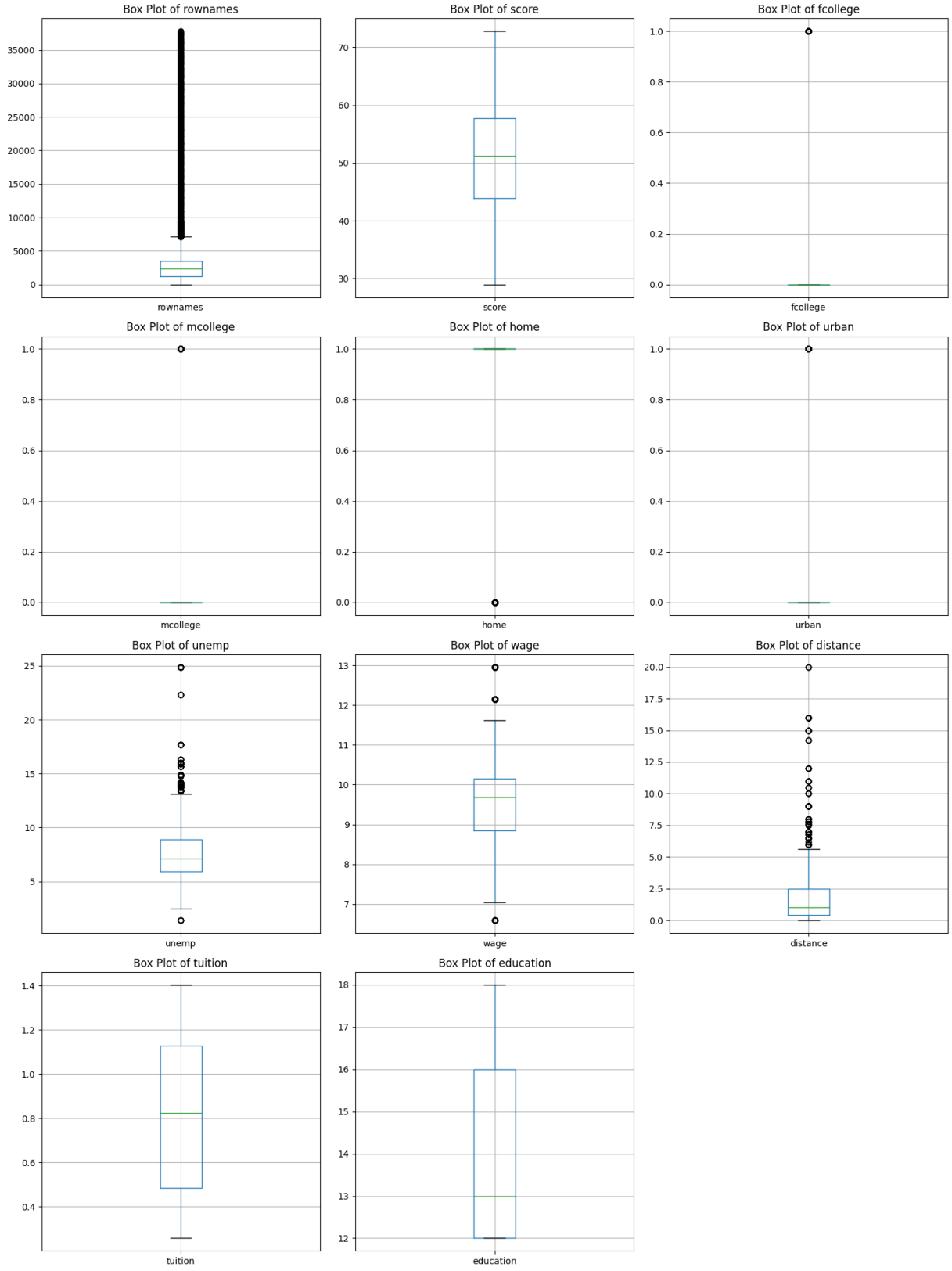
Na podstawie rysunku można stwierdzić, że w danych bool (fcollege, mcollege, home, urban) dominuje jedna z wartości. False w przypadku fcollege, mcollege i urban oraz True w przypadku home.

Macierz korelacji danych numerycznych(w tym bool):



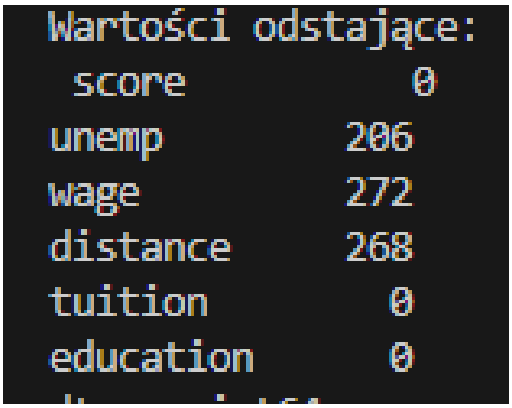
Kolumna ‘score’ ma największą korelację z kolumną ‘education’ (0.47) a następnie z kolumną ‘fcollege’ (0.25). Kolumny, które mają brak istotnego powiązania z kolumną ‘score’ to: ‘rownames’(0.02), ‘urban’(-0.09), ‘unemp’(-0.03),’distance’(-0.07).

Wykres skrzynkowy danych numerycznych:

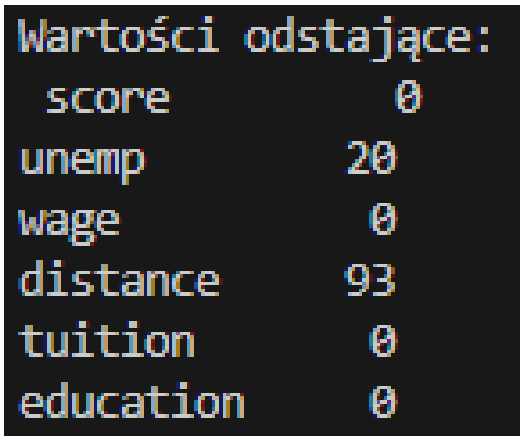


Na podstawie rysunku wykresów skrzynkowych każdej kolumny numerycznej można stwierdzić, że kolumny ‘unemp’, ‘wage’ oraz ‘distance’ zawierają wiele wartości odstających.

Przy użyciu formuły IQR = Q3 – Q1 i obliczeniu górnej oraz dolnej granicy Q1-1.5*IQR, Q3+1.5*IQR znaleziono wartości odstające w kolumnach ‘unemp’(206), ‘wage’(272) oraz ‘distance’(268).



Wartości odstające IQR * 3:



3. Inżynieria cech i przygotowanie danych.

Na początek usunięto kolumnę ‘rownames’ z danych, na których będzie trenowany model, ponieważ kolumna nie ma wpływu na ‘score’.

```
df.drop(columns=['rownames'],inplace=True)
```

Następnie kolumny typu ‘bool’, które zawierają wartości ‘yes’ lub ‘no’ zamieniono na kolumny numeryczne. Wartości ‘yes’ zostały zamienione na 1, wartości ‘no’ zostały zamienione na 0.

```
# Converting yes/no columns to binary 1/0
for column in boolean_columns:
    df[column] = df[column].apply(lambda x: 1 if x == 'yes' else 0)
```

Resztę kolumn kategorycznych zakodowano na wartości liczbowe, aby algorytmy uczenia maszynowego mogły je interpretować i uwzględniać w obliczeniach.

```
# Encoding categorical columns
for column in categorical_columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le  # Store label encoders for future use if needed
```

Następnie stworzono nową strukturę danych bez wartości odstających 3 IQR.

```
#generate dataframe without outliers 3IQR
Q1 = df[numerical_columns].quantile(0.25)
Q3 = df[numerical_columns].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 3 * IQR
upper_bound = Q3 + 3 * IQR
df_no_outliers = df[~((df[numerical_columns] < lower_bound) | (df[numerical_columns] > upper_bound)).any(axis=1)]
```

Kolumny numeryczne zeskalowano w celu ujednolicenia zakresu wartości i zapobieganiu dominacji cech o większej skali(w obu dataframes).

```
# Scale numerical columns
scaler = StandardScaler()
df[numerical_columns] = scaler.fit_transform(df[numerical_columns])
df_no_outliers[numerical_columns] = scaler.fit_transform(df_no_outliers[numerical_columns])
```

Na koniec podzielono zbiory na treningowe i testowe.

```python
# Split data into train and test sets
def split_df(df):
    y = df['score']
    X = df.drop(columns='score')
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    return X_train, y_train, X_test, y_test
```

4. Wybór i trenowanie modelu

W celu wybrania odpowiedniego algorytmu, przeprowadzono uczenie na wielu różnych modelach.

```python
# List of models for evaluation
models = {
    "Linear Regression" : lin_reg,
    "Gradient Boosting Regressor": gb_reg,
    "Support Vector Regressor": svr
}
def evaluate_models(models, X_train, y_train, X_test, y_test):
    results={}
    for model_name, model in models.items():
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)
        r2 = r2_score(y_test, y_pred)
        mse = root_mean_squared_error(y_test, y_pred)
        mae = mean_absolute_error(y_test, y_pred)
        # Store results
        results[model_name] = {"R^2": r2, "MSE": mse, "MAE": mae}
        # Print model performance
        print(f"{model_name} Performance:")
        print(f"R^2 Score: {r2:.2f}")
        print(f"Root Mean Squared Error (MSE): {mse:.2f}")
        print(f"Mean Absolute Error (MAE): {mae:.2f}\n")
    print('\n')
    return results
```

Następnie porównano wyniki.

```
Linear Regression Performance:
R^2 Score: 0.35
Root Mean Squared Error (MSE): 0.81
Mean Absolute Error (MAE): 0.66

Random Forest Regressor Performance:
R^2 Score: 0.29
Root Mean Squared Error (MSE): 0.84
Mean Absolute Error (MAE): 0.67

Decision Tree Regressor Performance:
R^2 Score: -0.22
Root Mean Squared Error (MSE): 1.10
Mean Absolute Error (MAE): 0.88

Gradient Boosting Regressor Performance:
R^2 Score: 0.37
Root Mean Squared Error (MSE): 0.80
Mean Absolute Error (MAE): 0.65

Support Vector Regressor Performance:
R^2 Score: 0.34
Root Mean Squared Error (MSE): 0.82
Mean Absolute Error (MAE): 0.66
```

5. Ocena i optymalizacja modelu

Na podstawie powyższych wyników, wybrano 3 najlepsze algorytmy, a następnie przeprowadzono uczenie na zmodyfikowanych zbiorach danych.

Zbiór danych:

- Bez wartości odstających 3 IQR.

```
Linear Regression Performance:
R^2 Score: 0.32
Root Mean Squared Error (MSE): 0.84
Mean Absolute Error (MAE): 0.69

Gradient Boosting Regressor Performance:
R^2 Score: 0.32
Root Mean Squared Error (MSE): 0.84
Mean Absolute Error (MAE): 0.69

Support Vector Regressor Performance:
R^2 Score: 0.31
Root Mean Squared Error (MSE): 0.84
Mean Absolute Error (MAE): 0.68
```

- Bez kolumn ‘unemp’, ‘urban’, ‘distance’ ze znikomą korelacją z kolumną ‘score’.

```
Linear Regression Performance:
R^2 Score: 0.35
Root Mean Squared Error (MSE): 0.81
Mean Absolute Error (MAE): 0.66

Gradient Boosting Regressor Performance:
R^2 Score: 0.36
Root Mean Squared Error (MSE): 0.80
Mean Absolute Error (MAE): 0.66

Support Vector Regressor Performance:
R^2 Score: 0.33
Root Mean Squared Error (MSE): 0.82
Mean Absolute Error (MAE): 0.66
```

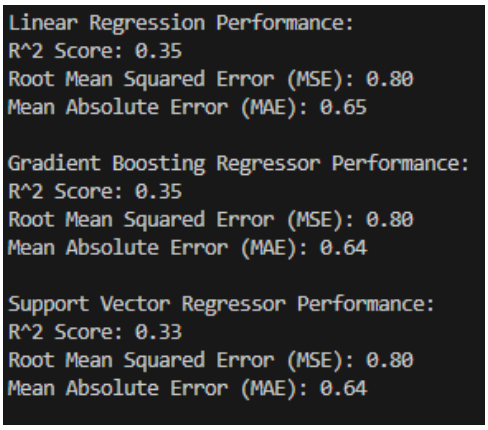
- Bez wartości odstających 3 IQR oraz kolumn ze znikomą korelacją.

```
Linear Regression Performance:
R^2 Score: 0.31
Root Mean Squared Error (MSE): 0.84
Mean Absolute Error (MAE): 0.69

Gradient Boosting Regressor Performance:
R^2 Score: 0.30
Root Mean Squared Error (MSE): 0.85
Mean Absolute Error (MAE): 0.69

Support Vector Regressor Performance:
R^2 Score: 0.30
Root Mean Squared Error (MSE): 0.85
Mean Absolute Error (MAE): 0.68
```

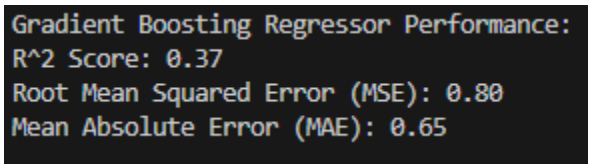
- Bez wartości odstających 1.5 IQR.

```
Linear Regression Performance:
R^2 Score: 0.35
Root Mean Squared Error (MSE): 0.80
Mean Absolute Error (MAE): 0.65

Gradient Boosting Regressor Performance:
R^2 Score: 0.33
Root Mean Squared Error (MSE): 0.80
Mean Absolute Error (MAE): 0.65

Support Vector Regressor Performance:
R^2 Score: 0.31
Root Mean Squared Error (MSE): 0.82
Mean Absolute Error (MAE): 0.65
```

- Bez wartości odstających 1.5 IQR oraz kolumn ze znikomą korelacją.



Na podstawie powyższych wyników można stwierdzić, że najlepszym modelem jest model uczony na podstawowym zbiorze danych przy pomocy algorytmu ‘Gradient Boosting Regressor’.



6. Wnioski

Aktualny model wykazuje ograniczoną zdolność do przewidywania, co potwierdzają niskie wartości R^2 oraz umiarkowane błędy RMSE i MAE.