cleaning-dataset

June 9, 2024

```
[ ]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     import folium
```

Celem jest predykcja ceny nieruchomości.

Zbiór danych zawiera 4802 instancje i kolumny określające:

BROKERTITLE: Title of the broker TYPE: Type of the house PRICE: Price of the house BEDS: Number of bedrooms BATH: Number of bathrooms PROPERTYSQFT: Square footage of the property ADDRESS: Full address of the house STATE: State of the house MAIN_ADDRESS: Main address information ADMINISTRATIVE_AREA_LEVEL_2: Administrative area level 2 information LOCALITY: Locality information SUBLOCALITY: Sublocality information STREET_NAME: Street name LONG_NAME: Long name FORMATTED_ADDRESS: Formatted address LATITUDE: Latitude coordinate of the house LONGITUDE: Longitude coordinate of the house

```
[ ]: import pandas as pd
     url = 'NY-House-Dataset.csv'
     data = pd.read_csv(url, sep= ';')
```

```
[ ]: data
```

```
[ ]:                                         BROKERTITLE                TYPE  \
0          Brokered by Douglas Elliman  -111 Fifth Ave      Condo for sale
1                                      Brokered by Serhant      Condo for sale
2                                   Brokered by Sowae Corp      House for sale
3                                      Brokered by COMPASS      Condo for sale
4     Brokered by Sotheby's International Realty - E…  Townhouse for sale
…                                                    …                 …
4796                                   Brokered by COMPASS      Co-op for sale
4797                       Brokered by Mjr Real Estate Llc      Co-op for sale
4798       Brokered by Douglas Elliman - 575 Madison Ave      Co-op for sale
4799             Brokered by E Realty International Corp      Condo for sale
4800                  Brokered by Nyc Realty Brokers Llc      Co-op for sale

          PRICE  BEDS       BATH  PROPERTYSQFT  \
0        315000     2   2.000000   1400.000000
```

1	195000000	7	10.000000	17545.000000
2	260000	4	2.000000	2015.000000
3	69000	3	1.000000	445.000000
4	55000000	7	2.373861	14175.000000
…	…	…	…	…
4796	599000	1	1.000000	2184.207862
4797	245000	1	1.000000	2184.207862
4798	1275000	1	1.000000	2184.207862
4799	598125	2	1.000000	655.000000
4800	349000	1	1.000000	750.000000

	ADDRESS \
0	2 E 55th St Unit 803
1	Central Park Tower Penthouse-217 W 57th New Yo…
2	620 Sinclair Ave
3	2 E 55th St Unit 908W33
4	5 E 64th St
…	…
4796	222 E 80th St Apt 3A
4797	97-40 62 Dr Unit Lg
4798	427 W 21st St Unit Garden
4799	91-23 Corona Ave Unit 4G
4800	460 Neptune Ave Apt 14O

	STATE \
0	New York, NY 10022
1	New York, NY 10019
2	Staten Island, NY 10312
3	Manhattan, NY 10022
4	New York, NY 10065
…	…
4796	Manhattan, NY 10075
4797	Rego Park, NY 11374
4798	New York, NY 10011
4799	Elmhurst, NY 11373
4800	Brooklyn, NY 11224

	MAIN_ADDRESS \
0	2 E 55th St Unit 803New York, NY 10022
1	Central Park Tower Penthouse-217 W 57th New Yo…
2	620 Sinclair AveStaten Island, NY 10312
3	2 E 55th St Unit 908W33Manhattan, NY 10022
4	5 E 64th StNew York, NY 10065
…	…
4796	222 E 80th St Apt 3AManhattan, NY 10075
4797	97-40 62 Dr Unit LgRego Park, NY 11374
4798	427 W 21st St Unit GardenNew York, NY 10011

```
4799         91-23 Corona Ave Unit 4GElmhurst, NY 11373
4800          460 Neptune Ave Apt 14OBrooklyn, NY 11224

     ADMINISTRATIVE_AREA_LEVEL_2         LOCALITY      SUBLOCALITY  \
0               New York County         New York        Manhattan
1                 United States         New York  New York County
2                 United States         New York  Richmond County
3                 United States         New York  New York County
4                 United States         New York  New York County
…                           …                …                …
4796                   New York  New York County         New York
4797              United States         New York    Queens County
4798              United States         New York  New York County
4799                   New York    Queens County           Queens
4800                   New York     Kings County         Brooklyn

           STREET_NAME         LONG_NAME  \
0     East 55th Street   Regis Residence
1             New York  West 57th Street
2        Staten Island   Sinclair Avenue
3             New York  East 55th Street
4             New York  East 64th Street
…                  …               …
4796         Manhattan               222
4797            Queens        62nd Drive
4798          New York  West 21st Street
4799          Flushing             91-23
4800      Coney Island               460

                                  FORMATTED_ADDRESS   LATITUDE  LONGITUDE
0     Regis Residence, 2 E 55th St #803, New York, N…  40.761255 -73.974483
1               217 W 57th St, New York, NY 10019, USA  40.766393 -73.980991
2        620 Sinclair Ave, Staten Island, NY 10312, USA  40.541805 -74.196109
3                 2 E 55th St, New York, NY 10022, USA  40.761398 -73.974613
4                 5 E 64th St, New York, NY 10065, USA  40.767224 -73.969856
…                                                 …        …        …
4796         222 E 80th St #3a, New York, NY 10075, USA  40.774350 -73.955879
4797            97-40 62nd Dr, Rego Park, NY 11374, USA  40.732538 -73.860152
4798              427 W 21st St, New York, NY 10011, USA  40.745882 -74.003398
4799     91-23 Corona Ave. #4b, Flushing, NY 11373, USA  40.742770 -73.872752
4800      460 Neptune Ave #14a, Brooklyn, NY 11224, USA  40.579147 -73.970949

[4801 rows x 17 columns]
```

Podział kolumn na dwie kategorie: dane numeryczne i dane kategoryczne.

[ ]:
```python
numeric_columns = {'PRICE', 'BEDS','BATH', 'PROPERTYSQFT', 'LATITUDE',␣
↪'LONGITUDE'}
category_columns = {'TYPE', 'BROKERTITLE', 'STATE', 'LONG_NAME', 'LOCALITY',␣
↪'ADDRESS', 'ADMINISTRATIVE_AREA_LEVEL_2', 'SUBLOCALITY', 'MAIN_ADDRESS',␣
↪'FORMATTED_ADDRESS', 'STREET_NAME'}
```

Zmienienie liter na małe w celu ujednolicenia danych.

[ ]:
```python
for column in category_columns:
    data[column] = data[column].str.lower()
```

Sprawdzenie unikalnych wartości oraz charakterystyki zbioru danych.

[ ]:
```python
data[list(category_columns)].nunique()
```

[ ]:
```
STATE                           308
FORMATTED_ADDRESS              4550
LONG_NAME                      2731
ADDRESS                        4582
LOCALITY                         11
BROKERTITLE                    1011
SUBLOCALITY                      21
MAIN_ADDRESS                   4583
ADMINISTRATIVE_AREA_LEVEL_2      29
TYPE                             13
STREET_NAME                     174
dtype: int64
```

[ ]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4801 entries, 0 to 4800
Data columns (total 17 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   BROKERTITLE                  4801 non-null   object
 1   TYPE                         4801 non-null   object
 2   PRICE                        4801 non-null   int64
 3   BEDS                         4801 non-null   int64
 4   BATH                         4801 non-null   float64
 5   PROPERTYSQFT                 4801 non-null   float64
 6   ADDRESS                      4801 non-null   object
 7   STATE                        4801 non-null   object
 8   MAIN_ADDRESS                 4801 non-null   object
 9   ADMINISTRATIVE_AREA_LEVEL_2  4801 non-null   object
 10  LOCALITY                     4801 non-null   object
 11  SUBLOCALITY                  4801 non-null   object
 12  STREET_NAME                  4801 non-null   object
```

```
 13  LONG_NAME                    4801 non-null   object
 14  FORMATTED_ADDRESS            4801 non-null   object
 15  LATITUDE                     4801 non-null   float64
 16  LONGITUDE                    4801 non-null   float64
dtypes: float64(4), int64(2), object(11)
memory usage: 637.8+ KB
```

[ ]:
```
data.describe().T
```

[ ]:
```
              count          mean           std          min            25%  \
PRICE        4801.0  2.356940e+06  3.135525e+07  2494.000000  499000.000000
BEDS         4801.0  3.356801e+00  2.602315e+00     1.000000       2.000000
BATH         4801.0  2.373861e+00  1.946962e+00     0.000000       1.000000
PROPERTYSQFT 4801.0  2.184208e+03  2.377141e+03   230.000000    1200.000000
LATITUDE     4801.0  4.071423e+01  8.767557e-02    40.499546      40.639375
LONGITUDE    4801.0 -7.394160e+01  1.010825e-01   -74.253033     -73.987143

                        50%           75%           max
PRICE         825000.000000  1.495000e+06  2.147484e+09
BEDS               3.000000  4.000000e+00  5.000000e+01
BATH               2.000000  3.000000e+00  5.000000e+01
PROPERTYSQFT    2184.207862  2.184208e+03  6.553500e+04
LATITUDE          40.726749  4.077192e+01  4.091273e+01
LONGITUDE        -73.949189 -7.387064e+01 -7.370245e+01
```

[ ]:
```
data.head()
```

[ ]:
```
                                         BROKERTITLE                TYPE  \
0        brokered by douglas elliman  -111 fifth ave      condo for sale
1                                brokered by serhant      condo for sale
2                             brokered by sowae corp      house for sale
3                                brokered by compass      condo for sale
4  brokered by sotheby's international realty - e…  townhouse for sale

       PRICE  BEDS       BATH  PROPERTYSQFT  \
0     315000     2   2.000000        1400.0
1  195000000     7  10.000000       17545.0
2     260000     4   2.000000        2015.0
3      69000     3   1.000000         445.0
4   55000000     7   2.373861       14175.0

                                             ADDRESS                    STATE  \
0                               2 e 55th st unit 803       new york, ny 10022
1  central park tower penthouse-217 w 57th new yo…       new york, ny 10019
2                                   620 sinclair ave  staten island, ny 10312
3                             2 e 55th st unit 908w33      manhattan, ny 10022
4                                        5 e 64th st       new york, ny 10065
```

```
                                        MAIN_ADDRESS  \
0             2 e 55th st unit 803new york, ny 10022
1  central park tower penthouse-217 w 57th new yo…
2            620 sinclair avestaten island, ny 10312
3         2 e 55th st unit 908w33manhattan, ny 10022
4                       5 e 64th stnew york, ny 10065

  ADMINISTRATIVE_AREA_LEVEL_2  LOCALITY      SUBLOCALITY       STREET_NAME  \
0             new york county  new york        manhattan  east 55th street
1               united states  new york  new york county          new york
2               united states  new york  richmond county     staten island
3               united states  new york  new york county          new york
4               united states  new york  new york county          new york

         LONG_NAME                                  FORMATTED_ADDRESS  \
0   regis residence  regis residence, 2 e 55th st #803, new york, n…
1  west 57th street            217 w 57th st, new york, ny 10019, usa
2   sinclair avenue     620 sinclair ave, staten island, ny 10312, usa
3  east 55th street              2 e 55th st, new york, ny 10022, usa
4  east 64th street              5 e 64th st, new york, ny 10065, usa

    LATITUDE  LONGITUDE
0  40.761255 -73.974483
1  40.766393 -73.980991
2  40.541805 -74.196109
3  40.761398 -74.974613
4  40.767224 -73.969856
```

Sprawdzenie braków danych.

[ ]:
```
data.isna().sum()
```

[ ]:
```
BROKERTITLE                    0
TYPE                           0
PRICE                          0
BEDS                           0
BATH                           0
PROPERTYSQFT                   0
ADDRESS                        0
STATE                          0
MAIN_ADDRESS                   0
ADMINISTRATIVE_AREA_LEVEL_2    0
LOCALITY                       0
SUBLOCALITY                    0
STREET_NAME                    0
LONG_NAME                      0
```

```
FORMATTED_ADDRESS              0
LATITUDE                       0
LONGITUDE                      0
dtype: int64
```

Sprawdzenie ilościi zduplikowanych wierszy i ich usunięcie.

```
[ ]: print('Duplicated rows: ', data.duplicated().sum())
     data.drop_duplicates(inplace=True)
```

```
Duplicated rows:  214
```

Usunięcie w kolumnie „BROKERTITLE" ciągów znaków 'llc', 'inc', zamiana wartości 'rlty' na 'realty' i zapisanie wyników tych zmian w nowej kolumnie „Broker".

BROKERTITLE

```
[ ]: data['BROKER'] = data['BROKERTITLE'].str.replace('llc','')
     data['BROKER'] = data['BROKER'].str.replace('inc','')
     data['BROKER'] = data['BROKER'].str.replace('rlty','realty')
     data['BROKER'] = data['BROKER'].str.replace('.','')
     def split_by_delimeter(value, separator):
         result = value.split(separator)[0] if separator in value else value
         result = result.strip()
         return result

     data['BROKER'] = data['BROKER'].apply(lambda x: split_by_delimeter(x, ' -'))
```

TYPE

```
[ ]: data.loc[data['TYPE'] == 'land for sale', ['BATH', 'BEDS']]
```

W kolumnie „TYPE" ciąg znaków "condop" zmieniony na "condo".

Stworzenie nowej kolumny „ANNOUNCEMENT_TYPE" która przypisuje ogłoszeniom odpowiednie kategorie na podstawie ich typu ('TYPE') przy użyciu wcześniej zdefiniowanego słownika.

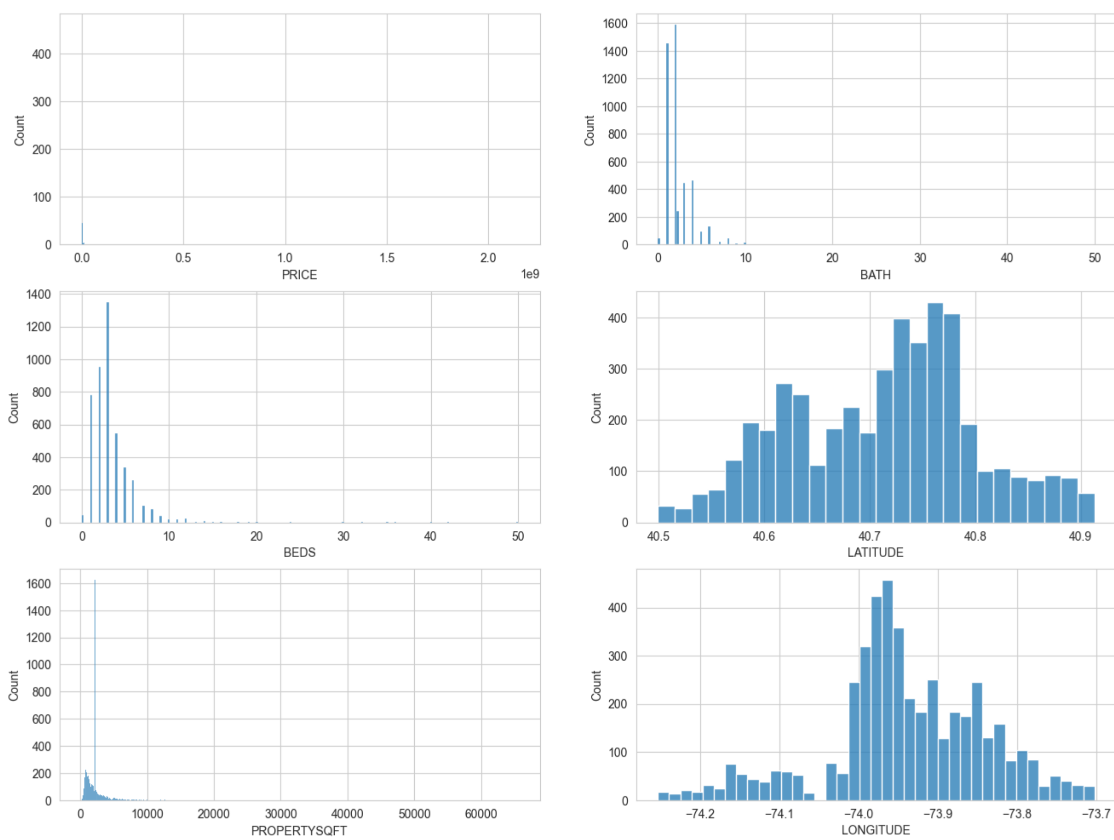
Usuwamy również ciąg znaków „for sale" z wartości w kolumnie TYPE.

```
[ ]: #Zamieniamy liczbe łazienek i sypialni na zero dla ogłoszeń, które są dla␣
      ↪sprzedaży działki
     data.loc[data['TYPE'] == 'land for sale', ['BATH', 'BEDS']]= 0
     data["TYPE"] = data["TYPE"].str.replace('condop', 'condo')

     data['ANNOUNCEMENT_TYPE'] = data['TYPE'].replace({
                                     'condo for sale': 'apartment',
                                      'townhouse for sale': 'home',
                                      'house for sale': 'home',
                                      'multi-family home for sale': 'home',
                                      'co-op for sale': 'co-op',
```

```
                                              'mobile house for sale': 'home',
                                              'land for sale': 'land',
                                              'foreclosure': 'other',
                                              'contingent': 'other',
                                              'pending': 'other',
                                              'coming soon': 'other',
                                             'for sale': 'other'})
data["TYPE"] = data["TYPE"].str.replace(" for sale", "")
```

PRICE - W kolumnie „PRICE” usuwamy wiersze, dla których cena jest większa niż 100 000 000 lub mniejsza niż 10 000 i zmienienie typ danych na liczby zmiennoprzecinkowe.

[ ]:
```
fig, axes = plt.subplots(3,2, figsize=(16, 12))
axes = axes.flatten()

i=0
for column in numeric_columns:
    sns.histplot(x=data[column], ax=axes[i])
    i=i+1

plt.show()
```



```
[ ]: data.drop(data.loc[(data["PRICE"] > 100000000) | (data["PRICE"]<10000)].index,␣
      ↪inplace= True)
     data['PRICE'] = data['PRICE'].astype(float)
```

BATH, BEDS

```
[ ]: data['BATH'].value_counts()
```

```
[ ]: data['BEDS'].value_counts()
```

Utworzenie nowej kolumny „PROPERTYSQFT1000", w której wartości są obliczane przez podzielenie wartości w kolumnie 'PROPERTYSQFT' przez 1000, zaokrąglając wynik w dół do liczby całkowitej , w celu znalezienia wartości odstających.

Zostały pozostawione wiersze z wartościami dla których metraż nieruchomości podzielony przez 1000 mieści się w zakresie od 0 do 8.

Pozostawiono wiersze z kolumny „BATH", w których liczba łazienek mieści się w zakresie od 0 do 10, ze względu na małą ilość wystąpień wartości spoza tego przedziału.

Pozostawiono wiersze z kolumny „BEDS" w których liczba sypialni mieści się w zakresie od 0 do 12, ze względu na małą ilość wystąpień wartości spoza tego przedziału.

```
[ ]: data['PROPERTYSQFT1000'] = data['PROPERTYSQFT'].apply(lambda x: x//1000)
     data['PROPERTYSQFT1000'].value_counts()
```

```
[ ]: data = data[(data['BATH'] >= 0) & (data['BATH'] <= 10)]
     data = data[(data['BEDS'] >= 0) & (data['BEDS'] <= 12)]
     data['BATH'] = data['BATH'].apply(lambda x: x if float.is_integer(x) else␣
      ↪int(x))
     data = data[(data['PROPERTYSQFT1000'] >= 0) & (data['PROPERTYSQFT1000'] <= 8)]

     data['PROPERTYSQFT'] = data['PROPERTYSQFT'].round(2)
```

Cleaning adresses

```
[ ]: data_api = data.copy()
```

Usunięcie kolumny „MAIN ADDRESS", ponieważ jest połączeniem kolumn „ADRESS" i „STATE".

```
[ ]: (data["ADDRESS"] + data["STATE"] == data["MAIN_ADDRESS"]).unique()
     data.drop("MAIN_ADDRESS", axis=1, inplace=True)
```

Utworzenie słownika „dictLocality", który mapuje nazwy hrabstw na nazwy dzielnic w Nowym Jorku.

Stworzenie listy „listOfBoroughs" zawierającej nazwy dzielnic (wartości powyżej wspomnianego słownika).

[ ]:
```python
dictLocality = {"new york county": "manhattan",  "kings county":
↪"brooklyn","bronx county":"the bronx", "richmond county": "staten island",␣
↪"queens county":"queens"}
listOfBoroughs = list(dictLocality.values())
```

Zamieniamy wartości wystepujace jako klucze w słowniku na odpowiadające im wartości we wszystkich kolumnach.

[ ]:
```python
for c in list(data.columns):
  for i in list(dictLocality.keys()):
    data[c] = data[c].astype(str).str.replace(i, dictLocality[i])
```

[ ]:
```python
# Słownik mapujący skrócone i niepoprawne nazwy stanów i dzielnic na ich␣
↪poprawne nazwy
state_replacements = {
    'nyc': 'new york',
    'ny': 'new york',
    'new yorkc': 'new york',
    'new york city': 'new york',
    r'kew gardens hill(?!s)': 'kew garden hills',
    r'kew gardens(?! hills)': 'kew garden hills',
    'kew gardens hills': 'kew garden hills',
    'kew gardens hillss': 'kew garden hills',
    r'(?<!\bthe\s)bronx new york': 'the bronx',
    'queens village': 'queens',
    r'(?<!\bthe\s)bronx ny': 'the bronx',
    'brooklyn heights': 'brooklyn',
    r'\b(?<!\bthe\s)(?<!\beast\s)(?<!\bwest\s)bronx\b': 'the bronx'
}


for col in ["ADDRESS", "ADMINISTRATIVE_AREA_LEVEL_2", "STATE", "LOCALITY",␣
↪"SUBLOCALITY", "STREET_NAME", "LONG_NAME", "FORMATTED_ADDRESS"]:
  for i in list(state_replacements.keys()):
    data[col] = data[col].str.replace(i, state_replacements[i], regex=True)
```

[ ]:
```python
print(data["LOCALITY"].unique())
# Przypisanie wartości z kolumny "LOCALITY" do kolumny "BOROUGH", jeśli wartość␣
↪znajduje się w liście istOfBoroughs.
data.loc[data["LOCALITY"].isin(listOfBoroughs), "BOROUGH"] = data['LOCALITY']
print(data["BOROUGH"].unique())
# Jeżeli wartość w kolumnie LOCALITY to flatbush to przepisujemy tą wartość do␣
↪kolumny NEIGHBOURHOOD
data.loc[data["LOCALITY"] == "flatbush", "NEIGHBOURHOOD"] = data['LOCALITY']
```

```
['new york' 'manhattan' 'the bronx' 'brooklyn' 'queens' 'staten island'
 'united states' 'flatbush']
```

```
[nan 'manhattan' 'the bronx' 'brooklyn' 'queens' 'staten island']
```

[ ]:
```python
print(data["SUBLOCALITY"].unique())
# Przepisanie wartości z SUBLOCALITY do kolumny BOROUGH, jeśli wartość jest␣
↪zawarta w listOfBoroughs oraz komórka w kolumnie BOROUGH jest pusta.
data.loc[(data["SUBLOCALITY"].isin(listOfBoroughs)) &(data["BOROUGH"].isna()) ,␣
↪"BOROUGH"] = data["SUBLOCALITY"]

# Jeżeli kolumna SUBLOCALITY zawiera "snyder avenue" to przepisujemy tą wartość␣
↪do kolumny STREET.
data.loc[data["SUBLOCALITY"] == "snyder avenue" , "STREET"] =␣
↪data["SUBLOCALITY"]

# Jeśli wartość w kolumnie SUBLOCALITY nie znajduje się w listOfBoroughs, nie␣
↪jest równa "new york" ani "snyder avenue" to przypisujemy tą wartość do␣
↪odpowiadającej komórki w kolumnie NEIGHBOURHOOD.
data.loc[(data["SUBLOCALITY"].isin(list(filter(lambda x: x not in␣
↪listOfBoroughs, list(data["SUBLOCALITY"]))))) & (~(data["SUBLOCALITY"].
↪isin(["new york", "snyder avenue"]))) , "NEIGHBOURHOOD"] =␣
↪data["SUBLOCALITY"]
```

```
['manhattan' 'staten island' 'brooklyn' 'new york' 'east bronx'
 'the bronx' 'queens' 'coney island' 'jackson heights' 'riverdale'
 'rego park' 'fort hamilton' 'flushing' 'dumbo' 'snew yorkder avenue']
```

[ ]:
```python
data
```

[ ]:
```
                                          BROKERTITLE   TYPE      PRICE BEDS  \
0       brokered by douglas elliman  -111 fifth ave  condo   315000.0    2
2                            brokered by sowae corp  house   260000.0    4
3                               brokered by compass  condo    69000.0    3
5                            brokered by sowae corp  house   690000.0    5
6     brokered by douglas elliman - 575 madison ave  condo   899500.0    2
…                                                 …     …          …   …
4796                            brokered by compass  co-op   599000.0    1
4797                  brokered by mjr real estate llc  co-op   245000.0    1
4798  brokered by douglas elliman - 575 madison ave  co-op  1275000.0    1
4799        brokered by e realty international corp  condo   598125.0    2
4800             brokered by nyc realty brokers llc  co-op   349000.0    1

     BATH PROPERTYSQFT                     ADDRESS  \
0     2.0       1400.0       2 e 55th st unit 803
2     2.0       2015.0           620 sinclair ave
3     1.0        445.0    2 e 55th st unit 908w33
5     2.0       4004.0                584 park pl
6     2.0      2184.21     157 w 126th st unit 1b
…    …           …                          …
```

```
4796  1.0      2184.21       222 e 80th st apt 3a
4797  1.0      2184.21        97-40 62 dr unit lg
4798  1.0      2184.21  427 w 21st st unit garden
4799  1.0        655.0   91-23 corona ave unit 4g
4800  1.0        750.0    460 neptune ave apt 14o

                              STATE ADMINISTRATIVE_AREA_LEVEL_2   LOCALITY  \
0          new york, new york 10022                   manhattan   new york
2     staten island, new york 10312               united states   new york
3         manhattan, new york 10022               united states   new york
5          brooklyn, new york 11238               united states   new york
6          new york, new york 10027                    new york  manhattan
…                               …                         …          …
4796      manhattan, new york 10075                    new york  manhattan
4797      rego park, new york 11374               united states   new york
4798       new york, new york 10011               united states   new york
4799       elmhurst, new york 11373                    new york     queens
4800       brooklyn, new york 11224                    new york   brooklyn

      …          LONG_NAME  \
0     …   regis residence
2     …   sinclair avenue
3     …  east 55th street
5     …        park place
6     …               157
…   …                 …
4796  …               222
4797  …        62nd drive
4798  …  west 21st street
4799  …             91-23
4800  …               460

                                      FORMATTED_ADDRESS    LATITUDE  \
0     regis residence, 2 e 55th st #803, new york, n…   40.761255
2     620 sinclair ave, staten island, new york 1031…  40.5418051
3            2 e 55th st, new york, new york 10022, usa  40.7613979
5            584 park pl, brooklyn, new york 11238, usa  40.6743632
6     157 w 126th st #1b, new york, new york 10027, usa   40.809448
…                                                   …          …
4796   222 e 80th st #3a, new york, new york 10075, usa    40.77435
4797      97-40 62nd dr, rego park, new york 11374, usa  40.7325379
4798       427 w 21st st, new york, new york 10011, usa  40.7458817
4799  91-23 corona ave. #4b, flushing, new york 1137…  40.7427705
4800  460 neptune ave #14a, brooklyn, new york 11224…   40.579147

        LONGITUDE                                  BROKER ANNOUNCEMENT_TYPE  \
0     -73.9744834              brokered by douglas elliman         apartment
```

```
2     -74.1961086                    brokered by sowae corp              home
3     -73.9746128                       brokered by compass         apartment
5     -73.9587248                    brokered by sowae corp              home
6      -73.946777               brokered by douglas elliman         apartment
…           …                                         …                 …
4796   -73.955879                       brokered by compass             co-op
4797  -73.8601516               brokered by mjr real estate             co-op
4798  -74.0033976               brokered by douglas elliman             co-op
4799  -73.8727516  brokered by e realty international corp         apartment
4800  -73.9709488            brokered by nyc realty brokers             co-op

     PROPERTYSQFT1000        BOROUGH NEIGHBOURHOOD STREET
0                 1.0      manhattan          NaN    NaN
2                 2.0  staten island          NaN    NaN
3                 0.0      manhattan          NaN    NaN
5                 4.0       brooklyn          NaN    NaN
6                 2.0      manhattan          NaN    NaN
…                …            …           …     …
4796              2.0      manhattan          NaN    NaN
4797              2.0         queens          NaN    NaN
4798              2.0      manhattan          NaN    NaN
4799              0.0         queens          NaN    NaN
4800              0.0       brooklyn          NaN    NaN

[4507 rows x 22 columns]
```

[ ]:
```python
print(data["ADMINISTRATIVE_AREA_LEVEL_2"].unique())
#Jeśli wartość z kolumny ADMINISTRATIVE_AREA_LEVEL_2 znajduje się w
 ↪listOfBoroughs, a kolumna "BOROUGH" jest pusta to uzupełniamy kolumne
 ↪Borough sprawdzaną wartością.
data.loc[(data["ADMINISTRATIVE_AREA_LEVEL_2"].isin(listOfBoroughs))
 ↪&(data["BOROUGH"].isna()) , "BOROUGH"] = data["ADMINISTRATIVE_AREA_LEVEL_2"]

# Uzupełnienie kolumny POSTCODE wartościami z kolumny
 ↪ADMINISTRATIVE_AREA_LEVEL_2, jeśli nie są zawarte w listOfBoroughs, nie są
 ↪równe "new york" ani "united states".
data.loc[(data["ADMINISTRATIVE_AREA_LEVEL_2"].isin(list(filter(lambda x: x not
 ↪in listOfBoroughs, list(data["ADMINISTRATIVE_AREA_LEVEL_2"]))))) &
 ↪(data["ADMINISTRATIVE_AREA_LEVEL_2"] != "new york") &
 ↪(data["ADMINISTRATIVE_AREA_LEVEL_2"] != "united states") , "POSTCODE"] =
 ↪data["ADMINISTRATIVE_AREA_LEVEL_2"]
```

```
['manhattan' 'united states' 'new york' 'the bronx' '11214' '10301'
 '10309' '10303' '11234' '11414' '10310' '10003' '11417' '10304'
 'brooklyn' '10463' 'queens' '10017' '10306' '10471' '11229' '10312'
 '11412' '10465' '10002' '10466' '11237' '11218']
```

[ ]:
```
data["STREET_NAME"].unique()
```

[ ]:
```
array(['east 55th street', 'staten island', 'new york', 'brooklyn',
       'manhattan', 'morrison avenue', 'midwood', 'concourse village',
       'flushing', 'elmhurst', 'annadale', 'queens', 'fort hamilton',
       'north riverdale', 'rego park', 'forest hills', 'the bronx',
       'dongan hills', 'jackson heights', 'clifton', 'mariners harbor',
       'dyker heights', 'williamsburg', 'concourse', 'mid island',
       'centre street', 'cobble hill', 'park slope', 'brighton beach',
       'flatbush', 'prospect heights', 'woodhaven', 'bedford-stuyvesant',
       'jamaica', 'spuyten duyvil', 'bay ridge', 'shore acres', 'bayside',
       'glen oaks', 'fresh meadows', 'highbridge', 'sheepshead bay',
       'rector place', 'kew garden hills', 'bushwick', 'hudson hill',
       'rosedale', 'east bronx', 'parkchester', 'borough park',
       'little haiti', 'canarsie', 'kensington', 'east 110th street',
       'east new york', 'pelham bay', 'howard beach', 'downtown brooklyn',
       'city island', 'fieldston', 'westchester square', 'rockaway park',
       'riverdale', 'bulls head', 'gerritsen beach', 'crown heights',
       'new dorp', 'west bronx', 'ocean hill', 'gravesend',
       'castleton corners', 'seagate', 'gowanus', 'windsor terrace',
       'bath beach', 'norwood', 'whitestone', 'surf avenue',
       'great kills', 'homecrest', 'long island city', 'woodside',
       'maspeth', 'sunset park', 'douglaston', 'astoria', 'dumbo',
       'new springville', 'corona', 'madison', 'bensonhurst',
       'fordham manor', 'greenpoint', 'coney island', 'mill basin',
       'carroll gardens', 'old fulton street', 'ozone park',
       'east 74th street', 'floral park', '35th avenue', 'far rockaway',
       'beechhurst', 'henry hudson parkway', 'rosebank', 'kingsbridge',
       'bergen beach', 'ridgewood', 'oakwood', 'clinton hill',
       'richmond hill', 'auburndale', 'southside', 'little caribbean',
       'peck slip', 'fort greene', 'boerum hill', 'foxhurst', 'red hook',
       'columbia street waterfront district', 'east flatbush',
       'sunnew yorkside', 'flatlands', 'mapleton', '98th place',
       'east 22nd street', 'manhattan beach', 'east elmhurst',
       'oxford avenue', 'bay terrace', 'arverne', 'midland beach',
       'vinegar hill', 'little neck', 'west brighton', 'east 96th street',
       'clason point', '5th avenue', 'shore road', 'west 56th street',
       'middle village', 'west 111th street', 'prospect lefferts gardens',
       'woodstock', 'melrose', 'east 10th street', '139th street',
       'park avenue', 'west 65th street', 'john street',
       'east end avenue', 'brownsville', '3g', 'west 13th street',
       'college point', 'central park west', 'east 88th street',
       'allerton', 'morrisania', 'west 64th street', '61st street',
       'cypress hills', '2501', '67th drive', 'todt hill',
       'saunders street', 'mount eden'], dtype=object)
```

[ ]:
```
len(data["STREET_NAME"].unique())
```

[ ]: 167

Utworzenie słownika z rozwinięciami skrótów i zamiana wartości z kolumny na wartości ze słownika. Nazwy ulicy zostają dodane do kolumny “streets”, jeśli zawierają jedno ze słów kluczowych. Przypisanie wartości z kolumn do kolumn dla wierszy, w których wartość znajduje się na liście i jednocześnie wartość w kolumnie jest pusta np. kolumny “STREET_NAME” i “BOROUGH”, “STREET_NAME” i “NEIGHBOURHOOD”, “STREET_NAME” i “STREET” . Uzupełnienie brakujących informacji w w kolumnach “POSTCODE”, “BOROUGH” i “NEIGHBOURHOOD”. Przypisanie wartości do konkretnych kolumn.

[ ]:
```
addresses_shortcut = {
        ' st': ' street',
        ' ave': ' avenue',
        ' rd': ' road',
        ' blvd': ' boulevard',
        ' dr': ' drive',
        ' pkwy': ' parkway',
        ' ct': ' court',
        ' ln': ' lane',
        ' pl': ' place',
        ' sq': ' square',
        ' apt': '',
        ' ste': '',
        ' num': ''
    }
for c in ["STREET_NAME", "ADDRESS", "STATE", "LONG_NAME", "FORMATTED_ADDRESS"]:
  for add in addresses_shortcut.keys():
    mask = data[c].str.contains(fr'\b{add}\b')
    data.loc[mask, c] = data.loc[mask, c].str.replace(add,␣
 ↪addresses_shortcut[add])

streets = []
keyWords = ["street", "parkway", "avenue", "drive", "road"]
for street_name in data["STREET_NAME"]:
    for keyword in keyWords:
        if keyword in street_name:
            streets.append(street_name)

streets
```

[ ]:
```
# Jeśli wartość z kolumny STREET_NAME znajduje się w listOfBoroughs, a kolumna␣
 ↪BOROUGH jest pusta, uzupełniamy kolumnę BOROUGH tą wartością.
data.loc[(data["STREET_NAME"].isin(listOfBoroughs))& (data["BOROUGH"].isna()),␣
 ↪"BOROUGH"] = data["STREET_NAME"]
```

```
# Uzupełnienie pustych wartości w kolumnie NEIGHBOURHOOD wartością z kolumny␣
↪STREET_NAME, jeśli nie jest zawarta w liście streets, nie jest równa "new␣
↪york", nie należy do listOfBoroughs.
data.loc[(data["STREET_NAME"].isin(list(filter(lambda x: (x not in streets) &␣
↪(x != "new york") & (x not in listOfBoroughs), list(data["STREET_NAME"])))))␣
↪& (data["NEIGHBOURHOOD"].isna()), "NEIGHBOURHOOD"] = data["STREET_NAME"]
data.loc[(data["STREET_NAME"].isin(streets)) & (data["STREET"].isna()),␣
↪"STREET"] = data["STREET_NAME"]
```

[ ]:
```
print(data["STATE"].unique())
# Uzupełnienie pustych wartości w kolumnie POSTCODE wartościami z kolumny STATE.
data.loc[data["POSTCODE"].isna(), "POSTCODE"] = data["STATE"].str.slice(-5)

# Jeżeli BOROUGH jest pusta to weź wartość z kolumny STATE po przecinku - pod␣
↪warunkiem że należy do listOfBoroughs
data.loc[(data["BOROUGH"].isna()) &(data["STATE"].str.split(", ").str.get(0).
↪isin(listOfBoroughs)) , "BOROUGH"] = data["STATE"].str.split(", ").str.get(0)

# Jeżeli NEIGHBOURHOOD jest pusta to weź wartość z kolumny STATE po przecinku -␣
↪pod warunkiem że nie należy do listOfBoroughs ani nie jest rowna "new york",␣
↪"ny", "nyc"
data.loc[(data["NEIGHBOURHOOD"].isna()) & (~(data["STATE"].str.split(", ").str.
↪get(0).isin(listOfBoroughs))) &(~(data["STATE"].str.split(", ").str.get(0).
↪isin(["new york", "ny", "nyc" ]))) , "NEIGHBOURHOOD"] = data["STATE"].str.
↪split(", ").str.get(0)
```

[ ]:
```
data.loc[data["POSTCODE"].isna(), "POSTCODE"] = data["FORMATTED_ADDRESS"].str.
↪split(", ").str.get(-2).str.slice(-5)
data.loc[(data["BOROUGH"].isna()) & (data["FORMATTED_ADDRESS"].str.split(", ").
↪str.get(-3).isin(listOfBoroughs)), "BOROUGH"] = data["FORMATTED_ADDRESS"].
↪str.split(", ").str.get(-3)
data.loc[(data["NEIGHBOURHOOD"].isna()) & (~(data["FORMATTED_ADDRESS"].str.
↪split(", ").str.get(-3).isin(listOfBoroughs))) &␣
↪(~(data["FORMATTED_ADDRESS"].str.split(", ").str.get(-3).isin(["new␣
↪york"]))), "NEIGHBOURHOOD"] = data["FORMATTED_ADDRESS"].str.split(", ").str.
↪get(-3)
```

[ ]:
```
data.loc[data["FORMATTED_ADDRESS"].str.split(", ").str.get(-4).str.split().str.
↪get(0).str.replace("-","").str.contains('\d'), "HOUSE_NUMBER"] =␣
↪data["FORMATTED_ADDRESS"].str.split(", ").str.get(-4).str.split().str.get(0)
data.loc[(data["STREET"].isna()) & (data["FORMATTED_ADDRESS"].str.split(", ").
↪str.get(-4).str.split().str.get(0).str.replace("-","").str.contains('\d')),␣
↪"STREET"] = data["FORMATTED_ADDRESS"].str.split(", ").str.get(-4).str.
↪split().str.slice(start=1).str.join(" ")
```

```python
data.loc[(data["STREET"].isna()) & (~(data["FORMATTED_ADDRESS"].str.split(", ").
↪str.get(-4).str.split().str.get(0).str.replace("-","").str.contains('\d'))),␣
↪"STREET"] = data["FORMATTED_ADDRESS"].str.split(", ").str.get(-4)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
<>:3: SyntaxWarning: invalid escape sequence '\d'
<>:1: SyntaxWarning: invalid escape sequence '\d'
<>:2: SyntaxWarning: invalid escape sequence '\d'
<>:3: SyntaxWarning: invalid escape sequence '\d'
/var/folders/h4/pdn3pcp16vxc0wjz5jwhfhm40000gn/T/ipykernel_48297/2895737151.py:1
: SyntaxWarning: invalid escape sequence '\d'
  data.loc[data["FORMATTED_ADDRESS"].str.split(",
").str.get(-4).str.split().str.get(0).str.replace("-","").str.contains('\d'),
"HOUSE_NUMBER"] = data["FORMATTED_ADDRESS"].str.split(",
").str.get(-4).str.split().str.get(0)
/var/folders/h4/pdn3pcp16vxc0wjz5jwhfhm40000gn/T/ipykernel_48297/2895737151.py:2
: SyntaxWarning: invalid escape sequence '\d'
  data.loc[(data["STREET"].isna()) & (data["FORMATTED_ADDRESS"].str.split(",
").str.get(-4).str.split().str.get(0).str.replace("-","").str.contains('\d')),
"STREET"] = data["FORMATTED_ADDRESS"].str.split(",
").str.get(-4).str.split().str.slice(start=1).str.join(" ")
/var/folders/h4/pdn3pcp16vxc0wjz5jwhfhm40000gn/T/ipykernel_48297/2895737151.py:3
: SyntaxWarning: invalid escape sequence '\d'
  data.loc[(data["STREET"].isna()) & (~(data["FORMATTED_ADDRESS"].str.split(",
").str.get(-4).str.split().str.get(0).str.replace("-","").str.contains('\d'))),
"STREET"] = data["FORMATTED_ADDRESS"].str.split(", ").str.get(-4)
```

[ ]:
```python
data.loc[(data["LONG_NAME"].str.replace("-", "").str.isdigit()),␣
↪"HOUSE_NUMBER"] = data["LONG_NAME"]
data.loc[(~(data["LONG_NAME"].str.replace("-", "").str.isdigit())) &␣
↪(data["STREET"].isna()) & (data["LONG_NAME"] != "parking lot"), "STREET"] =␣
↪data["LONG_NAME"]
```

[ ]:
```python
data.loc[(data["ADDRESS"].str.split(" ").str.get(0).str.replace("-", "").str.
↪isdigit()) & (data["HOUSE_NUMBER"].isna()), "HOUSE_NUMBER"] =␣
↪data["ADDRESS"].str.split(" ").str.get(0).str.split().str.get(0)
```

[ ]:
```python
data.loc[data["ADDRESS"] == "98A-98G Discala Ln", "HOUSE_NUMBER"] = "98A-98G"
```

Zamiana skrótów kierunków na pełne nazwy, usunięcie zbędnych znaków w nazwach ulic, zamiana liter na małe, usunięcie skrótów i pojedynczych liter w nazwach ulic. Usunięcie części kolumn i duplikatów.

[ ]:
```python
import re
#Zamiana skrótów na pełne nazwy
```

```python
data.loc[data["STREET"].str.contains(r'\bE\b'), "STREET"] = data["STREET"].str.
↪replace("e", "east")
data.loc[data["STREET"].str.contains(r'\bW\b'), "STREET"] = data["STREET"].str.
↪replace("w", "west")
data.loc[data["STREET"].str.contains(r'\bS\b'), "STREET"] = data["STREET"].str.
↪replace("s", "south")
data.loc[data["STREET"].str.contains(r'\bN\b'), "STREET"] = data["STREET"].str.
↪replace("n", "north")

#Czyszczenie nazw ulic
data.loc[data["STREET"].str.contains(r' #.*'), "STREET"] = data["STREET"].str.
↪replace(r' #.*', "")
data["STREET"]= data["STREET"].str.replace(".", "")
data["STREET"]= data["STREET"].str.lower()
data["STREET"].unique()

def clean_street_names(street_name):

    parts = street_name.split(' ')
    cleaned_parts = [part for part in parts if re.match(r'^\d*(?:th|st|rd|nd)?
↪$', part) or not any(c.isdigit() for c in part) ]
    cleaned_parts = [part for part in cleaned_parts if not part.isdigit() and␣
↪len(part) != 1]

    last_part = cleaned_parts[-1] if cleaned_parts else None
    if last_part and any(character.isdigit() for character in last_part):
        cleaned_parts.remove(last_part)

    return ' '.join(cleaned_parts)

data['STREET'] = data['STREET'].apply(clean_street_names)

data['STREET']
```

[ ]:
```
0        east 55th street
2         sinclair avenue
3             55th street
5              park place
6            126th street
                  …
4796          80th street
4797           62nd drive
4798          21st street
4799        corona avenue
4800       neptune avenue
Name: STREET, Length: 4507, dtype: object
```

[ ]:
```
data.columns
```

[ ]:
```
Index(['BROKERTITLE', 'TYPE', 'PRICE', 'BEDS', 'BATH', 'PROPERTYSQFT',
       'ADDRESS', 'STATE', 'ADMINISTRATIVE_AREA_LEVEL_2', 'LOCALITY',
       'SUBLOCALITY', 'STREET_NAME', 'LONG_NAME', 'FORMATTED_ADDRESS',
       'LATITUDE', 'LONGITUDE', 'BROKER', 'ANNOUNCEMENT_TYPE',
       'PROPERTYSQFT1000', 'BOROUGH', 'NEIGHBOURHOOD', 'STREET', 'POSTCODE',
       'HOUSE_NUMBER'],
      dtype='object')
```

[ ]:
```
data.drop(columns = ['BROKERTITLE',
       'ADDRESS', 'STATE', 'ADMINISTRATIVE_AREA_LEVEL_2', 'LOCALITY',
       'SUBLOCALITY', 'STREET_NAME', 'LONG_NAME', 'FORMATTED_ADDRESS',
       'PROPERTYSQFT1000'], inplace=True)
```

[ ]:
```
category_columns = list(set(data.columns) - set(numeric_columns))
```

[ ]:
```
for column in category_columns:
    data[column] = data[column].str.lower()
```

[ ]:
```
data.columns
```

[ ]:
```
Index(['TYPE', 'PRICE', 'BEDS', 'BATH', 'PROPERTYSQFT', 'LATITUDE',
       'LONGITUDE', 'BROKER', 'ANNOUNCEMENT_TYPE', 'BOROUGH', 'NEIGHBOURHOOD',
       'STREET', 'POSTCODE', 'HOUSE_NUMBER'],
      dtype='object')
```

[ ]:
```
print('Duplicated rows: ', data.duplicated().sum())
data.drop_duplicates(inplace=True)
```

```
Duplicated rows:  1
```

[ ]:
```
data.to_excel('data.xlsx')
```

GEOPY Pobieranie danych adresowych z geolokatora na podstawie współrzędnych geograficznych w celu uzupełnienia danych w kolumnie neighbourhood.

[ ]:
```
#kod został zakomentowany aby dane nie zostaly utracone przy uruchamianiu kodu
"""from geopy.geocoders import Nominatim
import pandas as pd

data_api = pd.read_excel('data.xlsx')

data_api['API_ADDRESS_NAME'] = None
data_api['API_ADDRESS'] = None
api_keys = set()
```

```
num = 0
geolocator = Nominatim(user_agent="key_for_library1")

api_keys = set()
try:
    # Pętla służąca do pobrania danych dla każdego wiersza w ramce.
    for ind, row in data_api.iterrows():
        latitude = row['LATITUDE']
        longitude = row['LONGITUDE']
        address = geolocator.reverse(f"{latitude},{longitude}")
        data_api.at[ind, 'API_ADDRESS_NAME'] = address.raw['display_name']
        data_api.at[ind, 'API_ADDRESS'] = address.raw
        keys = set(address.raw['address'].keys())
        api_keys.update(keys)
        print(ind)
except Exception as e:
    data_api.to_excel('api.xlsx')

data_api.to_excel('api.xlsx')


data_api = pd.read_excel('api.xlsx')

geolocator = Nominatim(user_agent="key_for_library2")

try:
    # Pętla służąca do pobrania danych dla każdego wiersza w ramce.
    for ind, row in data_api.iterrows():
        if ind>= 2010:
            latitude = row['LATITUDE']
            longitude = row['LONGITUDE']
            address = geolocator.reverse(f"{latitude},{longitude}")
            data_api.at[ind, 'API_ADDRESS_NAME'] = address.raw['display_name']
            data_api.at[ind, 'API_ADDRESS'] = address.raw
            keys = set(address.raw['address'].keys())
            api_keys.update(keys)
            print(ind)
except Exception as e:
    data_api.to_excel('api_2.xlsx')

data_api.to_excel('api_2.xlsx')
print('Ilość kluczy: ',len(api_keys))
print('Klucze: ',api_keys)
"""
```

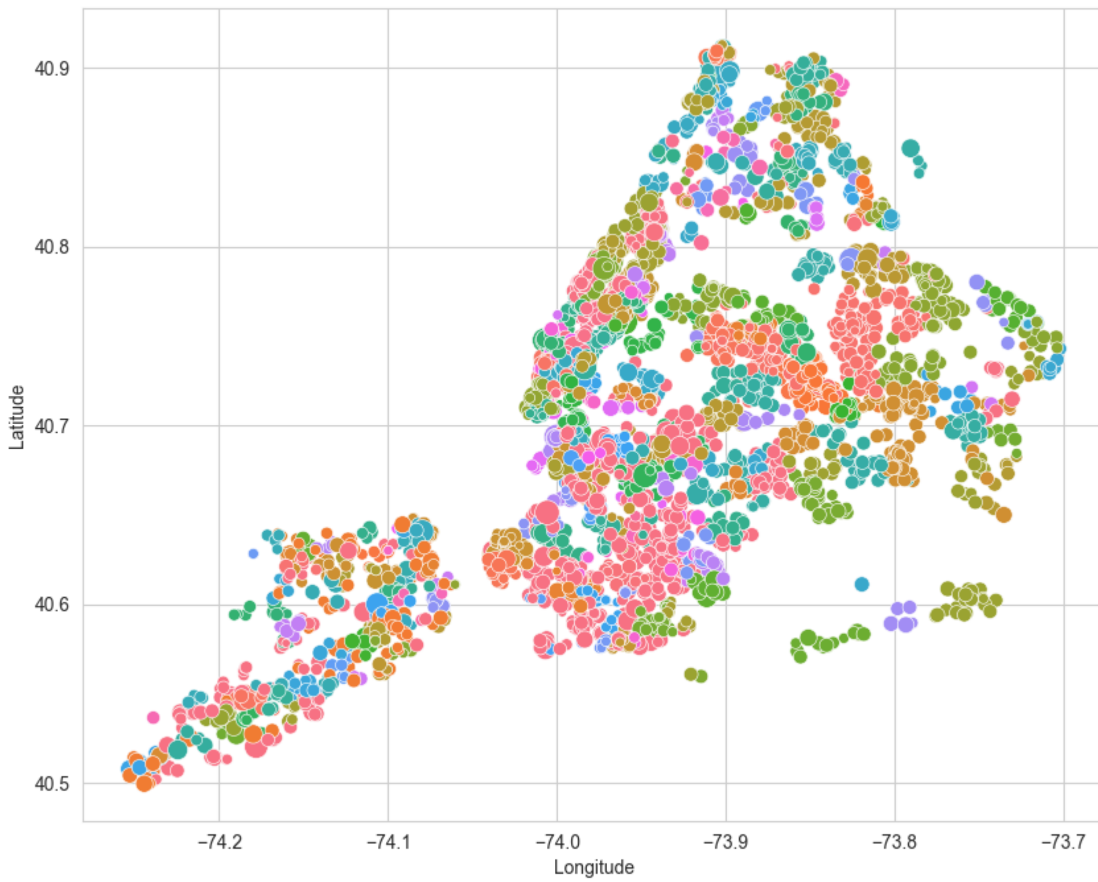
[ ]:
```
'from geopy.geocoders import Nominatim\nimport pandas as pd\n\ndata_api =
pd.read_excel(\'data.xlsx\')\n\ndata_api[\'API_ADDRESS_NAME\'] =
```

```
None\ndata_api[\'API_ADDRESS\'] = None\napi_keys = set()\n\nnum = 0\ngeolocator
= Nominatim(user_agent="key_for_library1")\n\napi_keys = set()\ntry:\n    #
Pętla służąca do pobrania danych dla każdego wiersza w ramce.\n    for ind, row
in data_api.iterrows():\n        latitude = row[\'LATITUDE\']\n        longitude
= row[\'LONGITUDE\']\n        address =
geolocator.reverse(f"{latitude},{longitude}")\n        data_api.at[ind,
\'API_ADDRESS_NAME\'] = address.raw[\'display_name\']\n        data_api.at[ind,
\'API_ADDRESS\'] = address.raw\n        keys =
set(address.raw[\'address\'].keys())\n        api_keys.update(keys)\n
print(ind)\nexcept Exception as e:\n
data_api.to_excel(\'api.xlsx\')\n\ndata_api.to_excel(\'api.xlsx\')\n\n\ndata_api
= pd.read_excel(\'api.xlsx\')\n\ngeolocator =
Nominatim(user_agent="key_for_library2")\n\ntry:\n    # Pętla służąca do
pobrania danych dla każdego wiersza w ramce.\n    for ind, row in
data_api.iterrows():\n        if ind>= 2010:\n            latitude =
row[\'LATITUDE\']\n            longitude = row[\'LONGITUDE\']\n
address = geolocator.reverse(f"{latitude},{longitude}")\n
data_api.at[ind, \'API_ADDRESS_NAME\'] = address.raw[\'display_name\']\n
data_api.at[ind, \'API_ADDRESS\'] = address.raw\n            keys =
set(address.raw[\'address\'].keys())\n            api_keys.update(keys)\n
print(ind)\nexcept Exception as e:\n    data_api.to_excel(\'api_2.xlsx\')\n\ndat
a_api.to_excel(\'api_2.xlsx\')\nprint(\'Ilość kluczy:
\',len(api_keys))\nprint(\'Klucze: \',api_keys)\n'
```

[ ]:
```python
import pandas as pd
import json

api_keys = ['neighbourhood', 'postcode', 'road', 'house_number']

data = pd.read_excel('api_2.xlsx')

#strukturyzowanie slownika danych uzyskanych dzięki API
for key in api_keys:
    for ind in data.index:
        try:
            row = data.at[ind, 'API_ADDRESS']
            row = str(row).replace("'", "\"")
            row = json.loads(row)
            keys = list(row['address'].keys())
            if key in keys:
                data.at[ind, key] = row['address'][key].lower()
            else:
                data.at[ind, key] = ''
        except Exception as err:
            continue
data['NEIGHBOURHOOD'] = data['NEIGHBOURHOOD'].fillna(data['neighbourhood'])
data['NEIGHBOURHOOD'] = data['NEIGHBOURHOOD'].str.lower()
```

```
data = data[['TYPE', 'PRICE', 'BEDS', 'BATH', 'PROPERTYSQFT', 'LATITUDE',␣
 ↪'LONGITUDE', 'BROKER', 'ANNOUNCEMENT_TYPE', 'BOROUGH', 'STREET', 'POSTCODE',␣
 ↪'HOUSE_NUMBER', 'NEIGHBOURHOOD']]

#data.to_excel('cat_api.xlsx')
```

[ ]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(data=data, x='LONGITUDE', y='LATITUDE', hue='NEIGHBOURHOOD',␣
 ↪size='PROPERTYSQFT', sizes=(20, 200), legend=False)
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



[ ]:
```
data['NEIGHBOURHOOD'] = data['NEIGHBOURHOOD'].fillna('')
data['NEIGHBOURHOOD'].value_counts()
```

[ ]:
```
NEIGHBOURHOOD
prospect heights            669
flushing                    202
```

```
manhattan community board 8    180
jamaica                         99
forest hills                    88
                              …
new brighton                     1
university heights               1
kips bay                         1
belle harbor                     1
chinatown                        1
Name: count, Length: 282, dtype: int64
```

[ ]:
```python
data['NEIGHBOURHOOD'] = data['NEIGHBOURHOOD'].apply(lambda x: x if all(borough␣
↪not in x for borough in listOfBoroughs) and 'bronx' not in x else '')
```

[ ]:
```python
data['NEIGHBOURHOOD'] = data['NEIGHBOURHOOD'].replace('', None)
data['NEIGHBOURHOOD'].value_counts()
```

[ ]:
```
NEIGHBOURHOOD
prospect heights      669
flushing              202
jamaica                99
forest hills           88
streetaten island      73
                     …
westchester square      1
vinegar hill            1
parkville               1
peck slip               1
chinatown               1
Name: count, Length: 264, dtype: int64
```

[ ]:
```python
data.isna().sum()
```

[ ]:
```
TYPE                  0
PRICE                 0
BEDS                  0
BATH                  0
PROPERTYSQFT          0
LATITUDE              0
LONGITUDE             0
BROKER                0
ANNOUNCEMENT_TYPE     0
BOROUGH               0
STREET                0
POSTCODE              0
HOUSE_NUMBER          3
NEIGHBOURHOOD       791
```

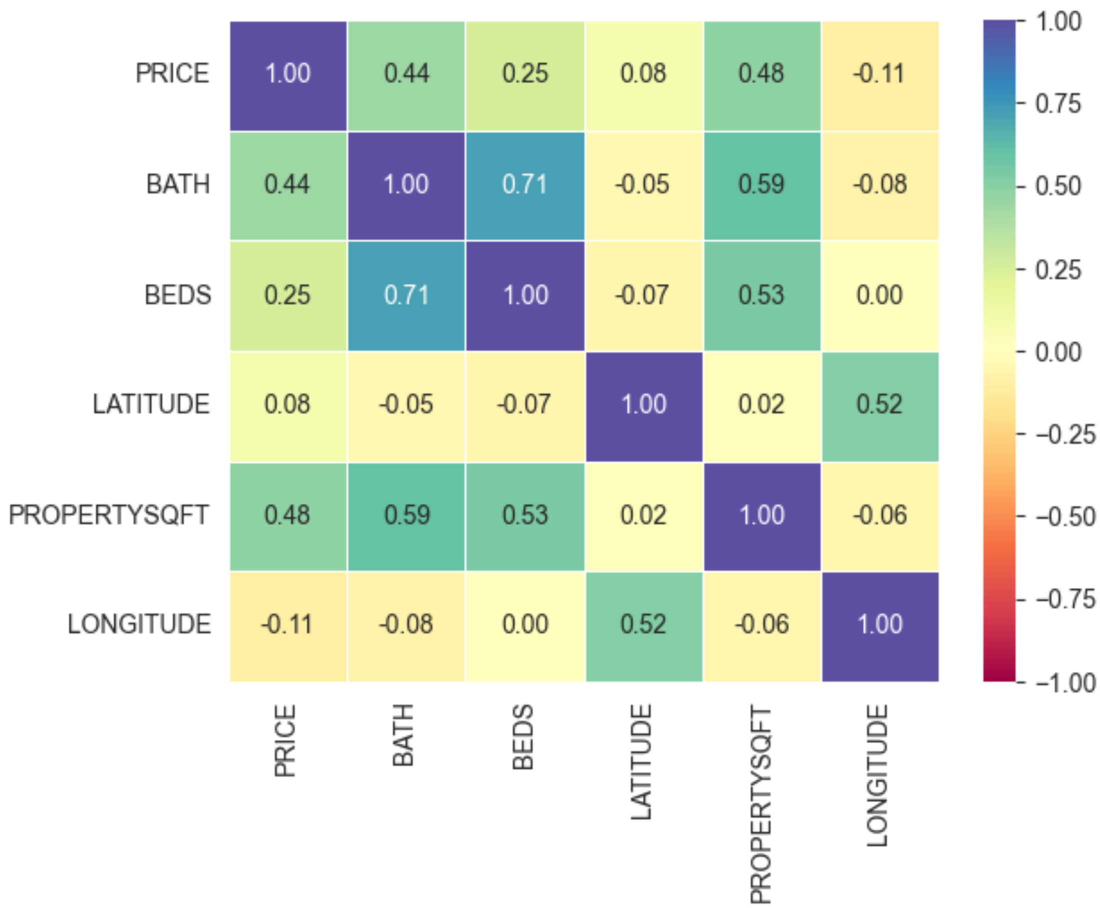
```
dtype: int64
```

[ ]:
```python
category_columns = list(set(data.columns) - numeric_columns)
numeric_columns = list(numeric_columns)
data[list(category_columns)].nunique()
```

[ ]:
```
BROKER                947
TYPE                   12
POSTCODE              178
HOUSE_NUMBER         2397
ANNOUNCEMENT_TYPE       5
STREET               1714
NEIGHBOURHOOD         264
BOROUGH                 5
dtype: int64
```

[ ]:
```python
sns.heatmap(data[numeric_columns].corr(), annot=True, cmap='Spectral',␣
 ↪linewidths=0.5,fmt=".2f", vmax=1, vmin=-1)
```

[ ]:
```
<Axes: >
```



[ ]:
```
#data.to_excel('ssdata.xlsx')
```

[ ]:
```
# Dopasowanie i przeskalowanie danych dla danej kolumny
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
numeric_columns.remove('PRICE')

mm_scalers = {}
for column in numeric_columns:
    mm_scalers[column] = MinMaxScaler()
    data[column] = mm_scalers[column].fit_transform(data[column].values.
↪reshape(-1, 1))
```

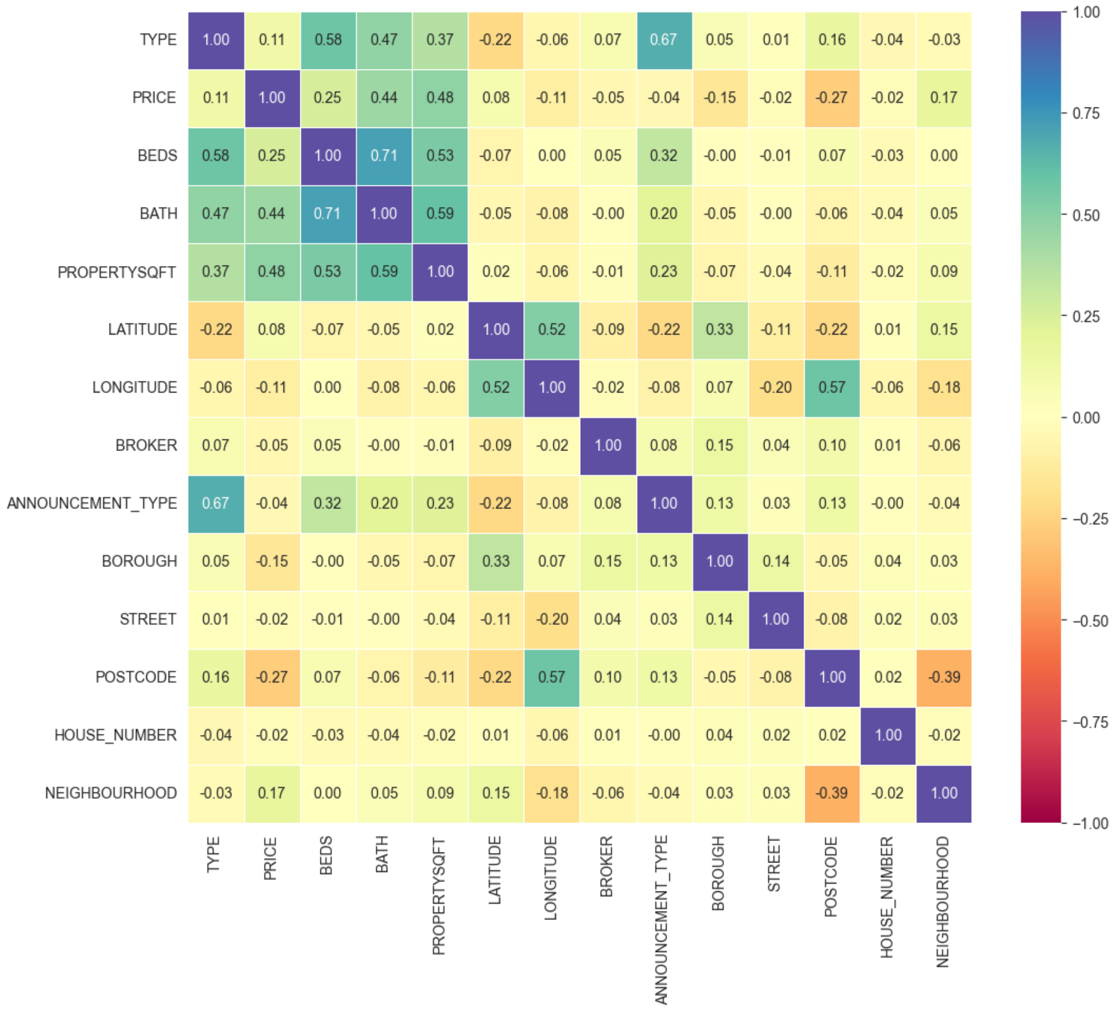
5

[ ]:
```
#Zamiana kategorii na wartości liczbowe
from sklearn.preprocessing import LabelEncoder
label_encoders = {}
for column in category_columns:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])
```

[ ]:
```
for column in category_columns:
    for class_index, class_name in enumerate(label_encoders[column].classes_):
        print(f"{class_name}: {class_index}")
```

[ ]:
```
plt.figure(figsize=(12, 10))
sns.heatmap(data.corr(), annot=True, cmap='Spectral', linewidths=0.5,fmt=".2f",␣
↪vmax=1, vmin=-1)
```

[ ]:
```
<Axes: >
```



Utworzenie macierzy korelacji dla wszystkich kolumn.

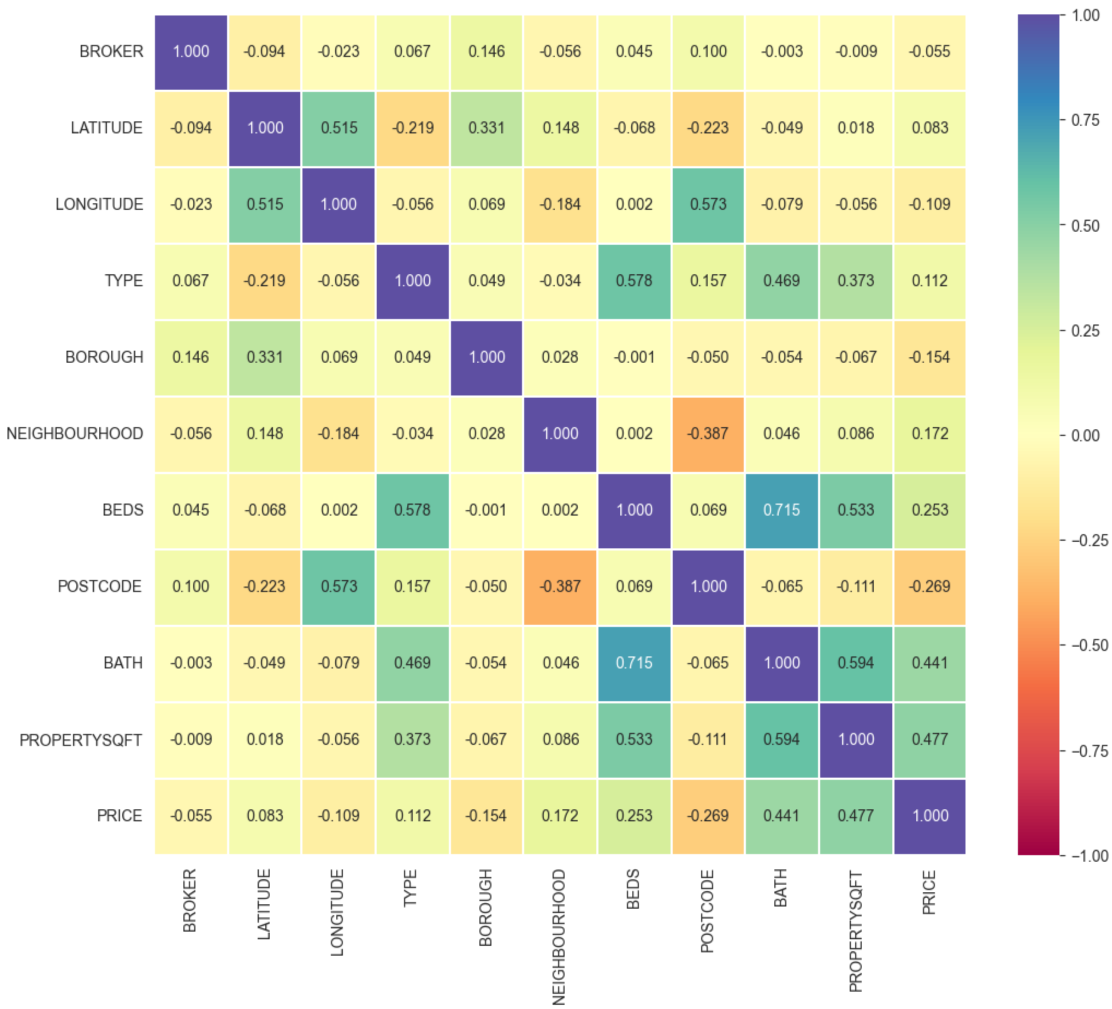
Stworzenie data_relevant, zawierającej kolumny z dataframe data, których bezwzględna wartość korelacji z cenami nieruchomości wynosi co najmniej 0.03.

[ ]:
```
corr_matrix = data.corr()
# Lista przechowująca pary kolumn, których współczynnik korelacji przekracza 0.8
pairs = []
for i in range(len(corr_matrix.columns)):
  for j in range(i+1, len(corr_matrix.columns)):
    if abs(corr_matrix.iloc[i, j]) > 0.8:
      pairs.append((corr_matrix.columns[i], corr_matrix.columns[j], corr_matrix.
↪iloc[i, j]))

for pair in pairs:
  print(pair)
```

```
[ ]: #Filtrowanie DataFrame tak aby zawierał tylko kolumny silnie skorelowane z ceną
     corr_matrix = data.corr()
     corr_df = pd.DataFrame(abs(corr_matrix['PRICE'].drop('PRICE')).
      ↪sort_values(ascending=True))
     relevant_columns = corr_df[abs(corr_df['PRICE'])>=0.05].index
     relevant_columns = list(relevant_columns)
     relevant_columns.append('PRICE')
     data_relevant = data[relevant_columns]
```

```
[ ]: plt.figure(figsize=(12, 10))
     sns.heatmap(data_relevant.corr(), annot=True, cmap='Spectral', linewidths=0.
      ↪3,fmt=".3f", vmax=1, vmin=-1)
```

```
[ ]: <Axes: >
```



[ ]:
```
#data.to_excel('clean_data.xlsx')
data_relevant.to_excel('clean_data_relevant.xlsx')
```

Analiza głównych składowych PCA na danych: określenie liczby składowych głównych, stworzenie listy składowych głównych i nazw kolumn dla wynikowych składowych głównych, stworzenie instancji klasy PCA. Stworzenie ramki danych zawierającą przekształcone wartości zmiennych za pomocą PCA.

[ ]:
```
from sklearn.decomposition import PCA
X = data_relevant.drop(columns =['PRICE'])
y = data_relevant['PRICE']

n_comp = 3
col_names = ['feature_'+str(i) for i in range(0,n_comp)]
pca = PCA(n_components=n_comp)
data_relevant_pca = pd.DataFrame(pca.fit_transform(X), columns = col_names)
print(pca.explained_variance_ratio_)
data_relevant_pca
```

```
[0.88876572 0.086828   0.02420772]
```

[ ]:

	feature_0	feature_1	feature_2
0	-161.996266	-115.604465	26.933015
1	405.600518	-113.137165	1.529599
2	-221.973645	-114.140947	26.179353
3	408.315389	-26.271092	-46.062553
4	-161.897677	-113.999705	22.199013
…	…	…	…
4501	-221.522691	-106.138266	5.696170
4502	155.724427	-17.484736	-69.383974
4503	-162.175676	-118.498272	35.479340
4504	-145.177221	89.001470	-38.528559
4505	219.603802	97.280650	5.694335

[4506 rows x 3 columns]

[ ]:
```
data_relevant_pca.to_excel('clean_data_relevant_pca.xlsx')
```

Planujemy użyć zarówno clean_data_relevant_pca oraz clean_data_relevant do trenowania modelu w następnej części projektu.

model-podstawowy-benchmark

June 9, 2024

[1]:
```
import pandas as pd
cleaned_data = pd.read_excel('clean_data_relevant.xlsx')
cleaned_data = cleaned_data[['BROKER', 'LATITUDE', 'LONGITUDE', 'TYPE',␣
↪'BOROUGH','NEIGHBOURHOOD', 'BEDS', 'POSTCODE', 'BATH', 'PROPERTYSQFT',␣
↪'PRICE']]
```

[2]:
```
cleaned_data
```

[2]:
```
      BROKER  LATITUDE  LONGITUDE  TYPE  BOROUGH  NEIGHBOURHOOD      BEDS  \
0        277  0.633396   0.505918     2        1            264  0.166667
1        844  0.102276   0.103390     6        3            259  0.333333
2        217  0.633742   0.505683     2        1            264  0.250000
3        844  0.423098   0.534539     6        0            192  0.416667
4        277  0.750035   0.556240     2        1            264  0.166667
…        …       …         …    …       …             …        …
4501     217  0.665089   0.539708     0        1            263  0.083333
4502     591  0.563894   0.713574     0        2            196  0.083333
4503     277  0.596189   0.453402     0        1            264  0.083333
4504     288  0.588660   0.690689     2        2             91  0.166667
4505     653  0.192653   0.512337     0        0             62  0.083333

      POSTCODE  BATH  PROPERTYSQFT    PRICE
0           18   0.2      0.134948   315000
1           54   0.2      0.205882   260000
2           18   0.1      0.024798    69000
3          127   0.2      0.435294   690000
4           23   0.2      0.225399   899500
…          …    …          …        …
4501        40   0.1      0.225399   599000
4502       147   0.1      0.225399   245000
4503         9   0.1      0.225399  1275000
4504       146   0.1      0.049020   598125
4505       114   0.1      0.059977   349000

[4506 rows x 11 columns]
```

```
[3]: from sklearn.model_selection import train_test_split

X = cleaned_data.drop(columns='PRICE')
y = cleaned_data['PRICE']
```

```
[4]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

```
[5]: from sklearn.linear_model import LinearRegression, HuberRegressor

reg = LinearRegression().fit(X_train, y_train)

print('Współczynniki: ',reg.coef_)
print('Wyraz wolny', reg.intercept_)

y_pred = reg.predict(X_test)

from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error,␣
 ↪median_absolute_error
import numpy as np

print("\nR-squared:", r2_score(y_test, y_pred))
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Median Absolute Error:", median_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

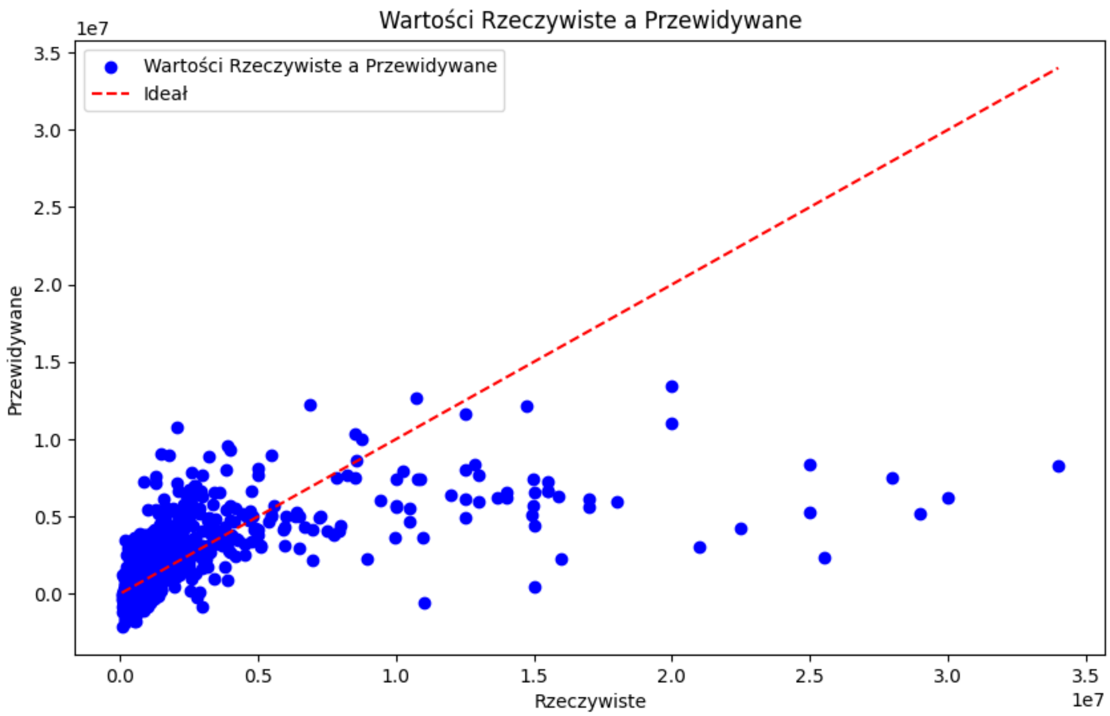
```
Współczynniki:  [ 4.82637713e+01  1.04212168e+06  4.80951990e+05 -5.98126037e+04
 -3.87799376e+05  2.53902786e+03 -2.46696958e+06 -1.27222931e+04
  8.99609911e+06  8.26434460e+06]
Wyraz wolny -625624.917808031

R-squared: 0.35106361164412225
Mean Absolute Error: 1336828.6355151676
Median Absolute Error: 774706.2717598878
Mean Squared Error: 6797303881835.929
Root Mean Squared Error: 2607163.953769676
```

Wartość współczynnika determinacji (R^2) wynosi około 0.35, co oznacza, że około 35% zmienności zmiennej PRICE jest wyjaśniona przez nasz model regresji liniowej. Idealna wartość tej miary wynosi 1, więc nasz model sprawuje się umiarkowanie w przewidywaniu nowych wartości.

Wysokie wartości średniego absolutnego błędu, mediany absolutnego błędu oraz błędu średniokwadratowego sugerują, że model ma ograniczoną jakość predykcyjną.

RMSE, czyli pierwiastek średniokwadratowy błędu, wynoszący około 2.6 miliona, oznacza że wartości przewidywane przez nasz model średnio odbiegają od rzeczywistych wartości o 2.6 miliona.

[6]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Wartości Rzeczywiste a␣
↪Przewidywane')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--',␣
↪color='red', label='Ideał')
plt.title('Wartości Rzeczywiste a Przewidywane')
plt.xlabel('Rzeczywiste')
plt.ylabel('Przewidywane')
plt.legend()
plt.show()
```

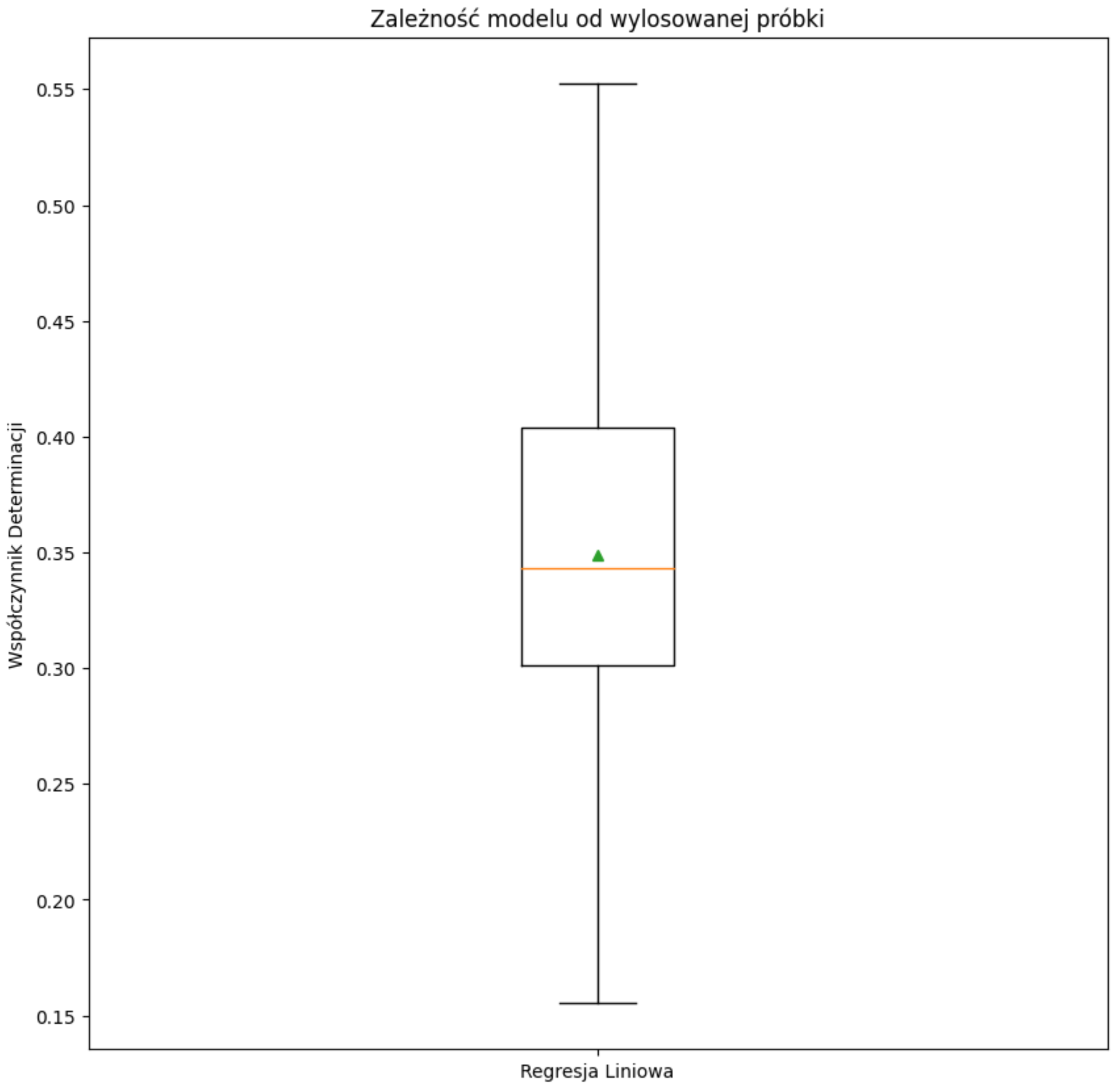


Z wykresu możemy wnioskować, że model średnio sobie radzi z przewidywaniem wyższych wartości. Może być to spowodowane niewystarczającą liczbą przypadków ogłoszeń z cenami >1.5 miliona lub wybraniem złego modelu.

[7]:
```
from sklearn.model_selection import cross_val_score, RepeatedKFold
from sklearn.pipeline import Pipeline
cv = RepeatedKFold(n_splits=10, n_repeats=10,random_state=1)
pipe_linear = Pipeline([
    ('linear', LinearRegression(fit_intercept=True))
])
```

```python
model_linear_scores = cross_val_score(pipe_linear, X, y, scoring='r2', cv = cv)
print("Cross Validation Score: ", model_linear_scores)
plt.figure(figsize=(10, 10))
plt.boxplot([model_linear_scores], labels=['Regresja Liniowa'], showmeans=True)
plt.title('Zależność modelu od wylosowanej próbki')
plt.ylabel('Współczynnik Determinacji')
plt.show()
```

```
Cross Validation Score:  [0.3048472  0.42354097 0.34307738 0.32622976 0.40298448
0.33415731
 0.35389578 0.43982456 0.34043789 0.23120915 0.48988149 0.24331674
 0.49782386 0.28851387 0.38692206 0.31823726 0.5076094  0.24137524
 0.20087974 0.3717199  0.30186297 0.36090288 0.36091703 0.47216805
 0.32848872 0.40213783 0.28580038 0.3227488  0.31614519 0.31932509
 0.27444135 0.34849079 0.42773341 0.55234023 0.23181206 0.34301315
 0.18369045 0.37432225 0.30588735 0.45629086 0.32173945 0.35631834
 0.33146502 0.30836542 0.48871109 0.33438085 0.32668307 0.21656633
 0.40950851 0.46846566 0.24628029 0.35716169 0.38562963 0.36004001
 0.31114419 0.44091716 0.52581639 0.24018291 0.21947161 0.36402104
 0.33655075 0.28235667 0.48104963 0.38214224 0.44749608 0.2721098
 0.3114152  0.38592847 0.32025077 0.27704416 0.29393787 0.37346854
 0.37172132 0.40549431 0.41534919 0.32788038 0.44813303 0.26441936
 0.20789418 0.31583986 0.18180707 0.37524692 0.28147825 0.4478128
 0.37382166 0.35915256 0.42621198 0.28800913 0.38681281 0.37627307
 0.40969144 0.29843979 0.4057372  0.46327359 0.33209954 0.30475667
 0.48181861 0.15528443 0.2295418  0.35885797]
```



Na podstawie wyników walidacji krzyżowej możemy powiedzieć, że model nie jest stabilny, ponieważ wyniki walidacji krzyżowej różnią się dosyć znacząco dla różnych podziałów danych. Przykładowo najwyższy wynik R^2 to 0.55 a najniższy około 0.15.

[8]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression

pipe_linear = Pipeline([
    ('linear', LinearRegression(fit_intercept=True))
])

param_grid = {
    'linear__fit_intercept': [True, False],
    'linear__n_jobs': [None, 1, 2, 4, 8],
```

```python
    'linear__positive': [False, True]

}

cv = RepeatedKFold(n_splits=10, n_repeats=10, random_state=1)

# Perform grid search
grid_search = GridSearchCV(pipe_linear, param_grid, cv=cv, scoring='r2')
grid_search.fit(X, y)

# Get the best parameters and best score
best_params = grid_search.best_params_
best_score = grid_search.best_score_

print("Best Parameters:", best_params)
print("Best Score:", best_score)
```

```
Best Parameters: {'linear__fit_intercept': True, 'linear__n_jobs': None,
'linear__positive': False}
Best Score: 0.3488448065360991
```

[9]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_x=X.copy()
vif_data = pd.DataFrame()
vif_data["feature"] = vif_x.columns

vif_data["VIF"] = [variance_inflation_factor(vif_x.values, i)
                          for i in range(len(vif_x.columns))]

print(vif_data)
```

```
        feature        VIF
0        BROKER   3.424444
1      LATITUDE  24.088323
2     LONGITUDE  46.314859
3          TYPE   3.725739
4       BOROUGH   3.215143
5  NEIGHBOURHOOD   4.328808
6          BEDS   9.242304
7      POSTCODE  13.026365
8          BATH   8.219750
9   PROPERTYSQFT   5.652389
```

Wysokie wartości VIF dla zmiennych BEDS, POSTCODE, BATH, PROPERTYSQFT, LATITUDE oraz LONGITUDE wskazują na ich silną korelację z innymi zmiennymi w zbiorze danych. Z uwagi na to, że wysokie wartości VIF są zazwyczaj niepożądane i mogą świadczyć o możliwej wieloliniowości modelu, rozważenie działań mających na celu zmniejszenie korelacji, np. przez zastosowanie analizy głównych składowych (PCA), może być wskazane.

[10]:
```
from sklearn.decomposition import PCA

n_comp = 6
col_names = ['feature_'+str(i) for i in range(0,n_comp)]
pca = PCA(n_components=n_comp)
data_relevant_pca = pd.DataFrame(pca.fit_transform(X), columns = col_names)

data_relevant_pca
```

[10]:
```
      feature_0   feature_1  feature_2  feature_3  feature_4  feature_5
0    -161.996266 -115.604465  26.933015  -1.502278  -0.524361  -0.060368
1     405.600518 -113.137165   1.529599   1.664794   1.029621   0.612129
2    -221.973645 -114.140947  26.179353  -1.457868  -0.480821  -0.040537
3     408.315389  -26.271092 -46.062553   0.790193  -1.800866  -0.051927
4    -161.897677 -113.999705  22.199013  -1.565854  -0.508710  -0.188432
…            …           …          …          …          …          …
4501 -221.522691 -106.138266   5.696170  -3.734768  -0.407511  -0.058016
4502  155.724427  -17.484736 -69.383974  -5.250187   0.523221   0.006469
4503 -162.175676 -118.498272  35.479340  -3.388526  -0.508557   0.015195
4504 -145.177221   89.001470 -38.528559  -2.844816   0.720788   0.006368
4505  219.603802   97.280650   5.694335  -4.703789  -1.571506   0.261213

[4506 rows x 6 columns]
```

[11]:
```
X_train, X_test, y_train, y_test = train_test_split(data_relevant_pca, y,␣
↪test_size=0.3, random_state=42)
```

[12]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_x=data_relevant_pca.copy()

vif_data = pd.DataFrame()
vif_data["feature"] = vif_x.columns

vif_data["VIF"] = [variance_inflation_factor(vif_x.values, i)
                          for i in range(len(vif_x.columns))]

print(vif_data)
```

```
     feature  VIF
0  feature_0  1.0
1  feature_1  1.0
2  feature_2  1.0
3  feature_3  1.0
4  feature_4  1.0
5  feature_5  1.0
```

Wartości VIF wynoszące 1.0 dla wszystkich zmiennych wskazują na brak korelacji między nimi. Taka sytuacja jest korzystne dla stabilności wybranego modelu.

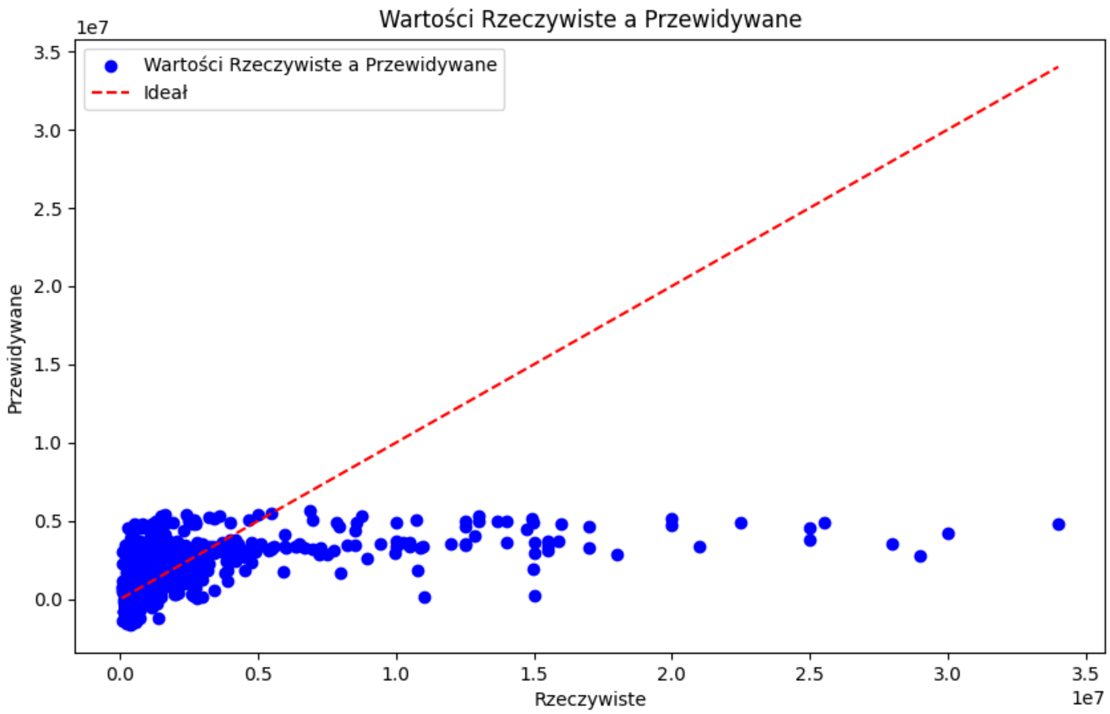
[13]:
```
reg_pca = LinearRegression().fit(X_train, y_train)

print('Współczynniki modelu: ',reg_pca.coef_)
print('Wyraz wolny: ', reg_pca.intercept_)

y_pred = reg_pca.predict(X_test)

print("\nR-squared:", r2_score(y_test, y_pred))
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Median Absolute Error:", median_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Współczynniki modelu:  [-8.00408231e+02 -8.48885438e+03  1.45686807e+04
1.32674542e+05
 -4.64438459e+05 -2.75789608e+06]
Wyraz wolny:  1710548.1834018864

R-squared: 0.1867849813104121
Mean Absolute Error: 1472007.5710042594
Median Absolute Error: 833256.5642988225
Mean Squared Error: 8518045377776.891
Root Mean Squared Error: 2918569.06338995
```

[13]:
```

```

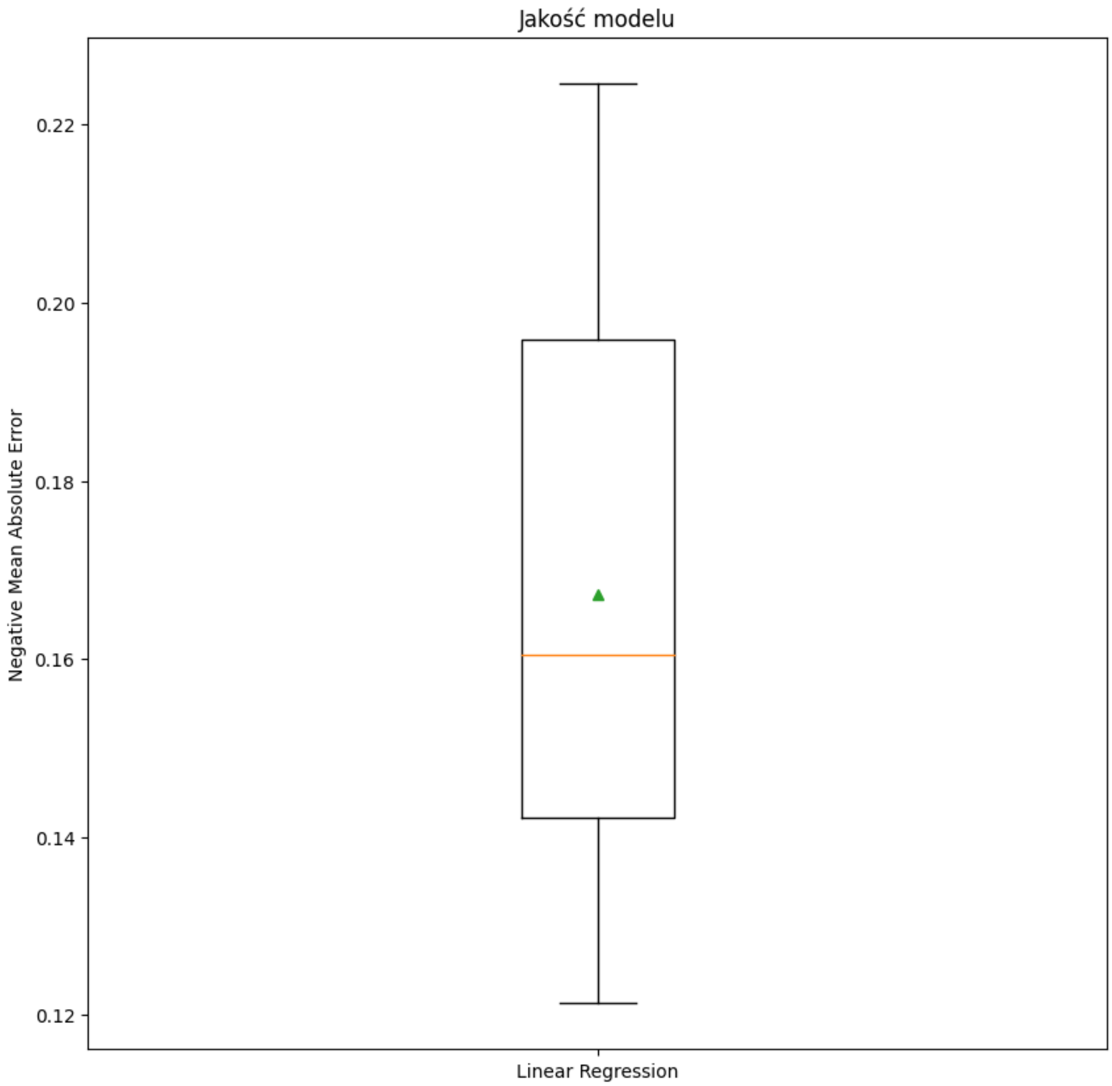
Model wytrenowany na danych po transformacji PCA wydaje się mieć ograniczoną zdolność do wyjaśniania zmienności cechy PRICE na podstawie dostarczonego zbioru - wynik współczynika determinacji wynosi około 0.15. Ponadto, wysokie wartości błędów wskazują na znaczną rozbieżność między przewidywanymi a rzeczywistymi wartościami zmiennej PRICE.

[14]:
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', label='Wartości Rzeczywiste a␣
↪Przewidywane')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--',␣
↪color='red', label='Ideał')
plt.title('Wartości Rzeczywiste a Przewidywane')
plt.xlabel('Rzeczywiste')
plt.ylabel('Przewidywane')
plt.legend()
plt.show()
```



[15]:
```python
cv = RepeatedKFold(n_splits=5, n_repeats=5,random_state=1)
pipe_linear = Pipeline([
    ('linear', LinearRegression(fit_intercept=True))
])

model_linear_scores = cross_val_score(pipe_linear, data_relevant_pca, y,␣
 ↪scoring='r2', cv = cv)
print("Cross Validation Score: ", model_linear_scores)
plt.figure(figsize=(10, 10))
plt.boxplot([model_linear_scores], labels=['Linear Regression'], showmeans=True)
plt.title('Jakość modelu')
plt.ylabel('Negative Mean Absolute Error')
plt.show()
```

```
Cross Validation Score:  [0.22456969 0.12135091 0.16043687 0.2207802  0.1433581
0.17514762
 0.19745571 0.16164154 0.15713593 0.13710569 0.16294432 0.19592756
 0.21469362 0.13717919 0.12834579 0.14219564 0.20406104 0.14507362
 0.13142583 0.17862491 0.18951502 0.14731603 0.15102189 0.13690514
 0.21530505]
```


Jakość modelu
0.22
0.20
0.18
0.16
0.14
0.12
Negative Mean Absolute Error
Linear Regression

dwa-dodatkowe-modele

June 9, 2024

1 Regresja

[20]:
```
from matplotlib import pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn import metrics
cleaned_data = pd.read_excel('clean_data_relevant.xlsx')
cleaned_data = cleaned_data[['BROKER', 'LATITUDE', 'LONGITUDE', 'TYPE',␣
↪'BOROUGH','NEIGHBOURHOOD', 'BEDS', 'POSTCODE', 'BATH', 'PROPERTYSQFT',␣
↪'PRICE']]
```

[21]:
```
from sklearn.model_selection import train_test_split

X = cleaned_data.drop(columns='PRICE')
y = cleaned_data['PRICE']
#y = np.round(y, decimals=-2) #przy cenach posiadłości część dzisiętna nie ma␣
↪aż tak dużo znaczenia znaczenia. Zakładając że ostatnie 2 miejsca wynoszą 99␣
↪to nawet dla minimalnej wartości w zbiorze danych stonowi to jedynie 0.2%
```

[22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
trained_models = []
```

1.1 Modele

1.1.1 Linear Model - model podstawowy

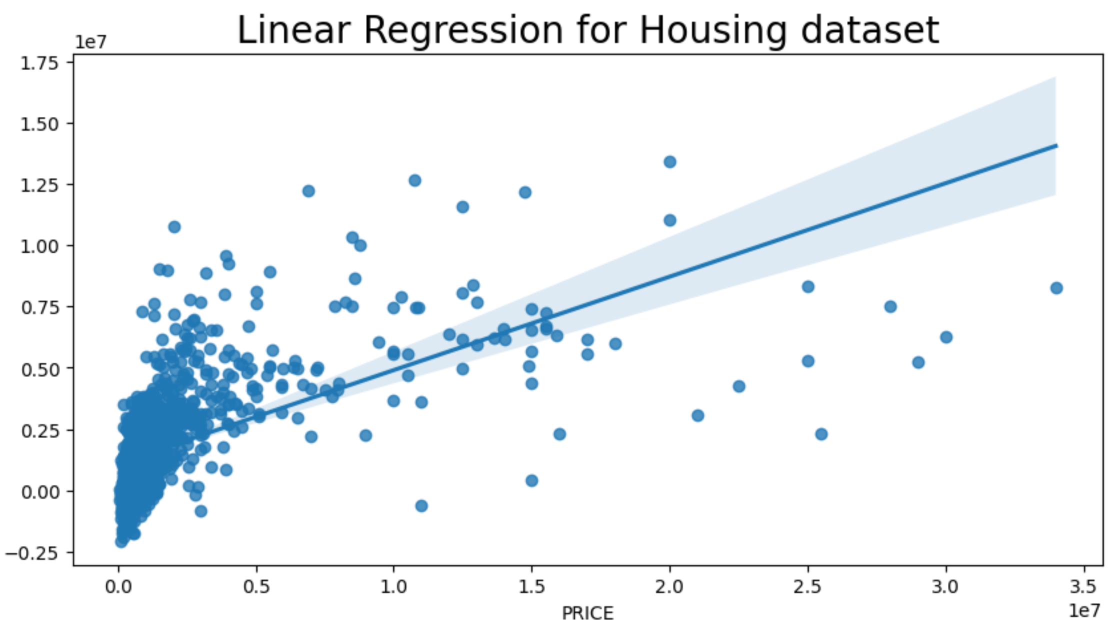
[23]:
```
from sklearn.linear_model import LinearRegression, HuberRegressor

reg = LinearRegression().fit(X_train, y_train)

print('Współczynniki: ',reg.coef_)
print('Wyraz wolny', reg.intercept_)

y_pred = reg.predict(X_test)
```

```
cv_score = cross_val_score(estimator = reg, X = X_train, y = y_train, cv = 20)

RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Linear Model', round(RMSE,2) , round(R2,4),␣
 ↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
 ↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
Współczynniki:  [ 4.82637713e+01  1.04212168e+06  4.80951990e+05 -5.98126037e+04
 -3.87799376e+05  2.53902786e+03 -2.46696958e+06 -1.27222931e+04
  8.99609911e+06  8.26434460e+06]
Wyraz wolny -625624.917808031
RMSE: 2607163.9538
R2: 0.3511
Cross Validated R2:  [0.3458, 0.1537, 0.5281, 0.4193, 0.3508, 0.2672, 0.0612,
0.1882, 0.548, 0.2921, 0.1532, 0.4424, 0.0416, 0.2398, 0.4881, 0.5676, 0.4707,
0.5462, -0.0848, 0.2286]
Mean Cross Validated R2:  0.3124
Min Cross Validated R2:  -0.0848
Max Cross Validated R2:  0.5676
```

[24]:

```
plt.figure(figsize = (10,5))
sns.regplot(x=y_test,y=y_pred)
plt.title('Linear Regression for Housing dataset', fontsize = 20)
plt.show()
```



[25]:
```
from sklearn.linear_model import LinearRegression, HuberRegressor

reg = HuberRegressor().fit(X_train, y_train)

print('Współczynniki: ',reg.coef_)
print('Wyraz wolny', reg.intercept_)

y_pred = reg.predict(X_test)


cv_score = cross_val_score(estimator = reg, X = X_train, y = y_train, cv = 20)

RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Huber Model', round(RMSE,2) , round(R2,4),␣
↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
Współczynniki:  [-4.78448114e+01  5.52961796e+05  3.35432760e+05  6.80072102e+04
 -2.09702782e+05 -1.03320073e+02  3.74172633e+05 -5.37332761e+03
  5.26283624e+05  4.50102613e+05]
Wyraz wolny 765420.4355215558
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

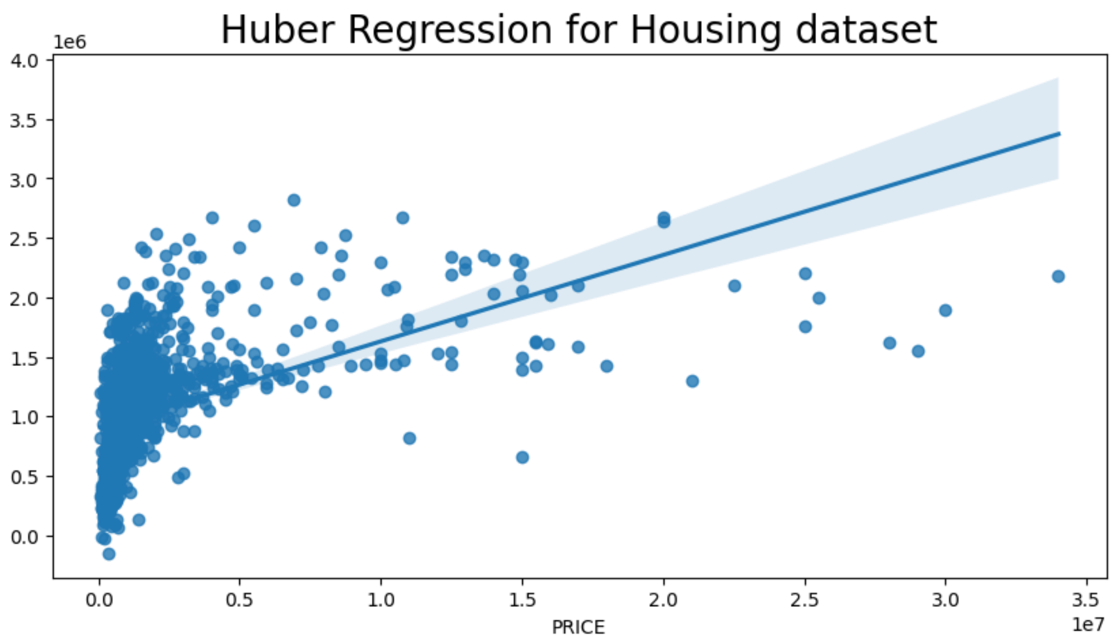
```
Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
```

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

```
RMSE: 3110867.2428
R2: 0.0761
Cross Validated R2:  [0.0882, 0.2082, 0.1447, 0.0718, 0.0313, 0.0541, 0.2153,
0.1253, 0.04, 0.0194, 0.0095, 0.1026, 0.1088, 0.0383, 0.0795, 0.1062, 0.0581,
0.1332, 0.2582, 0.0244]
Mean Cross Validated R2:  0.0959
Min Cross Validated R2:  0.0095
Max Cross Validated R2:  0.2582

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_huber.py:342:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

[26]:

```
plt.figure(figsize = (10,5))
sns.regplot(x=y_test,y=y_pred)
plt.title('Huber Regression for Housing dataset', fontsize = 20)
plt.show()
```



1.1.2 Forest Regressor

[27]:
```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators = 10, random_state = 0)
forest_reg.fit(X_train, y_train)
y_pred = forest_reg.predict(X_test)


cv_score = cross_val_score(estimator = forest_reg, X = X_train, y = y_train, cv␣
 ↪= 20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = forest_reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Forest Regressor Model = 10 trees', round(RMSE,2) ,␣
 ↪round(R2,4), round(cv_score.mean(),4), round(cv_score.min(),4),␣
 ↪round(cv_score.max(),4), list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 1940629.4766
R2: 0.6405
Cross Validated R2:  [0.7517, -0.0809, 0.692, 0.5232, 0.7029, 0.1126, 0.1084,
0.2822, 0.8527, 0.4304, 0.2866, 0.2914, 0.2893, 0.6414, 0.6216, 0.7545, 0.7896,
0.5274, 0.0793, 0.4204]
Mean Cross Validated R2:  0.4538
Min Cross Validated R2:  -0.0809
Max Cross Validated R2:  0.8527
```

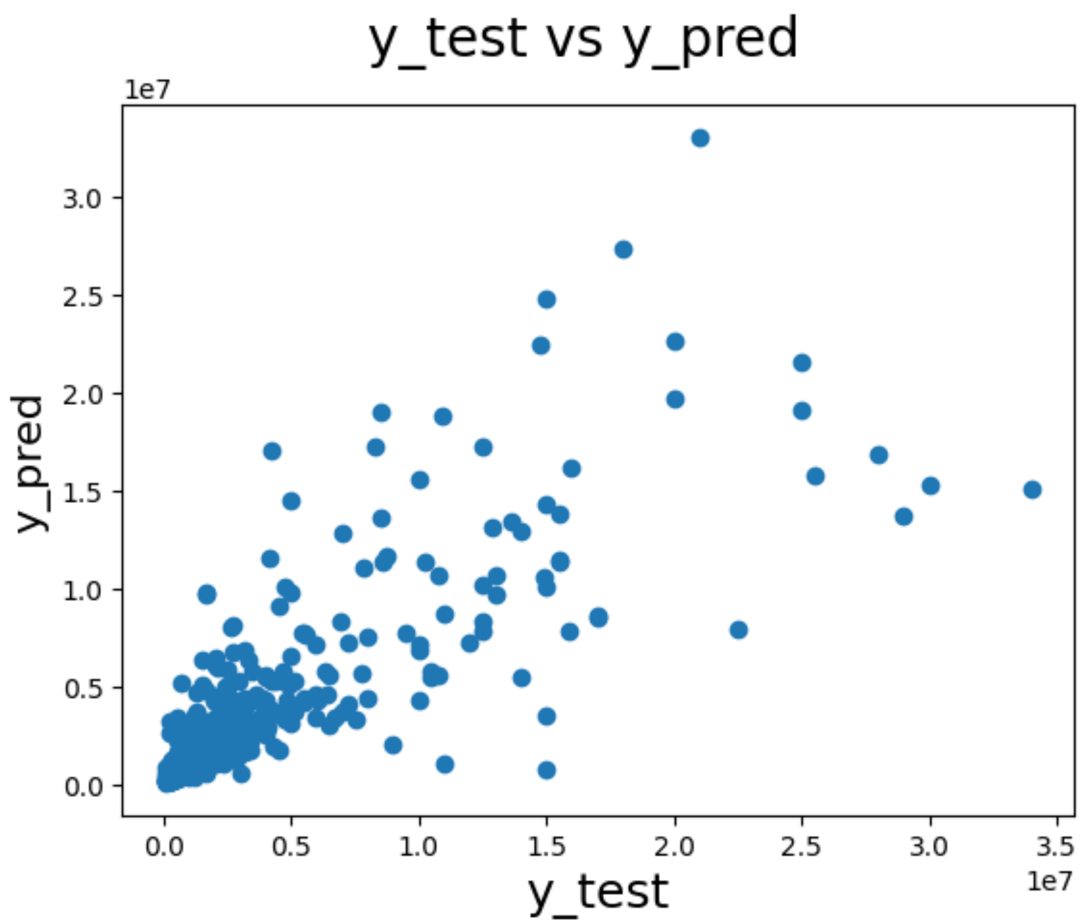
[28]:
```
from sklearn.ensemble import RandomForestRegressor

forest_reg_30 = RandomForestRegressor(n_estimators = 30, random_state = 0)
forest_reg_30.fit(X_train, y_train)
y_pred = forest_reg_30.predict(X_test)


cv_score = cross_val_score(estimator = forest_reg_30, X = X_train, y = y_train,␣
 ↪cv = 20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = forest_reg_30.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
```

```
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Forest Regressor Model = 30 trees', round(RMSE,2) ,␣
↪round(R2,4), round(cv_score.mean(),4), round(cv_score.min(),4),␣
↪round(cv_score.max(),4), list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 1795072.1975
R2: 0.6924
Cross Validated R2:  [0.7542, 0.1516, 0.7353, 0.445, 0.6769, 0.3038, 0.3995,
0.3781, 0.8728, 0.4241, 0.3019, 0.3444, 0.2986, 0.5409, 0.7355, 0.7796, 0.8565,
0.6355, 0.5446, 0.4774]
Mean Cross Validated R2:  0.5328
Min Cross Validated R2:  0.1516
Max Cross Validated R2:  0.8728
```

[29]:

```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)              # Plot heading
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)                          # Y-label
plt.show()
```



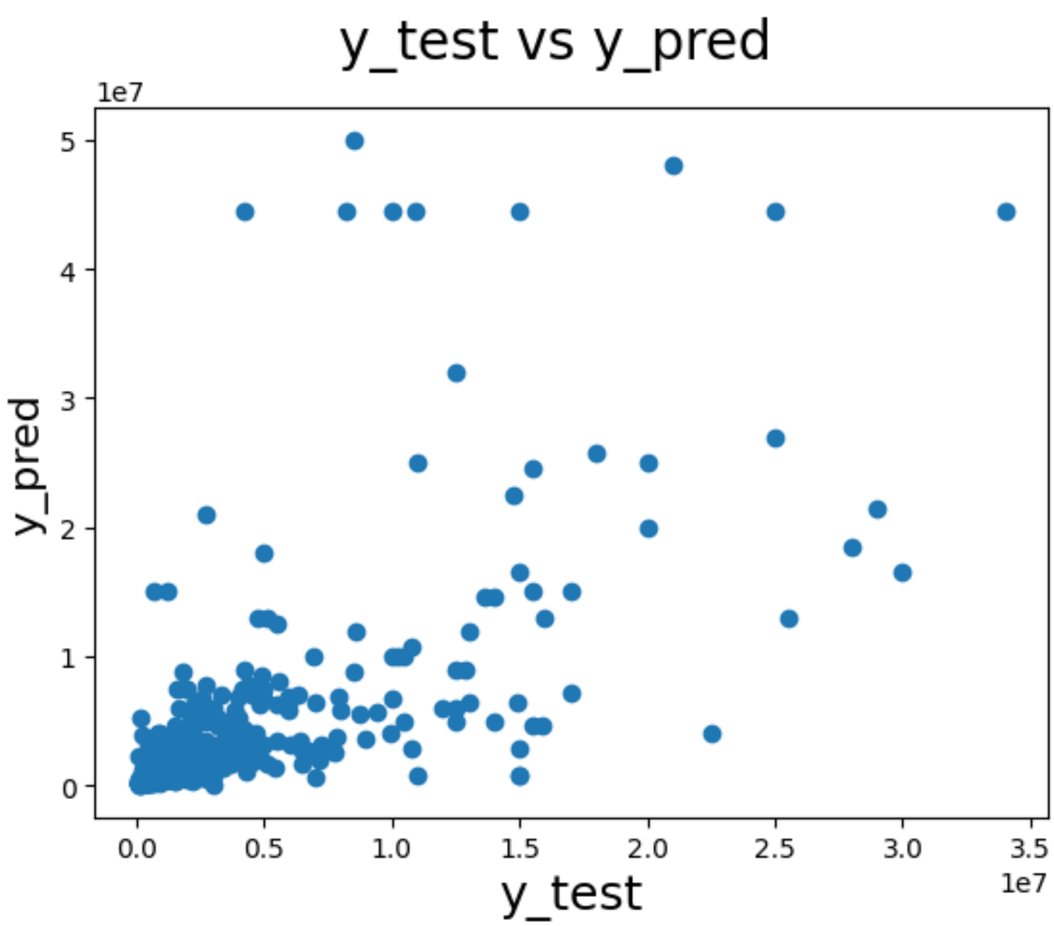
1.1.3 Decision Tree

[30]:
```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)
y_pred = tree_reg.predict(X_test)


cv_score = cross_val_score(estimator = tree_reg, X = X_train, y = y_train, cv =␣
  ↪20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = tree_reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
```

```
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Decision Tree', round(RMSE,2) , round(R2,4),␣
 ↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
 ↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 3271182.2776
R2: -0.0216
Cross Validated R2:  [0.5702, -2.84, -0.445, 0.4527, -0.24, 0.1602, 0.0184,
0.2744, 0.6666, -0.0912, 0.0288, 0.2898, -0.1282, 0.4911, 0.6577, 0.312, 0.2001,
0.2164, -0.8715, 0.2628]
Mean Cross Validated R2:  -0.0007
Min Cross Validated R2:  -2.84
Max Cross Validated R2:  0.6666
```

[31]:
```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_pred', fontsize=16)
plt.show()
```

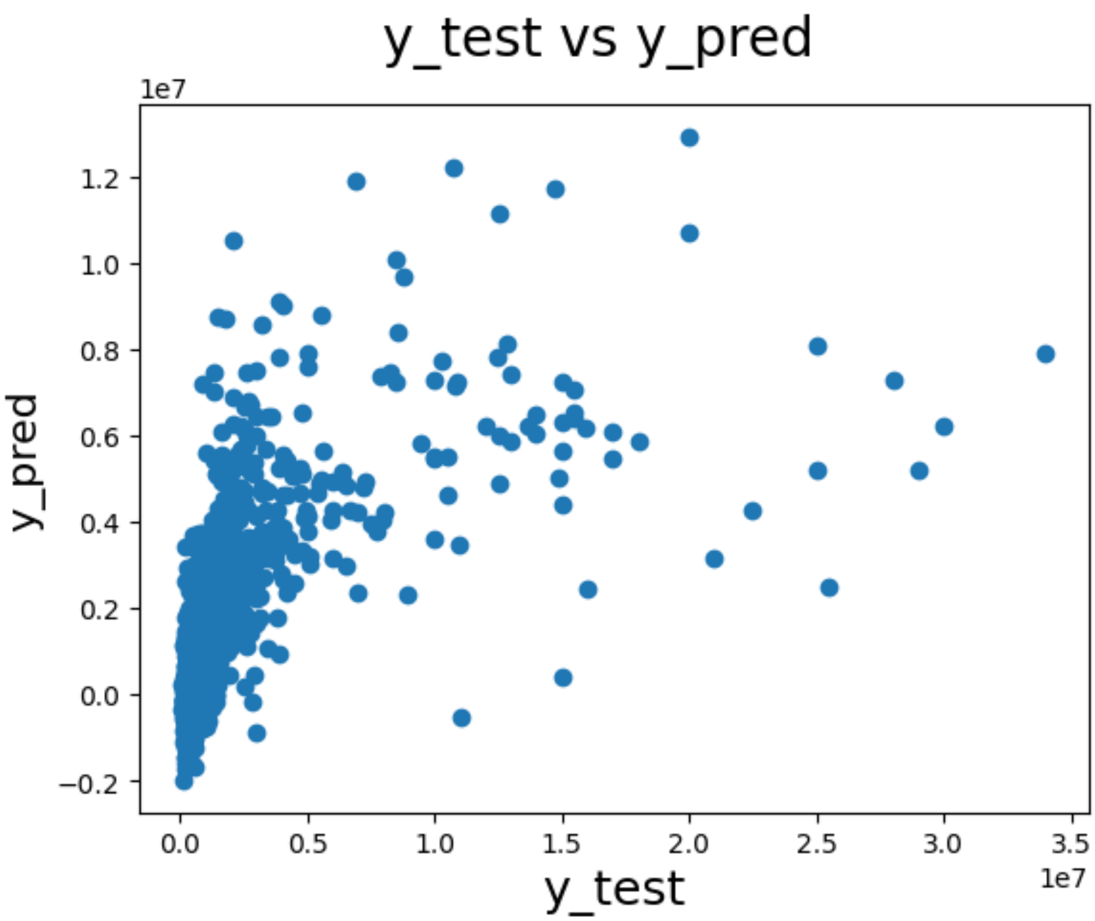


1.1.4 Ridge Regression

[32]:
```
from sklearn.linear_model import Ridge

ridge_reg = Ridge(alpha=3, solver="cholesky")
ridge_reg.fit(X_train, y_train)
y_pred = ridge_reg.predict(X_test)


cv_score = cross_val_score(estimator = ridge_reg, X = X_train, y = y_train, cv␣
↪= 20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = ridge_reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Ridge Model', round(RMSE,2) , round(R2,4),␣
↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 2604068.0257
R2: 0.3526
Cross Validated R2:  [0.3422, 0.1821, 0.5286, 0.4153, 0.3451, 0.2662, 0.0868,
0.1965, 0.5346, 0.2871, 0.1596, 0.4547, 0.0792, 0.2423, 0.4827, 0.5581, 0.4564,
0.5475, 0.0004, 0.2287]
Mean Cross Validated R2:  0.3197
Min Cross Validated R2:  0.0004
Max Cross Validated R2:  0.5581
```

[33]:
```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_pred', fontsize=16)
plt.show()
```



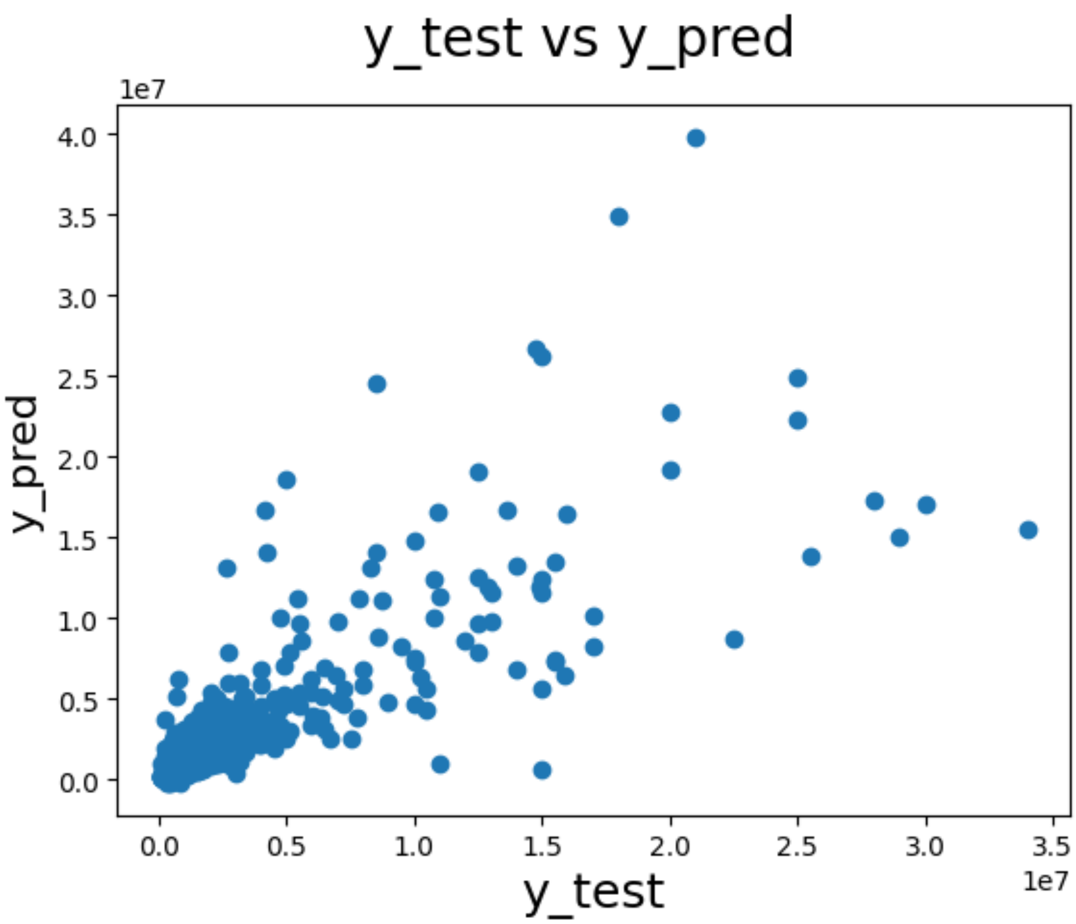
1.1.5 XGBoost

[34]:
```
from xgboost import XGBRegressor

XGBR = XGBRegressor(n_estimators=1000, max_depth=7, eta=0.1, subsample=0.8,␣
 ↪colsample_bytree=0.8)
XGBR.fit(X_train, y_train)
y_pred = XGBR.predict(X_test)


cv_score = cross_val_score(estimator = XGBR, X = X_train, y = y_train, cv = 20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = XGBR.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
```

```
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['XGBRegressor', round(RMSE,2) , round(R2,4),␣
 ↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
 ↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 1879060.5311
R2: 0.6629
Cross Validated R2:  [0.7585, -0.5486, 0.5267, 0.664, 0.6568, 0.4718, 0.4094,
0.414, 0.8921, 0.6182, 0.7159, -0.2433, 0.3732, 0.6535, 0.796, 0.7811, 0.8598,
0.7116, 0.5855, 0.3537]
Mean Cross Validated R2:  0.5225
Min Cross Validated R2:  -0.5486
Max Cross Validated R2:  0.8921
```

[35]:
```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('y_test vs y_pred', fontsize=20)
plt.xlabel('y_test', fontsize=18)
plt.ylabel('y_pred', fontsize=16)
plt.show()
```



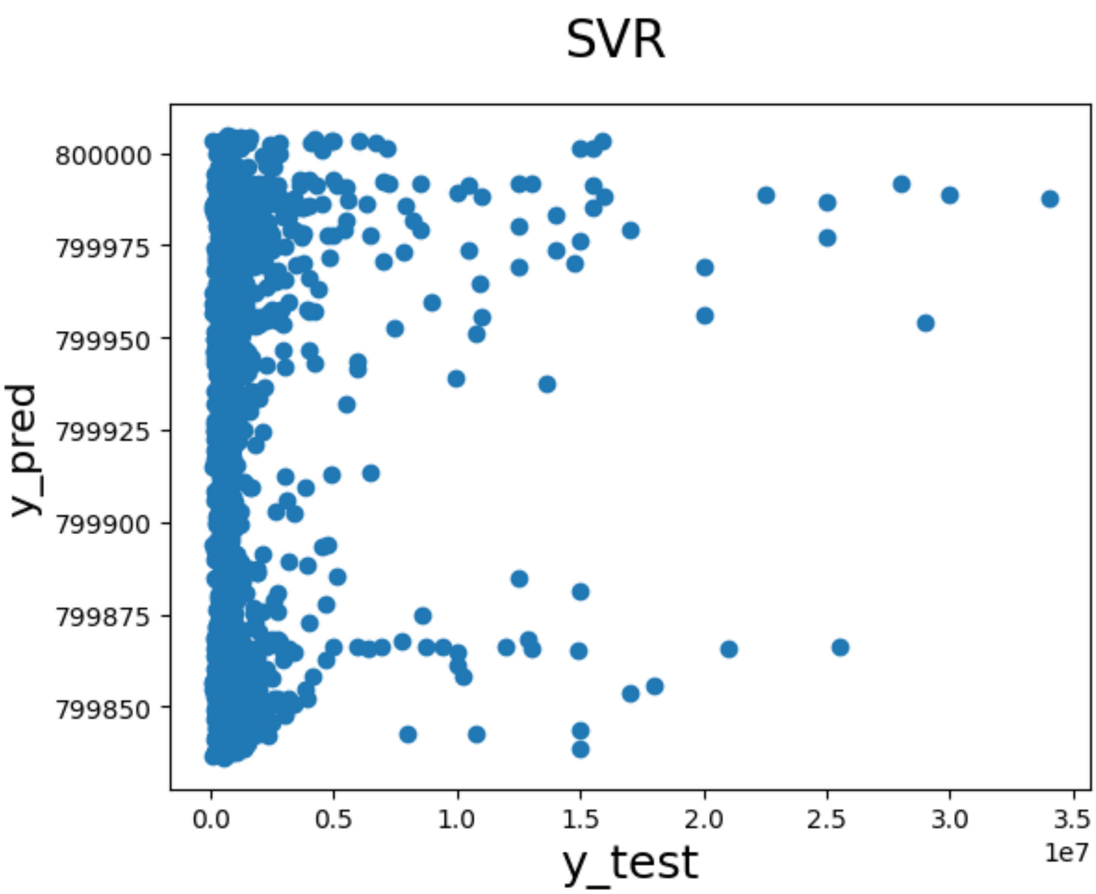
1.1.6 SVR

[36]:
```
from sklearn.svm import SVR
svr_clf = SVR(kernel = 'rbf')

svr_clf.fit(X_train, y_train)
y_pred = svr_clf.predict(X_test)

cv_score = cross_val_score(estimator = svr_clf, X = X_train, y = y_train, cv =
 ↪20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = svr_clf.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
```

```
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['SVR', round(RMSE,2) , round(R2,4), round(cv_score.
 ↪mean(),4), round(cv_score.min(),4), round(cv_score.max(),4), list(map(lambda␣
 ↪x: round(x,4) ,cv_score)) ])
```

```
RMSE: 3366019.8979
R2: -0.0817
Cross Validated R2:  [-0.056, -0.1082, -0.1063, -0.0855, -0.0628, -0.0479,
-0.0372, -0.0459, -0.0884, -0.0641, -0.0652, -0.134, -0.1696, -0.0637, -0.0861,
-0.0799, -0.0733, -0.0934, -0.0609, -0.0494]
Mean Cross Validated R2:  -0.0789
Min Cross Validated R2:  -0.1696
Max Cross Validated R2:  -0.0372
```

[37]:
```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('SVR', fontsize=20)                # Plot heading
plt.xlabel('y_test', fontsize=18)                           # X-label
plt.ylabel('y_pred', fontsize=16)                           # Y-label
plt.show()
```

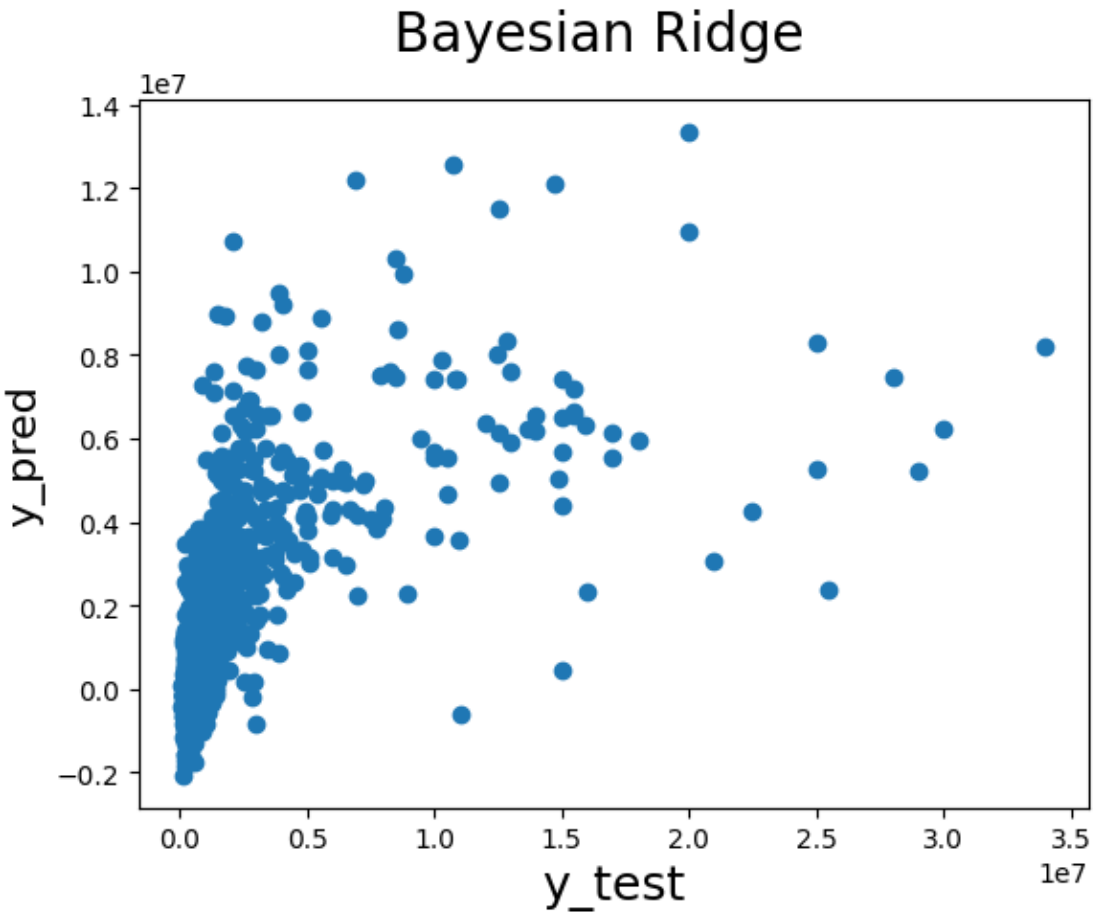


1.1.7 Bayesian regression

[38]:
```
from sklearn.linear_model import BayesianRidge
bayesian_reg = BayesianRidge()
bayesian_reg.fit(X_train, y_train)
y_pred = bayesian_reg.predict(X_test)


cv_score = cross_val_score(estimator = bayesian_reg, X = X_train, y = y_train,␣
 ↪cv = 20)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
R2 = bayesian_reg.score(X_test, y_test)

print('RMSE:', round(RMSE,4))
print('R2:', round(R2,4))
print("Cross Validated R2: ", list(map(lambda x: round(x,4) ,cv_score)))
print("Mean Cross Validated R2: ", round(cv_score.mean(),4) )
print("Min Cross Validated R2: ", round(cv_score.min(),4) )
print("Max Cross Validated R2: ", round(cv_score.max(),4) )
trained_models.append(['Bayesian Reg', round(RMSE,2) , round(R2,4),␣
 ↪round(cv_score.mean(),4), round(cv_score.min(),4), round(cv_score.max(),4),␣
 ↪list(map(lambda x: round(x,4) ,cv_score)) ])
```

```
RMSE: 2606303.0194
R2: 0.3515
Cross Validated R2:  [0.3452, 0.1593, 0.5284, 0.4186, 0.3498, 0.2671, 0.0668,
0.19, 0.5454, 0.2912, 0.1544, 0.4451, 0.0489, 0.2404, 0.4872, 0.5658, 0.4678,
0.5469, -0.0677, 0.2287]
Mean Cross Validated R2:  0.314
Min Cross Validated R2:  -0.0677
Max Cross Validated R2:  0.5658
```

[39]:
```
fig = plt.figure()
plt.scatter(y_test,y_pred)
fig.suptitle('Bayesian Ridge', fontsize=20)
plt.xlabel('y_test', fontsize=18)                          # X-label
plt.ylabel('y_pred', fontsize=16)                          # Y-label
plt.show()
```



1.2 Podsumowanie Regresji

[40]:
```
trained_models = pd.DataFrame( trained_models, columns=['Model','RMSE','R2␣
↪Score','Mean Cross Validated R2 Score','Min Cross Validated R2 Score','Max␣
↪Cross Validated R2 Score','Cross Validated R2 Scores'])
trained_models
```

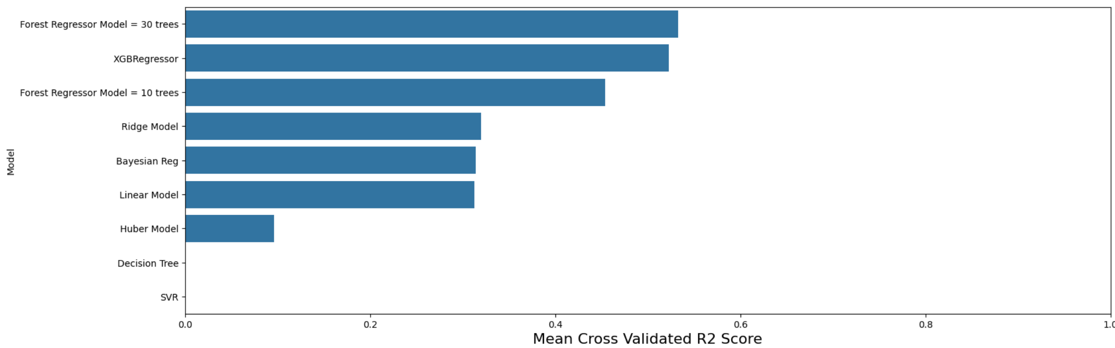
[40]:

	Model	RMSE	R2 Score	\
0	Linear Model	2607163.95	0.3511	
1	Huber Model	3110867.24	0.0761	
2	Forest Regressor Model = 10 trees	1940629.48	0.6405	
3	Forest Regressor Model = 30 trees	1795072.20	0.6924	
4	Decision Tree	3271182.28	-0.0216	
5	Ridge Model	2604068.03	0.3526	
6	XGBRegressor	1879060.53	0.6629	
7	SVR	3366019.90	-0.0817	
8	Bayesian Reg	2606303.02	0.3515	

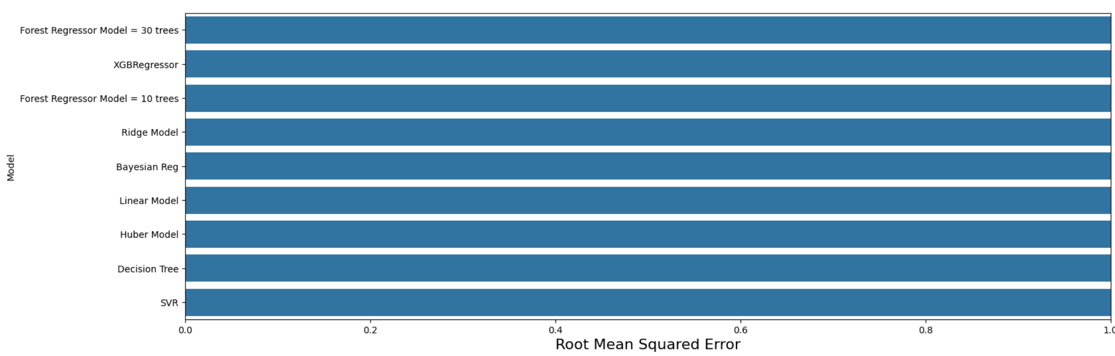
```
   Mean Cross Validated R2 Score  Min Cross Validated R2 Score  \
0                         0.3124                       -0.0848
1                         0.0959                        0.0095
2                         0.4538                       -0.0809
3                         0.5328                        0.1516
4                        -0.0007                       -2.8400
5                         0.3197                        0.0004
6                         0.5225                       -0.5486
7                        -0.0789                       -0.1696
8                         0.3140                       -0.0677

   Max Cross Validated R2 Score  \
0                        0.5676
1                        0.2582
2                        0.8527
3                        0.8728
4                        0.6666
5                        0.5581
6                        0.8921
7                       -0.0372
8                        0.5658

                        Cross Validated R2 Scores
0  [0.3458, 0.1537, 0.5281, 0.4193, 0.3508, 0.267…
1  [0.0882, 0.2082, 0.1447, 0.0718, 0.0313, 0.054…
2  [0.7517, -0.0809, 0.692, 0.5232, 0.7029, 0.112…
3  [0.7542, 0.1516, 0.7353, 0.445, 0.6769, 0.3038…
4  [0.5702, -2.84, -0.445, 0.4527, -0.24, 0.1602,…
5  [0.3422, 0.1821, 0.5286, 0.4153, 0.3451, 0.266…
6  [0.7585, -0.5486, 0.5267, 0.664, 0.6568, 0.471…
7  [-0.056, -0.1082, -0.1063, -0.0855, -0.0628, -…
8  [0.3452, 0.1593, 0.5284, 0.4186, 0.3498, 0.267…
```

[41]:

```python
f, axe = plt.subplots(1,1, figsize=(18,6))

trained_models.sort_values(by=['Mean Cross Validated R2 Score'],
 ↪ascending=False, inplace=True)

sns.barplot(x='Mean Cross Validated R2 Score', y='Model', data =
 ↪trained_models, ax = axe)
axe.set_xlabel('Mean Cross Validated R2 Score', size=16)
axe.set_ylabel('Model')
axe.set_xlim(0,1.0)

plt.show()
```



[42]:
```
f, axe = plt.subplots(1,1, figsize=(18,6))

trained_models.sort_values(by=['RMSE'], ascending=True, inplace=True)

sns.barplot(x='RMSE', y='Model', data = trained_models, ax = axe)
axe.set_xlabel('Root Mean Squared Error', size=16)
axe.set_ylabel('Model')
axe.set_xlim(0,1.0)

plt.show()
```



2 Klasyfikacja cen

Dzielimy zbiór danych na 5 przedziałów, aby móc przewidzieć etykiety zamiast wartości ciągłych.

[43]:
```
import pandas as pd
X = cleaned_data.drop(columns='PRICE')
y = cleaned_data['PRICE']
y = np.round(y, decimals=-3) #przy cenach posiadłości część nie ma aż tak dużo␣
↪znaczenia znaczenia. Zakładając że ostatnie 2 miejsca wynoszą 99 to nawet␣
↪dla minimalnej wartości w zbiorze danych stonowi to jedynie 0.2%
```

```
sorted_prices = y.sort_values()

categories = pd.qcut(sorted_prices, q=3, labels=[ 'Low',  'Medium', 'High'])

category_ranges = {}
for category in categories.cat.categories:
    min_val = sorted_prices[categories == category].min()
    max_val = sorted_prices[categories == category].max()
    category_ranges[category] = (min_val, max_val)

print("Wartości dla każdego przedziału:")
for category, (min_val, max_val) in category_ranges.items():
    print(f"{category}: {min_val} - {max_val}")
```

```
Wartości dla każdego przedziału:
Low: 50000 - 599000
Medium: 600000 - 1175000
High: 1180000 - 60000000
```

[44]:
```
y =  y.apply(lambda x: 0 if x < 599000 else (1 if x < 1100000 else (2 if x <
 ↪2500000 else 3)))

print(y.value_counts())
```

```
PRICE
0    1477
1    1439
2     999
3     591
Name: count, dtype: int64
```

[45]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 ↪random_state=42)
trained_models = []
```

2.1 Modele Klasyfikacji

2.1.1 DecisionTreeClassifier

[46]:
```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
clf_gini = DecisionTreeClassifier(criterion='gini',  random_state=0)
clf_gini.fit(X_train, y_train)
y_pred_gini = clf_gini.predict(X_test)

conf_matrix = confusion_matrix(y_test, y_pred_gini)
```

```
class_report = classification_report(y_test, y_pred_gini)

print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
Confusion Matrix:
[[325  99   9   3]
 [ 81 266  92   5]
 [ 24  87 151  32]
 [  2   7  36 133]]

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.75      0.75       436
           1       0.58      0.60      0.59       444
           2       0.52      0.51      0.52       294
           3       0.77      0.75      0.76       178

    accuracy                           0.65      1352
   macro avg       0.66      0.65      0.65      1352
weighted avg       0.65      0.65      0.65      1352
```

2.1.2 Forest Classifier

[47]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, classification_report

# Instantiate the Random Forest Classifier
clf_forest = RandomForestClassifier(random_state=0)
clf_forest.fit(X_train, y_train)
y_pred_forest = clf_forest.predict(X_test)
conf_matrix_forest = confusion_matrix(y_test, y_pred_forest)
class_report_forest = classification_report(y_test, y_pred_forest)

print("Confusion Matrix (Random Forest Classifier):")
print(conf_matrix_forest)
print("\nClassification Report (Random Forest Classifier):")
print(class_report_forest)
```

```
Confusion Matrix (Random Forest Classifier):
[[341  89   4   2]
 [ 67 317  56   4]
 [ 17  68 179  30]
 [  2   2  35 139]]
```

```
Classification Report (Random Forest Classifier):
              precision    recall  f1-score   support

           0       0.80      0.78      0.79       436
           1       0.67      0.71      0.69       444
           2       0.65      0.61      0.63       294
           3       0.79      0.78      0.79       178

    accuracy                           0.72      1352
   macro avg       0.73      0.72      0.72      1352
weighted avg       0.72      0.72      0.72      1352
```

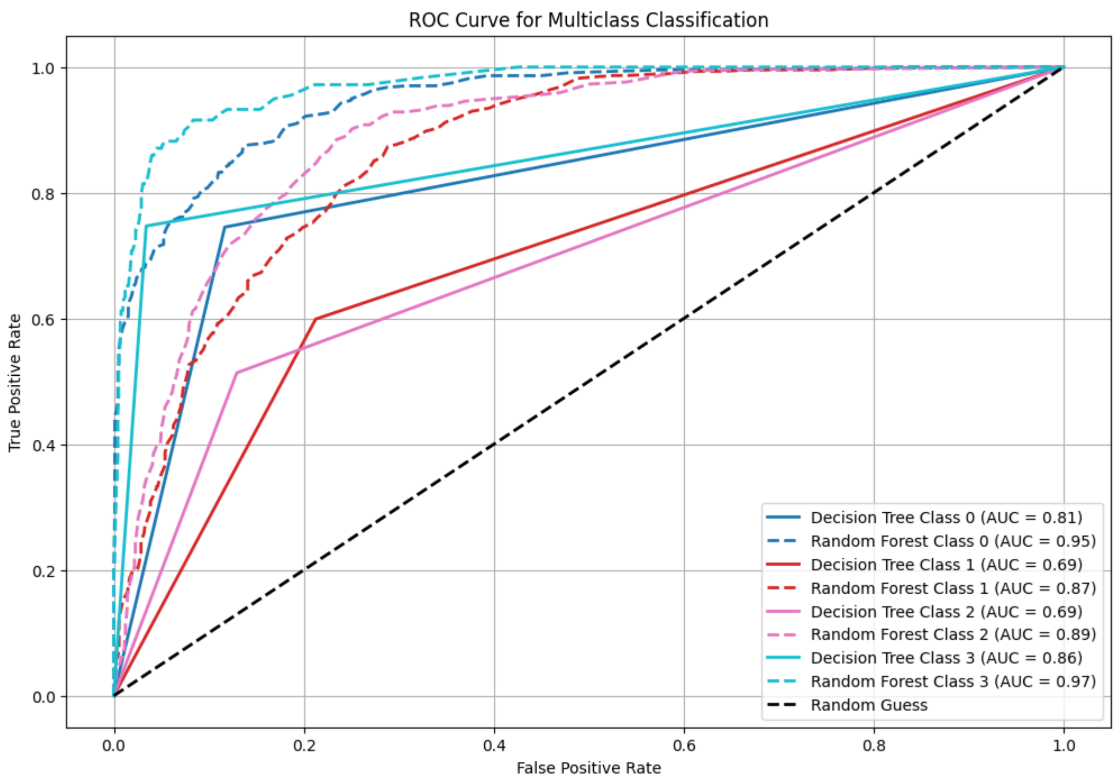
2.2 Podsumowanie Klasyfikacji

Ponieważ roc_curve i roc_auc_score nie obsługują oceny modeli do przewidywania wielu etykiet, postanowiłyśmy porównać modele pod względem jakości przewidywania dla każdej klasy osobno.

[48]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.preprocessing import label_binarize


y_test_binarized = label_binarize(y_test, classes=np.unique(y_test))
n_classes = y_test_binarized.shape[1]

clf_gini = DecisionTreeClassifier(criterion='gini', random_state=0)
clf_gini.fit(X_train, y_train)
y_pred_gini_proba = clf_gini.predict_proba(X_test)

clf_forest = RandomForestClassifier(random_state=0)
clf_forest.fit(X_train, y_train)
y_pred_forest_proba = clf_forest.predict_proba(X_test)

fpr_gini = dict()
tpr_gini = dict()
roc_auc_gini = dict()
fpr_forest = dict()
tpr_forest = dict()
roc_auc_forest = dict()

for i in range(n_classes):
    fpr_gini[i], tpr_gini[i], _ = roc_curve(y_test_binarized[:, i],␣
 ↪y_pred_gini_proba[:, i])
    roc_auc_gini[i] = roc_auc_score(y_test_binarized[:, i], y_pred_gini_proba[:
 ↪, i])
```

```
    fpr_forest[i], tpr_forest[i], _ = roc_curve(y_test_binarized[:, i],␣
↪y_pred_forest_proba[:, i])
    roc_auc_forest[i] = roc_auc_score(y_test_binarized[:, i],␣
↪y_pred_forest_proba[:, i])

plt.figure(figsize=(12, 8))
colors = plt.cm.get_cmap('tab10', n_classes)

for i in range(n_classes):
    plt.plot(fpr_gini[i], tpr_gini[i], color=colors(i), lw=2,
             label=f'Decision Tree Class {i} (AUC = {roc_auc_gini[i]:.2f})')
    plt.plot(fpr_forest[i], tpr_forest[i], color=colors(i), linestyle='--',␣
↪lw=2,
             label=f'Random Forest Class {i} (AUC = {roc_auc_forest[i]:.2f})')

plt.plot([0, 1], [0, 1], 'k--', lw=2, label='Random Guess')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Multiclass Classification')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

```
<ipython-input-48-2d643a2bc3e9>:31: MatplotlibDeprecationWarning: The get_cmap
function was deprecated in Matplotlib 3.7 and will be removed two minor releases
later. Use ``matplotlib.colormaps[name]`` or
``matplotlib.colormaps.get_cmap(obj)`` instead.
  colors = plt.cm.get_cmap('tab10', n_classes)
```



Z wykresu widać, że Forest Classifier jest zdecydowanie lepszym modelem dla przewidywania każdej z etykiet.

3 Wnioski

Ze względu na dobre dopasowanie, wybrałyśmy XGBRegressor oraz Forest Regressor Model (30 drzew) do dalszego etapu projektu. Spośród wielu wytrenowanych modeli, te dwa charakteryzowały się najlepszym dopasowaniem pod względem uśrednionej wartości współczynnika determinacji. Jeśli wyniki po przeprowadzeniu hiperparametryzacji będą niezadowalające, spróbujemy dopasować model klasyfikacji Forest Classifier.

hiperparametryzacja

June 9, 2024

1 Regresja

[ ]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
```

[ ]:
```
cleaned_data = pd.read_excel('clean_data_relevant.xlsx')
```

[ ]:
```
cleaned_data = cleaned_data[['BROKER', 'LATITUDE', 'LONGITUDE', 'TYPE',␣
↪'BOROUGH','NEIGHBOURHOOD', 'BEDS', 'POSTCODE', 'BATH', 'PROPERTYSQFT',␣
↪'PRICE']]
```

[ ]:
```
from sklearn.model_selection import train_test_split

X = cleaned_data.drop(columns='PRICE')
y = cleaned_data['PRICE']
```

[ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)
trained_models = []
```

1.1 Modele

[ ]:
```
linear = LinearRegression()

param_grid = {
    'fit_intercept': [True, False],
    'copy_X': [True, False],
    'n_jobs': [3, 4, 10,15,20,50,80],
    'positive': [True, False]
}
```

```
grid_search = GridSearchCV(estimator=linear, param_grid=param_grid, cv=5,␣
↪scoring='neg_mean_squared_error',  verbose=2)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model = grid_search.best_estimator_
y_pred = best_xgb_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Fitting 5 folds for each of 56 candidates, totalling 280 fits
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=False; total time=
```

```
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=10, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=15, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=True; total time=
```

```
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=20, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=50, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=False; total time=
```

```
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=True, n_jobs=80, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=False; total time=
0.1s
[CV] END copy_X=True, fit_intercept=False, n_jobs=4, positive=False; total time=
```

```
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.1s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=True; total time=
0.1s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=True; total time=
```

```
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.1s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=False; total
```

```
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=True, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=3, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=4, positive=False; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=True; total time=
```

```
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=False; total
```

```
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=True, n_jobs=80, positive=False; total
```

```
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=3, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=True; total time=
0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=4, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=True; total
```

```
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=10, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=15, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=False; total
```

```
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=20, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=50, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=True; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
[CV] END copy_X=False, fit_intercept=False, n_jobs=80, positive=False; total
time=   0.0s
Best parameters found:  {'copy_X': True, 'fit_intercept': True, 'n_jobs': 3,
'positive': False}
R-squared:  0.35106361164412225
```

```
[ ]: xgb_model = XGBRegressor()

     param_grid = {
         'n_estimators': [25, 50, 75]
     }

     grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
      ↪scoring='neg_mean_squared_error')
     grid_search.fit(X_train, y_train)

     best_params = grid_search.best_params_
     print("Best parameters found: ", best_params)

     best_xgb_model = grid_search.best_estimator_
     y_pred = best_xgb_model.predict(X_test)
     r2 = r2_score(y_test, y_pred)
     print("R-squared: ", r2)
```

```
Best parameters found:  {'n_estimators': 50}
R-squared:  0.6301618111670517
```

```
[ ]: xgb_model = XGBRegressor()

     param_grid = {
         'n_estimators': [25, 50, 75],
         'max_depth': [ 3, 5, 10]
     }

     grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
      ↪scoring='neg_mean_squared_error')
     grid_search.fit(X_train, y_train)
     best_params = grid_search.best_params_
     print("Best parameters found: ", best_params)

     best_xgb_model_2 = grid_search.best_estimator_
     y_pred = best_xgb_model_2.predict(X_test)
     r2 = r2_score(y_test, y_pred)
     print("R-squared: ", r2)
```

```
Best parameters found:  {'max_depth': 3, 'n_estimators': 25}
R-squared:  0.6634425443734272
```

```
[ ]: xgb_model = XGBRegressor()

     param_grid = {
         'n_estimators': [25, 50, 75],
         'max_depth': [ 3, 5, 10],
```

```
    'learning_rate': [0.05, 0.1, 0.15]
}

grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model_3 = grid_search.best_estimator_
y_pred = best_xgb_model_3.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'learning_rate': 0.15, 'max_depth': 3, 'n_estimators':
50}
R-squared:  0.6532830889068985
```

[ ]:
```
xgb_model = XGBRegressor()

param_grid = {
    'n_estimators': [25, 50, 75],
    'max_depth': [ 3, 5, 10],
    'learning_rate': [0.1, 0.15],
    'max_leaves':[25, 50, 200,0]
}


grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model_4 = grid_search.best_estimator_
y_pred = best_xgb_model_4.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'learning_rate': 0.15, 'max_depth': 3, 'max_leaves':
25, 'n_estimators': 50}
R-squared:  0.6532830889068985
```

[ ]:
```
xgb_model = XGBRegressor()

param_grid = {
```

```
     'n_estimators': [25, 50, 75],
    'max_depth': [ 3, 5, 10],
    'learning_rate': [0.1, 0.15],
    'max_leaves':[25, 50, 200,0],
    'tree_method': ['exact', 'approx', 'hist']
}


grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model_5 = grid_search.best_estimator_
y_pred = best_xgb_model_5.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'learning_rate': 0.15, 'max_depth': 3, 'max_leaves':
25, 'n_estimators': 75, 'tree_method': 'exact'}
R-squared:  0.6414683850465416
```

[ ]:
```
xgb_model = XGBRegressor()

param_grid = {
    'n_estimators': [25, 50, 75],
    'max_depth': [ 5,8, 10],
    'learning_rate': [0.05, 0.1, 0.15],
    'max_leaves':[25, 50,200,0],
    'tree_method': ['exact', 'approx', 'hist'],
    'gamma': [0, 0.1, 0.2]
}

grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model_6 = grid_search.best_estimator_
y_pred = best_xgb_model_6.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'gamma': 0, 'learning_rate': 0.1, 'max_depth': 5,
```

```
'max_leaves': 25, 'n_estimators': 75, 'tree_method': 'exact'}
R-squared:  0.6741242999093524
```

[ ]:
```
xgb_model = XGBRegressor()

param_grid = {
    'n_estimators': [25, 50, 75],
    'max_depth': [ 5,8, 10],
    'learning_rate': [0.05, 0.1, 0.15],
    'max_leaves':[25, 50,200,0],
    'tree_method': ['exact', 'approx', 'hist'],
    'gamma': [0.1, 0.2]
}

grid_search = RandomizedSearchCV(estimator=xgb_model,␣
 ↪param_distributions=param_grid, cv=5, scoring='neg_mean_squared_error',␣
 ↪n_iter=200)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_xgb_model_7 = grid_search.best_estimator_
y_pred = best_xgb_model_7.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'tree_method': 'exact', 'n_estimators': 75,
'max_leaves': 25, 'max_depth': 5, 'learning_rate': 0.1, 'gamma': 0.2}
R-squared:  0.6741242999093524
```

[ ]:
```
xgb_model = XGBRegressor()

param_grid = {
    'n_estimators': [100, 500, 750],
    'max_depth': [ 5, 7],
    'learning_rate': [0.05, 0.1],
    'max_leaves':[0, 10],
    'tree_method': [ 'approx', 'exact']
}

grid_search = RandomizedSearchCV(estimator=xgb_model,␣
 ↪param_distributions=param_grid, cv=5, scoring='neg_mean_squared_error',␣
 ↪n_iter=200)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)
```

```
best_xgb_model_8 = grid_search.best_estimator_
y_pred = best_xgb_model_8.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:305:
UserWarning: The total space of parameters 48 is smaller than n_iter=200.
Running 48 iterations. For exhaustive searches, use GridSearchCV.
  warnings.warn(
```

```
Best parameters found:  {'tree_method': 'exact', 'n_estimators': 500,
'max_leaves': 0, 'max_depth': 5, 'learning_rate': 0.05}
R-squared:  0.6813970124277131
```

[ ]:
```
rf_model = RandomForestRegressor()

param_grid = {
    'n_estimators': [20,50, 70],
    'max_depth': [None, 10, 20, 30],
    'criterion':['squared_error', 'absolute_error', 'friedman_mse', 'poisson']
}

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_rf_model = grid_search.best_estimator_
y_pred = best_rf_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
Best parameters found:  {'criterion': 'poisson', 'max_depth': 20,
'n_estimators': 50}
R-squared:  0.6630910611411529
```

[ ]:
```
rf_model = RandomForestRegressor()

param_grid = {
    'n_estimators': [20,50, 70],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'bootstrap':[True , False]
}

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
```

```python
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_rf_model_2 = grid_search.best_estimator_
y_pred = best_rf_model_2.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
```

```
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
```

```
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
```

```
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(

Best parameters found:  {'bootstrap': True, 'max_features': 'sqrt',
'n_estimators': 70}
```

```
R-squared:  0.6682914367280991
```

[ ]:
```python
rf_model = RandomForestRegressor()

param_grid = {
    'n_estimators': [20, 50],
    'max_depth': [None,5, 10],
    'criterion':['squared_error', 'friedman_mse', 'poisson'],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'bootstrap':[True , False]
}

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5,␣
 ↪scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print("Best parameters found: ", best_params)

best_rf_model_3 = grid_search.best_estimator_
y_pred = best_rf_model_3.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("R-squared: ", r2)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
```

```
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:413:
FutureWarning: `max_features='auto'` has been deprecated in 1.1 and will be
removed in 1.3. To keep the past behaviour, explicitly set `max_features=1.0` or
remove this parameter as it is also the default value for RandomForestRegressors
and ExtraTreesRegressors.
  warn(
```

```
Best parameters found:  {'bootstrap': True, 'criterion': 'poisson', 'max_depth':
10, 'max_features': 'sqrt', 'n_estimators': 50}
R-squared:  0.6675360383586544
```

Po hiperparametryzacji najlepszeym modelem okazał się XGBRegressor o następujących hiperparametrach:

- 'gamma': 0,
- – 'learning_rate': 0.1,
- – 'max_depth': 5,
- – 'max_leaves': 25,
- – 'n_estimators': 75,
- 'tree_method': 'exact'

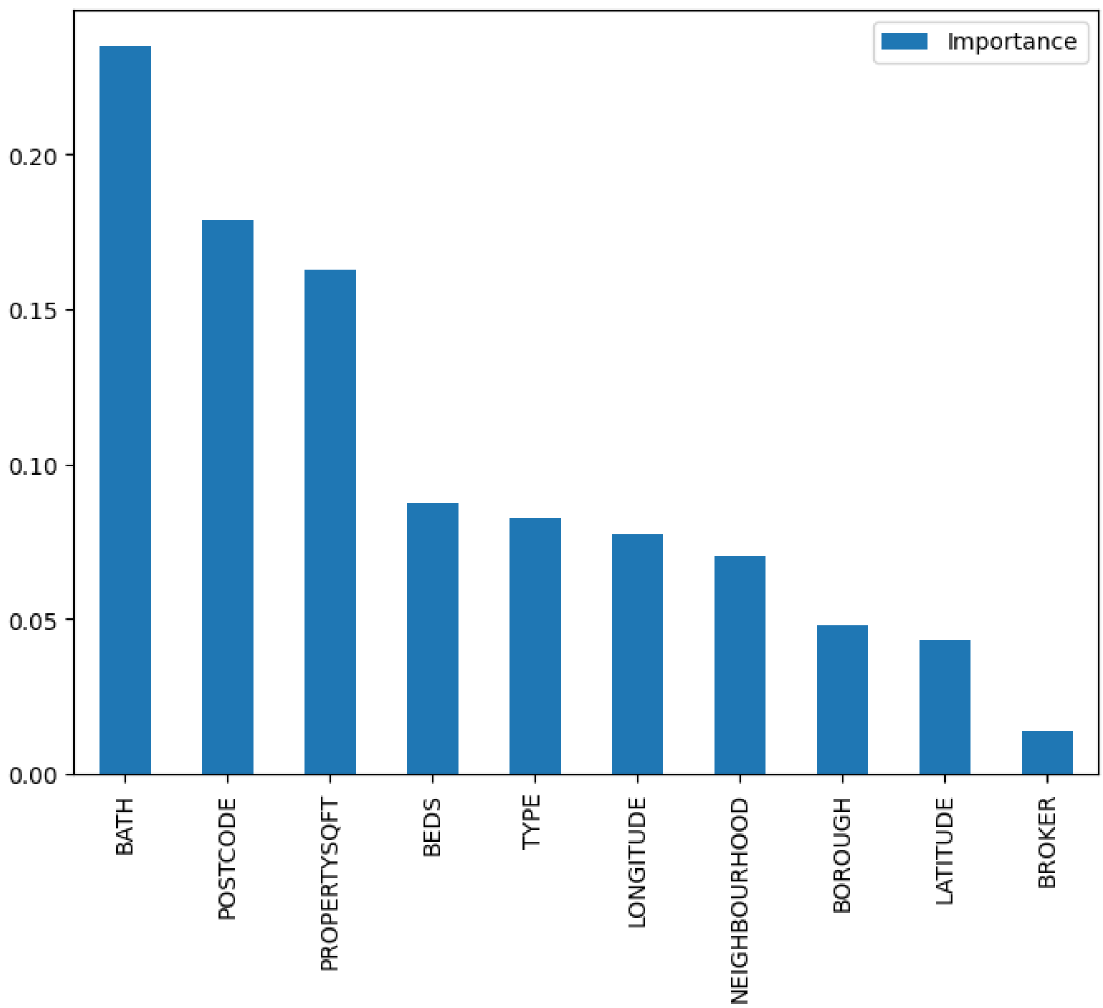
RY2 jest równe dla tego modelu 0.674, co wskazuje na umiarkowanie silny model regresji. Oznacza to, że 67.4% zmienności cen jest wyjaśniana przez zmienne niezależne użyte w szkoleniu modelu regresji.

Ważność cech

[ ]:
```
import lime
import shap
```

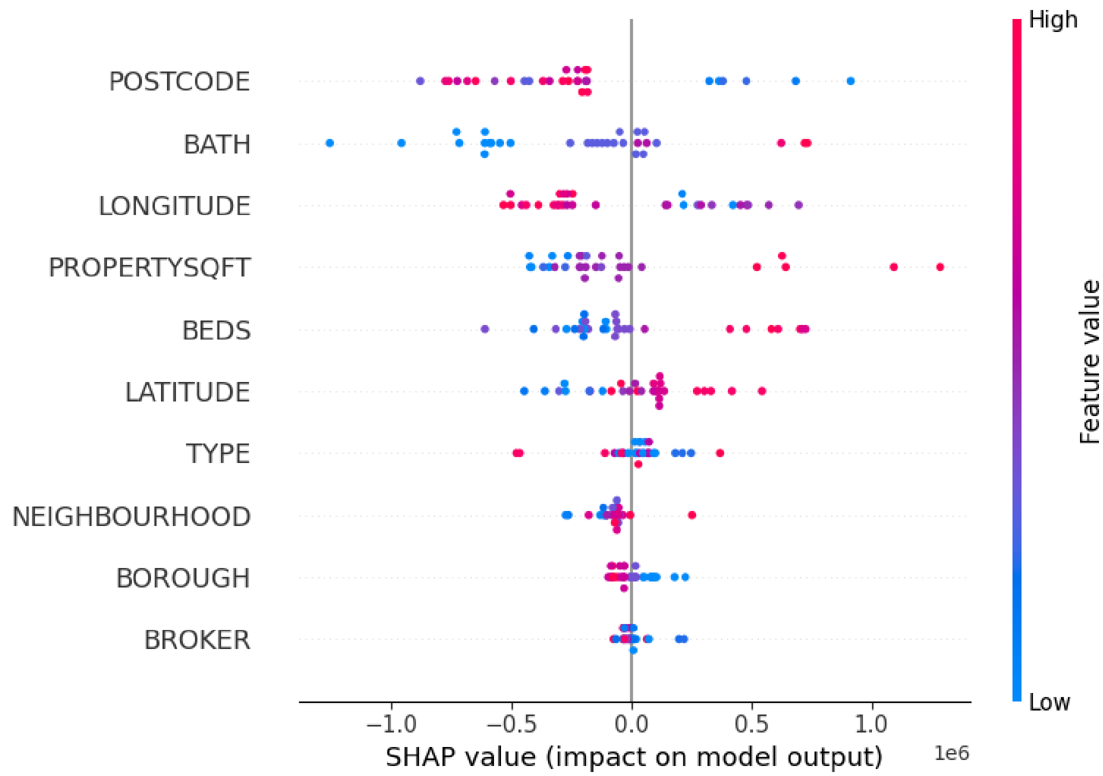
[ ]:
```
feat_importances = pd.DataFrame(best_xgb_model_7.feature_importances_,␣
↪index=X_train.columns, columns=["Importance"])
feat_importances.sort_values(by='Importance', ascending=False, inplace=True)
feat_importances.plot(kind='bar', figsize=(8,6))
shap.initjs()
```

```
<IPython.core.display.HTML object>
```



Na podstawie powyższego wykresu można stwierdzić, że czynniki takie jak liczba łazienek, lokalizacja i wielkość nieruchomości mają kluczowe znaczenie dla określania cen mieszkań. Cechy o niższych wynikach ważności nadal wnoszą wkład do modelu, ale mają mniejszy wpływ na ostateczną prognozę.

[ ]:
```
explainer = shap.Explainer(best_xgb_model_7,  feature_names = X_train.columns)
shap_values = explainer.shap_values(X_test)
shap_values = explainer.shap_values(X_test[:30])
shap.summary_plot(shap_values, X_test[:30])
```



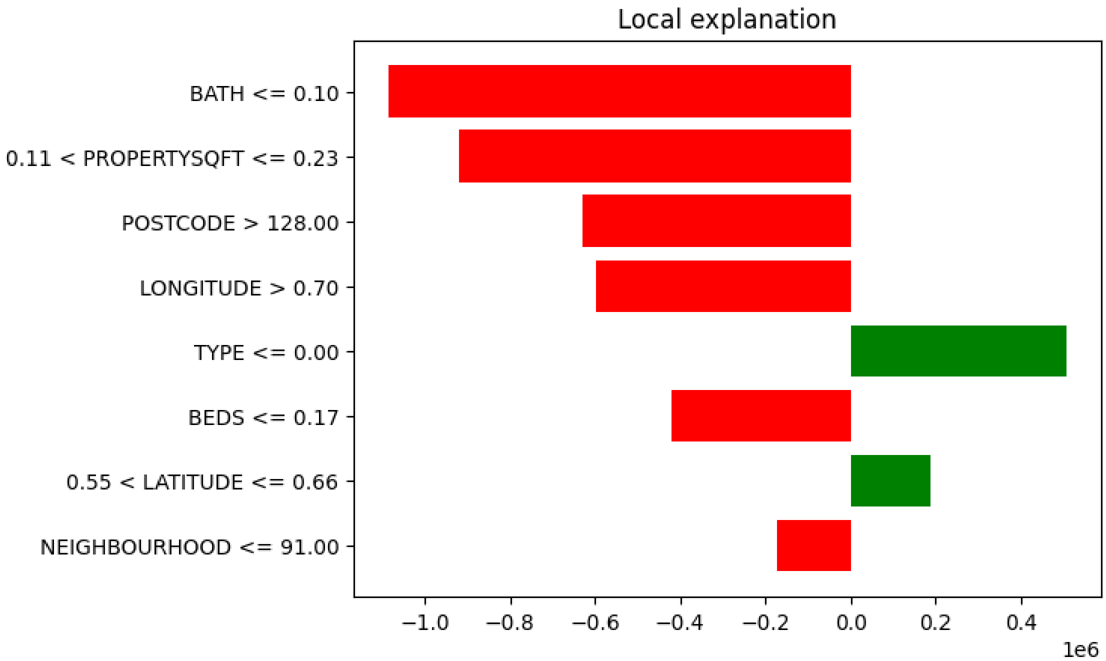
Rozrzut wartości SHAP dla każdej cechy pokazuje zakres jej wpływu na prognozy.

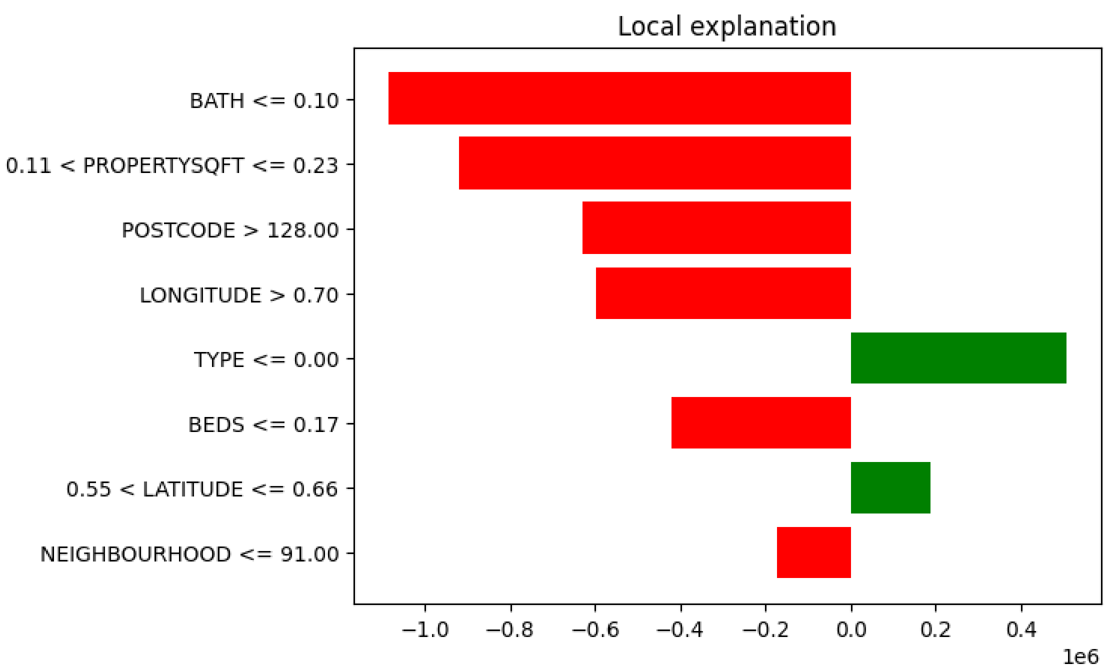
- POSTCODE: Wysokie wartości SHAP dla określonych kodów pocztowych sugerują, że lokalizacja może znacząco zwiększyć lub zmniejszyć przewidywaną cenę mieszkania. Podobne wnioski można wyciągnąć dla zmiennych LONGITUDE i LATITUDE.
- BATH, BEDS: Liczba łazienek (sypialni) również ma znaczący wpływ - większa liczba łazienek (sypialni) zwiększa przewidywaną cenę.
- PROPERTYSQFT: Większe rozmiary nieruchomości powodują wyższe ceny nieruchomości.
- TYPE, BOROUGH, NEIGHBOURHOOD, BROKER: Cechy mają mniejszy, ale nadal zauważalny wpływ na ceny mieszkań.

[ ]:
```
explainer2 = lime.lime_tabular.LimeTabularExplainer(X_train.values,␣
 ↪feature_names=X_train.columns.values.tolist(), class_names=['PRICE'],␣
 ↪verbose=True, mode='regression')

instance = X_train.iloc[4].values.reshape(1, -1)
explanation2 = explainer2.explain_instance(X_test.values[1000],␣
 ↪best_xgb_model_7.predict, num_features=8)
explanation2.as_pyplot_figure()
```

```
Intercept 3499325.1080469214
Prediction_local [374777.08142794]
Right: 262017.66
```

[ ]:

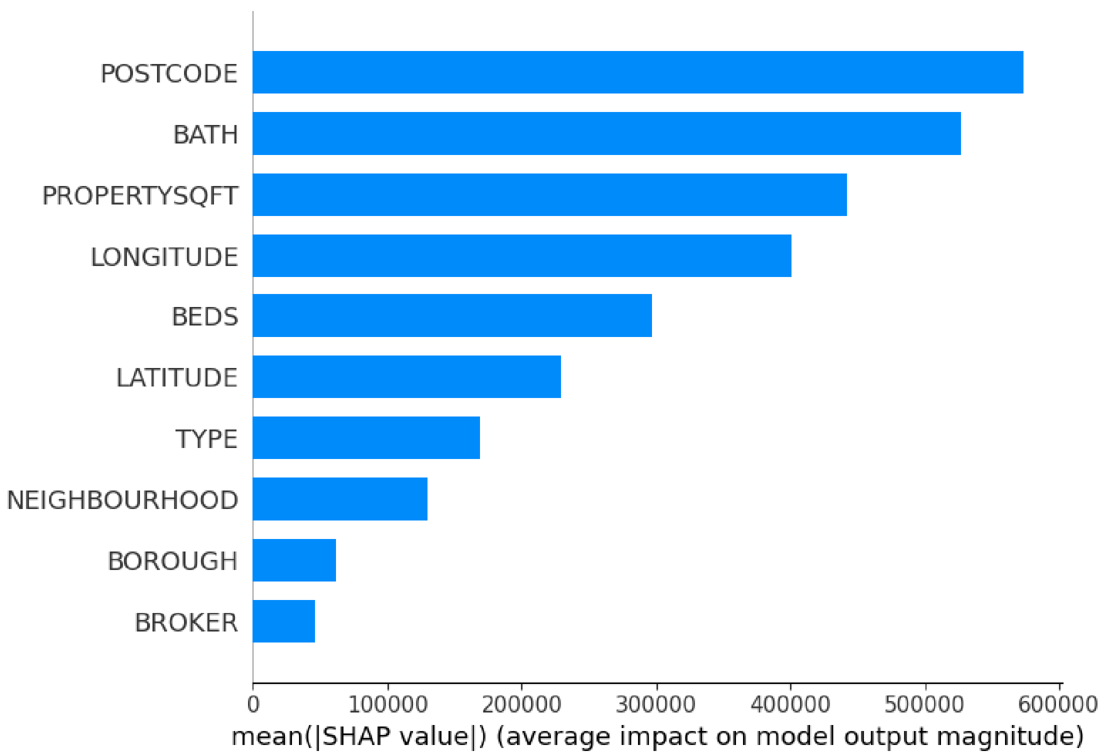




BATH <= 0.10: Liczba łazienek znacząco wpływa na obniżenie prognozy ceny gdy przeskalowana wartość jest mniejsza lub równa od 0.10. 0.11 < PROPERTYSQFT <= 0.23: oznacza, że jeżeli przeskalowana wartość powierzchni nieruchomości mieści się w przedziale od 0,11 do 0,23, to ma ona znaczny negatywny wpływ na prognozowaną cenę POSTCODE > 128.00: gdy wartość kategorii przypisana kodowi pocztowemu jest większa niż 128.00 to przewidywana cena jest niższa. LONGITUDE > 0.70: gdy przeskalowana wartość długości geograficznej jest większa niż 0.70, to prognozowana cena nieruchomości jest niższa. TYPE <= 0.00: Gdy typ budynku jest 'co-op' to zwiększa to cene nieruchomości. BEDS <= 0.17: Liczba łazienek wpływa na obniżenie prognozy ceny gdy jej przeskalowana wartość jest mniejsza lub równa od 0.17. 0.55 < LATITUDE <= 0.66: Szerokość geograficzna pozytywnie wpływa na cene nieruchomości gdy jej przeskalowana wartość mieści się w przedziale od 0.55 do 0.66. NEIGHBOURHOOD <= 91.00: To oznacza, że gdy wartość przyporządkowanej kategorii sąsiedztwa jest mniejsza lub równa 91.00, to prognozowana cena nieruchomości jest niższa - wpływ ten jest zdecydowanie mniejszy w porównaniu do innych cech.

[ ]:
```
shap_values = explainer.shap_values(X_test)
print("Variable Importance Plot - Global Interpretation")

shap.summary_plot(shap_values, X_test, plot_type="bar")
```

```
Variable Importance Plot - Global Interpretation
```



Wykres przedstawia globalną interpretację ważności zmiennych modelu za pomocą wartości SHAP. Najbardziej wpływowymi cechami są zmienne: POSTCODE, BATH i PROPERTYSQFT. Pozostałymi mniej istotnymi cechami są LONGITUDE, BEDS, LATITUDE, TYPE, NEIGHBOURHOOD, BOROUGH i BROKER, które również przyczyniają się do dokładności modelu.

Ze względu na umiarkowanie dobrą jakość wytrenowanych modeli regresyjnych, zdecydowałyśmy się nie skupiać na klasyfikacji.