homellc

December 12, 2023

Data Collecting

[123]:
```
import numpy as np
import pandas as pd
```

[124]:
```
#Reading CASE-SHILLER Index into a dataframe
df_CS = pd.read_csv(r"C:\Users\Prachi\Downloads\CSUSHPISA (1).csv")

#Changing dtype of date column
df_CS["DATE"] = pd.to_datetime(df_CS["DATE"])

#Selecting data post JUNE 2001
mask = df_CS["DATE"] >= "2001-07-01"
df_CS = df_CS[mask]

#Resetting Index
df_CS.reset_index(inplace = True)
df_CS.drop(columns = ["index"], inplace = True)

# Creating "Year" and "Month" columns
df_CS["Year"] = pd.DatetimeIndex(df_CS["DATE"]).year
df_CS["Month"] = pd.DatetimeIndex(df_CS["DATE"]).month
print(df_CS.shape)
df_CS.tail()
```

```
(247, 4)
```

[124]:

	DATE	CSUSHPISA	Year	Month
242	2022-02-01	291.153	2022	2
243	2022-03-01	296.445	2022	3
244	2022-04-01	300.573	2022	4
245	2022-05-01	303.762	2022	5
246	2022-06-01	304.724	2022	6

[125]:
```
# Reading Unemployment Rate Data into a dataframe
df_unemp = pd.read_csv(r"C:\Users\Prachi\Downloads\UNRATE (2).csv", names =
 ↪["DATE", "UNRATE"], skiprows = 1)
df_unemp.drop(df_unemp.index[267:912], inplace = True)
```

```
print(df_unemp.shape)
df_unemp.tail()
```

```
(247, 2)
```

[125]:

	DATE	UNRATE
242	2022-02-01	3.8
243	2022-03-01	3.6
244	2022-04-01	3.6
245	2022-05-01	3.6
246	2022-06-01	3.6

[126]:
```
# Reading Per Capita GDP Data into a dataframe
df_pcgdp = pd.read_csv(r"C:\Users\Prachi\Downloads\A939RX0Q048SBEA.csv", names␣
↪= ["DATE", "Per_Capita_GDP"], skiprows = 1)
print(df_pcgdp.shape)
df_pcgdp.tail()
```

```
(83, 2)
```

[126]:

	DATE	Per_Capita_GDP
78	2021-04-01	64157.0
79	2021-07-01	64615.0
80	2021-10-01	65651.0
81	2022-01-01	65286.0
82	2022-04-01	65127.0

[127]:
```
# Interest Rate Data
df_Fed_rate = pd.read_csv(r"C:\Users\Prachi\Downloads\FEDFUNDS (1).csv")
df_Fed_rate.drop(df_Fed_rate.index[267:1828], inplace = True)
print(df_Fed_rate.shape)
df_Fed_rate.tail()
```

```
(247, 2)
```

[127]:

	DATE	FEDFUNDS
242	2022-02-01	0.08
243	2022-03-01	0.20
244	2022-04-01	0.33
245	2022-05-01	0.77
246	2022-06-01	1.21

[128]:
```
# Reading Per Capita GDP Data into a dataframe
df_cons_price_index = pd.read_csv(r"C:\Users\Prachi\Downloads\WPUSI012011.csv",␣
↪names = ["DATE", "Cons_Materials"], skiprows = 1)
df_cons_price_index.drop(df_cons_price_index.index[267:611], inplace = True)
print(df_cons_price_index.shape)
```

```
df_cons_price_index.tail()
```

(247, 2)

[128]:

	DATE	Cons_Materials
242	2022-02-01	343.583
243	2022-03-01	345.852
244	2022-04-01	343.786
245	2022-05-01	353.015
246	2022-06-01	349.800

[129]:
```
# Consumer Price Index
df_CPI = pd.read_csv(r"C:\Users\Prachi\Downloads\CPIAUCSL (1).csv", names =
 ↪["DATE", "CPI"], skiprows = 1)
df_CPI.drop(df_CPI.index[267:923], inplace = True)
print(df_CPI.shape)
df_CPI.tail()
```

(247, 2)

[129]:

	DATE	CPI
242	2022-02-01	284.610
243	2022-03-01	287.472
244	2022-04-01	288.611
245	2022-05-01	291.268
246	2022-06-01	294.728

[130]:
```
# Monthly new house supply
df_house = pd.read_csv(r"C:\Users\Prachi\Downloads\MSACSR (1).csv", names =
 ↪["DATE", "Houses"], skiprows = 1)
df_house.drop(df_house.index[267:731], inplace = True)
print(df_house.shape)
df_house.tail()
```

(247, 2)

[130]:

	DATE	Houses
242	2022-02-01	6.2
243	2022-03-01	7.0
244	2022-04-01	8.5
245	2022-05-01	8.3
246	2022-06-01	9.5

[131]:
```
# Population above 65

df_oldpop = pd.read_csv(r"C:\Users\Prachi\Downloads\SPPOP65UPTOZSUSA (1).csv",
 ↪names = ["DATE", "old_percent"], skiprows = 1)
```

```
print(df_oldpop.shape)
df_oldpop.tail()
```

(22, 2)

[131]:

	DATE	old_percent
17	2018-01-01	15.397698
18	2019-01-01	15.791801
19	2020-01-01	16.223400
20	2021-01-01	16.678895
21	2022-01-01	17.128121

[132]:

```
# Urban Population Percent

df_urban = pd.read_csv(r"C:\Users\Prachi\Downloads\LREM64TTUSM156S.csv", names␣
↪= ["DATE", "Urban_pop"], skiprows = 1)
df_urban.drop(df_urban.index[267:851], inplace = True)
print(df_urban.shape)
df_urban.tail()
```

(247, 2)

[132]:

	DATE	Urban_pop
242	2022-02-01	70.900503
243	2022-03-01	71.266867
244	2022-04-01	71.222821
245	2022-05-01	71.370775
246	2022-06-01	71.228684

[133]:

```
# Housing Subsidies

df_subsidy = pd.read_csv(r"C:\Users\Prachi\Downloads\L312051A027NBEA (1).csv",␣
↪names = ["DATE", "Subsidy"], skiprows = 1)
print(df_subsidy.shape)
df_subsidy.tail()
```

(22, 2)

[133]:

	DATE	Subsidy
17	2018-01-01	38.859
18	2019-01-01	40.185
19	2020-01-01	44.147
20	2021-01-01	45.299
21	2022-01-01	48.021

[134]:

```
# Working age population
```

```
df_working = pd.read_csv(r"C:\Users\Prachi\Downloads\LFWA64TTUSM647S (2).csv",␣
↪names = ["DATE", "Working_Population"], skiprows = 1)
df_working.drop(df_working.index[267:563], inplace = True)
print(df_working.shape)
df_working.tail()
```

```
(247, 2)
```

[134]:

	DATE	Working_Population
242	2022-02-01	2.071042e+08
243	2022-03-01	2.070130e+08
244	2022-04-01	2.070650e+08
245	2022-05-01	2.072705e+08
246	2022-06-01	2.073947e+08

[135]:
```
# Real Median Household Income

df_income = pd.read_csv(r"C:\Users\Prachi\Downloads\MEHOINUSA672N.csv", names =␣
↪["DATE", "Income"], skiprows = 1)
print(df_income.shape)
df_income.tail()
```

```
(22, 2)
```

[135]:

	DATE	Income
17	2018-01-01	73030
18	2019-01-01	78250
19	2020-01-01	76660
20	2021-01-01	76330
21	2022-01-01	74580

[136]:
```
# Number of households

df_households = pd.read_csv(r"C:\Users\Prachi\Downloads\TTLHH (1).csv", names =␣
↪["DATE", "Num_Households"], skiprows = 1)
print(df_households.shape)
df_households.tail()
```

```
(22, 2)
```

[136]:

	DATE	Num_Households
17	2018-01-01	127586.0
18	2019-01-01	128579.0
19	2020-01-01	128451.0
20	2021-01-01	129224.0
21	2022-01-01	131202.0

Data Preprocessing

[137]:
```
# Merging Per Capita GDP (Quarterly data)
df_pcgdp["DATE"] = pd.to_datetime(df_pcgdp["DATE"])
df_CS = pd.merge(df_CS,df_pcgdp, how = "left")
df_CS.head()
```

[137]:
```
         DATE  CSUSHPISA  Year  Month  Per_Capita_GDP
0  2001-12-01    116.455  2001     12             NaN
1  2002-01-01    117.144  2002      1         50091.0
2  2002-02-01    117.845  2002      2             NaN
3  2002-03-01    118.687  2002      3             NaN
4  2002-04-01    119.611  2002      4         50286.0
```

[138]:
```
df = pd.DataFrame()
df_bymonth = [df_CS, df_working, df_house, df_CPI, df_unemp,␣
↪df_cons_price_index, df_Fed_rate]
for df1 in df_bymonth:
    df1["DATE"] = pd.to_datetime(df1["DATE"])
    df1 = df1.set_index("DATE")
    df = pd.concat([df,df1], axis = 1)
print(df.shape)
df.head(5)
```

```
(247, 10)
```

[138]:
```
            CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2001-12-01    116.455  2001     12             NaN        1.826419e+08
2002-01-01    117.144  2002      1         50091.0        1.825664e+08
2002-02-01    117.845  2002      2             NaN        1.827984e+08
2002-03-01    118.687  2002      3             NaN        1.830783e+08
2002-04-01    119.611  2002      4         50286.0        1.832605e+08

            Houses    CPI  UNRATE  Cons_Materials  FEDFUNDS
DATE
2001-12-01     3.8  177.4     5.7           141.7      1.82
2002-01-01     4.2  177.7     5.7           142.0      1.73
2002-02-01     4.0  178.0     5.7           142.2      1.74
2002-03-01     4.1  178.5     5.7           143.2      1.73
2002-04-01     4.3  179.3     5.9           143.5      1.75
```

[139]:
```
# Merging other dataframes
others = [df_urban, df_households, df_income, df_subsidy, df_oldpop]
for df1 in others:
    if "Year" not in df1.columns:
        df1["Year"] = pd.DatetimeIndex(df1["DATE"]).year
        df1.set_index("DATE", inplace = True)
        df = pd.merge(df, df1, how = "left", on = "Year")
```

```
    else:
        df1.set_index("DATE", inplace = True)
        df = pd.merge(df, df1, how = "left", on = "Year")
df["DATE"] = df_CS["DATE"]
df.set_index("DATE", inplace = True)
df.head()
```

[139]:
```
            CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2001-12-01    116.455  2001     12             NaN        1.826419e+08
2002-01-01    117.144  2002      1         50091.0        1.825664e+08
2002-02-01    117.144  2002      1         50091.0        1.825664e+08
2002-03-01    117.144  2002      1         50091.0        1.825664e+08
2002-04-01    117.144  2002      1         50091.0        1.825664e+08

            Houses    CPI  UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
DATE
2001-12-01     3.8  177.4     5.7           141.7      1.82  72.226108
2002-01-01     4.2  177.7     5.7           142.0      1.73  72.016140
2002-02-01     4.2  177.7     5.7           142.0      1.73  72.333005
2002-03-01     4.2  177.7     5.7           142.0      1.73  72.131883
2002-04-01     4.2  177.7     5.7           142.0      1.73  71.914847

            Num_Households  Income  Subsidy  old_percent
DATE
2001-12-01        108209.0   66360   20.573    12.296945
2002-01-01        109297.0   65820   24.183    12.287458
2002-02-01        109297.0   65820   24.183    12.287458
2002-03-01        109297.0   65820   24.183    12.287458
2002-04-01        109297.0   65820   24.183    12.287458
```

[140]:
```
print(df.shape)
```

```
(2917, 15)
```

[141]:
```
df.isna().sum()
```

[141]:
```
CSUSHPISA                0
Year                     0
Month                    0
Per_Capita_GDP        1945
Working_Population       0
Houses                   0
CPI                      0
UNRATE                   0
Cons_Materials           0
FEDFUNDS                 0
```

```
Urban_pop              0
Num_Households         0
Income                 0
Subsidy                0
old_percent            0
dtype: int64
```

The "Per_Capita_GDP" column has missing values because the data was quarterly,We will first fill in the missing values in the "Per_Capita_GDP" column using linear interpolation.

[142]:
```
# Filling missing values in the Per_Capita_GDP column using linear interpolation
df["Per_Capita_GDP"] = df["Per_Capita_GDP"].interpolate()
```

[143]:
```
df
```

[143]:
```
            CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2001-12-01    116.455  2001     12             NaN        1.826419e+08
2002-01-01    117.144  2002      1         50091.0        1.825664e+08
2002-02-01    117.144  2002      1         50091.0        1.825664e+08
2002-03-01    117.144  2002      1         50091.0        1.825664e+08
2002-04-01    117.144  2002      1         50091.0        1.825664e+08
…                 …   …     …           …                  …
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08

            Houses      CPI  UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
DATE
2001-12-01     3.8  177.400     5.7           141.7      1.82  72.226108
2002-01-01     4.2  177.700     5.7           142.0      1.73  72.016140
2002-02-01     4.2  177.700     5.7           142.0      1.73  72.333005
2002-03-01     4.2  177.700     5.7           142.0      1.73  72.131883
2002-04-01     4.2  177.700     5.7           142.0      1.73  71.914847
…              …      …     …           …        …       …
NaT            9.5  294.728     3.6           349.8      1.21  70.900503
NaT            9.5  294.728     3.6           349.8      1.21  71.266867
NaT            9.5  294.728     3.6           349.8      1.21  71.222821
NaT            9.5  294.728     3.6           349.8      1.21  71.370775
NaT            9.5  294.728     3.6           349.8      1.21  71.228684

            Num_Households  Income  Subsidy  old_percent
DATE
2001-12-01        108209.0   66360   20.573    12.296945
2002-01-01        109297.0   65820   24.183    12.287458
```

```
2002-02-01        109297.0   65820   24.183    12.287458
2002-03-01        109297.0   65820   24.183    12.287458
2002-04-01        109297.0   65820   24.183    12.287458
…                        …      …       …          …
NaT               131202.0   74580   48.021    17.128121
NaT               131202.0   74580   48.021    17.128121
NaT               131202.0   74580   48.021    17.128121
NaT               131202.0   74580   48.021    17.128121
NaT               131202.0   74580   48.021    17.128121

[2917 rows x 15 columns]
```

[144]:
```
df.dropna(inplace = True)
```

[145]:
```
df.isna().sum()
```

[145]:
```
CSUSHPISA             0
Year                  0
Month                 0
Per_Capita_GDP        0
Working_Population    0
Houses                0
CPI                   0
UNRATE                0
Cons_Materials        0
FEDFUNDS              0
Urban_pop             0
Num_Households        0
Income                0
Subsidy               0
old_percent           0
dtype: int64
```

[146]:
```
df
```

[146]:
```
            CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2002-01-01    117.144  2002      1         50091.0        1.825664e+08
2002-02-01    117.144  2002      1         50091.0        1.825664e+08
2002-03-01    117.144  2002      1         50091.0        1.825664e+08
2002-04-01    117.144  2002      1         50091.0        1.825664e+08
2002-05-01    117.144  2002      1         50091.0        1.825664e+08
…                 …     …      …               …                  …
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
NaT           304.724  2022      6         65127.0        2.073947e+08
```

```
NaT            304.724  2022      6        65127.0        2.073947e+08

             Houses       CPI  UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
DATE
2002-01-01      4.2  177.700     5.7           142.0      1.73  72.016140
2002-02-01      4.2  177.700     5.7           142.0      1.73  72.333005
2002-03-01      4.2  177.700     5.7           142.0      1.73  72.131883
2002-04-01      4.2  177.700     5.7           142.0      1.73  71.914847
2002-05-01      4.2  177.700     5.7           142.0      1.73  72.026012
…             …      …      …            …       …          …
NaT             9.5  294.728     3.6           349.8      1.21  70.900503
NaT             9.5  294.728     3.6           349.8      1.21  71.266867
NaT             9.5  294.728     3.6           349.8      1.21  71.222821
NaT             9.5  294.728     3.6           349.8      1.21  71.370775
NaT             9.5  294.728     3.6           349.8      1.21  71.228684

             Num_Households  Income  Subsidy  old_percent
DATE
2002-01-01         109297.0   65820   24.183    12.287458
2002-02-01         109297.0   65820   24.183    12.287458
2002-03-01         109297.0   65820   24.183    12.287458
2002-04-01         109297.0   65820   24.183    12.287458
2002-05-01         109297.0   65820   24.183    12.287458
…                       …      …       …            …
NaT                131202.0   74580   48.021    17.128121
NaT                131202.0   74580   48.021    17.128121
NaT                131202.0   74580   48.021    17.128121
NaT                131202.0   74580   48.021    17.128121
NaT                131202.0   74580   48.021    17.128121

[2916 rows x 15 columns]
```

[147]:
```
df.to_csv("prepared_dataset.csv")
```

[148]:
```
us_house_price_df = pd.read_csv("prepared_dataset.csv").set_index("DATE")
us_house_price_df.head()
```

[148]:
```
             CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2002-01-01     117.144  2002      1         50091.0        1.825664e+08
2002-02-01     117.144  2002      1         50091.0        1.825664e+08
2002-03-01     117.144  2002      1         50091.0        1.825664e+08
2002-04-01     117.144  2002      1         50091.0        1.825664e+08
2002-05-01     117.144  2002      1         50091.0        1.825664e+08

             Houses     CPI  UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
DATE
```

```
2002-01-01      4.2  177.7     5.7            142.0      1.73  72.016140
2002-02-01      4.2  177.7     5.7            142.0      1.73  72.333005
2002-03-01      4.2  177.7     5.7            142.0      1.73  72.131883
2002-04-01      4.2  177.7     5.7            142.0      1.73  71.914847
2002-05-01      4.2  177.7     5.7            142.0      1.73  72.026012

            Num_Households  Income  Subsidy  old_percent
DATE
2002-01-01        109297.0   65820   24.183    12.287458
2002-02-01        109297.0   65820   24.183    12.287458
2002-03-01        109297.0   65820   24.183    12.287458
2002-04-01        109297.0   65820   24.183    12.287458
2002-05-01        109297.0   65820   24.183    12.287458
```

[149]:
```python
#Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
```

[150]:
```python
us_house_price_df = pd.read_csv("prepared_dataset.csv").set_index("DATE")
us_house_price_df.head()
```

[150]:
```
            CSUSHPISA  Year  Month  Per_Capita_GDP  Working_Population  \
DATE
2002-01-01    117.144  2002      1         50091.0        1.825664e+08
2002-02-01    117.144  2002      1         50091.0        1.825664e+08
2002-03-01    117.144  2002      1         50091.0        1.825664e+08
2002-04-01    117.144  2002      1         50091.0        1.825664e+08
2002-05-01    117.144  2002      1         50091.0        1.825664e+08

            Houses    CPI  UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
DATE
2002-01-01      4.2  177.7     5.7            142.0      1.73  72.016140
2002-02-01      4.2  177.7     5.7            142.0      1.73  72.333005
2002-03-01      4.2  177.7     5.7            142.0      1.73  72.131883
2002-04-01      4.2  177.7     5.7            142.0      1.73  71.914847
2002-05-01      4.2  177.7     5.7            142.0      1.73  72.026012

            Num_Households  Income  Subsidy  old_percent
DATE
2002-01-01        109297.0   65820   24.183    12.287458
2002-02-01        109297.0   65820   24.183    12.287458
2002-03-01        109297.0   65820   24.183    12.287458
2002-04-01        109297.0   65820   24.183    12.287458
```

```
2002-05-01        109297.0   65820   24.183    12.287458
```

[151]:
```python
# Dropping year and month columns
us_house_price_df.drop(columns = ["Year", "Month"], inplace = True)
```

Exploratory Data Analysis (EDA) Summary Statistics: Calculate and display summary statistics for each variable, including mean, median, standard deviation, minimum, and maximum values. This gives you an overview of the data's central tendencies and variability.

[152]:
```python
# Summary statistics
summary_stats = us_house_price_df.describe()
summary_stats
```

[152]:
```
         CSUSHPISA  Per_Capita_GDP  Working_Population       Houses  \
count  2916.000000     2916.000000        2.916000e+03  2916.000000
mean    173.393158    56733.444444        1.990766e+08     5.939300
std      35.656839     3894.028787        6.902281e+06     1.900553
min     117.144000    50091.000000        1.825664e+08     3.300000
25%     146.394000    54100.000000        1.949518e+08     4.500000
50%     168.634000    55575.500000        2.011083e+08     5.400000
75%     187.993000    59440.540000        2.053709e+08     6.700000
max     304.724000    65651.000000        2.073947e+08    12.200000

               CPI       UNRATE  Cons_Materials     FEDFUNDS    Urban_pop  \
count  2916.000000  2916.000000     2916.000000  2916.000000  2916.000000
mean    225.173831     6.089300      204.752848     1.290226    69.550604
std      26.279576     1.963623       39.039207     1.538251     2.079692
min     177.700000     3.500000      142.000000     0.050000    60.193856
25%     203.800000     4.700000      182.500000     0.120000    67.699590
50%     228.329000     5.600000      205.500000     0.650000    70.097404
75%     244.004000     7.300000      219.700000     1.910000    71.363247
max     294.728000    14.700000      353.015000     5.260000    72.333005

       Num_Households        Income      Subsidy  old_percent
count     2916.000000   2916.000000  2916.000000  2916.000000
mean    120384.419753  68561.728395    33.665000    13.828983
std       6278.891569   4444.196095     5.771877     1.436413
min     109297.000000  63350.000000    24.183000    12.277934
25%     116011.000000  65760.000000    29.512000    12.507804
50%     121084.000000  66780.000000    33.283000    13.584437
75%     126224.000000  72090.000000    37.550000    15.066290
max     131202.000000  78250.000000    48.021000    17.128121
```

Correlation Analysis: Calculate the correlation matrix to measure the linear relationships between variables.

[153]:
```python
# Correlation matrix
corr_matrix = us_house_price_df.corr()
```

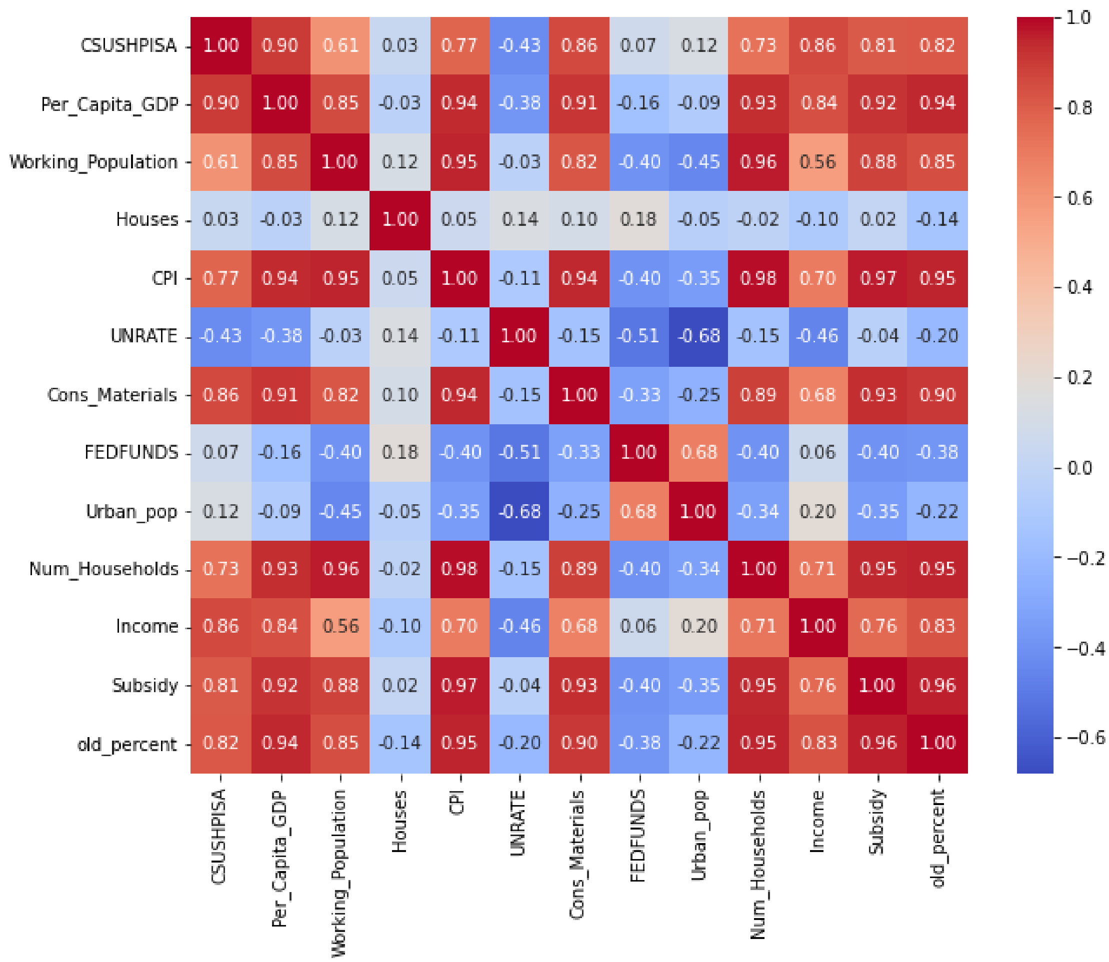
```
corr_matrix
```

[153]:

```
                    CSUSHPISA  Per_Capita_GDP  Working_Population    Houses  \
CSUSHPISA            1.000000        0.895990            0.612680  0.034838
Per_Capita_GDP       0.895990        1.000000            0.853468 -0.033405
Working_Population   0.612680        0.853468            1.000000  0.118453
Houses               0.034838       -0.033405            0.118453  1.000000
CPI                  0.772845        0.937533            0.947566  0.048284
UNRATE              -0.429101       -0.378423           -0.034271  0.138918
Cons_Materials       0.855816        0.913280            0.822703  0.099472
FEDFUNDS             0.072644       -0.160404           -0.398701  0.182683
Urban_pop            0.121200       -0.087122           -0.445005 -0.051654
Num_Households       0.729666        0.932990            0.963337 -0.022643
Income               0.860906        0.841389            0.560252 -0.100339
Subsidy              0.809054        0.917536            0.881389  0.015319
old_percent          0.816216        0.943933            0.845728 -0.141864

                         CPI    UNRATE  Cons_Materials  FEDFUNDS  Urban_pop  \
CSUSHPISA           0.772845 -0.429101        0.855816  0.072644   0.121200
Per_Capita_GDP      0.937533 -0.378423        0.913280 -0.160404  -0.087122
Working_Population  0.947566 -0.034271        0.822703 -0.398701  -0.445005
Houses              0.048284  0.138918        0.099472  0.182683  -0.051654
CPI                 1.000000 -0.108603        0.944273 -0.399932  -0.354494
UNRATE             -0.108603  1.000000       -0.148995 -0.514338  -0.680275
Cons_Materials      0.944273 -0.148995        1.000000 -0.333749  -0.249286
FEDFUNDS           -0.399932 -0.514338       -0.333749  1.000000   0.678694
Urban_pop          -0.354494 -0.680275       -0.249286  0.678694   1.000000
Num_Households      0.983142 -0.149748        0.885856 -0.396254  -0.335645
Income              0.696503 -0.458376        0.676953  0.060167   0.198227
Subsidy             0.965424 -0.044436        0.929830 -0.402370  -0.352257
old_percent         0.951391 -0.202771        0.902977 -0.381224  -0.222611

                    Num_Households    Income   Subsidy  old_percent
CSUSHPISA                 0.729666  0.860906  0.809054     0.816216
Per_Capita_GDP            0.932990  0.841389  0.917536     0.943933
Working_Population        0.963337  0.560252  0.881389     0.845728
Houses                   -0.022643 -0.100339  0.015319    -0.141864
CPI                       0.983142  0.696503  0.965424     0.951391
UNRATE                   -0.149748 -0.458376 -0.044436    -0.202771
Cons_Materials            0.885856  0.676953  0.929830     0.902977
FEDFUNDS                 -0.396254  0.060167 -0.402370    -0.381224
Urban_pop                -0.335645  0.198227 -0.352257    -0.222611
Num_Households            1.000000  0.713021  0.946049     0.952633
Income                    0.713021  1.000000  0.760592     0.828154
Subsidy                   0.946049  0.760592  1.000000     0.960280
old_percent               0.952633  0.828154  0.960280     1.000000
```

correlation matrix using a heatmap to identify strong positive and negative correlations

[154]:
```
# Visualize correlations using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.show()
```



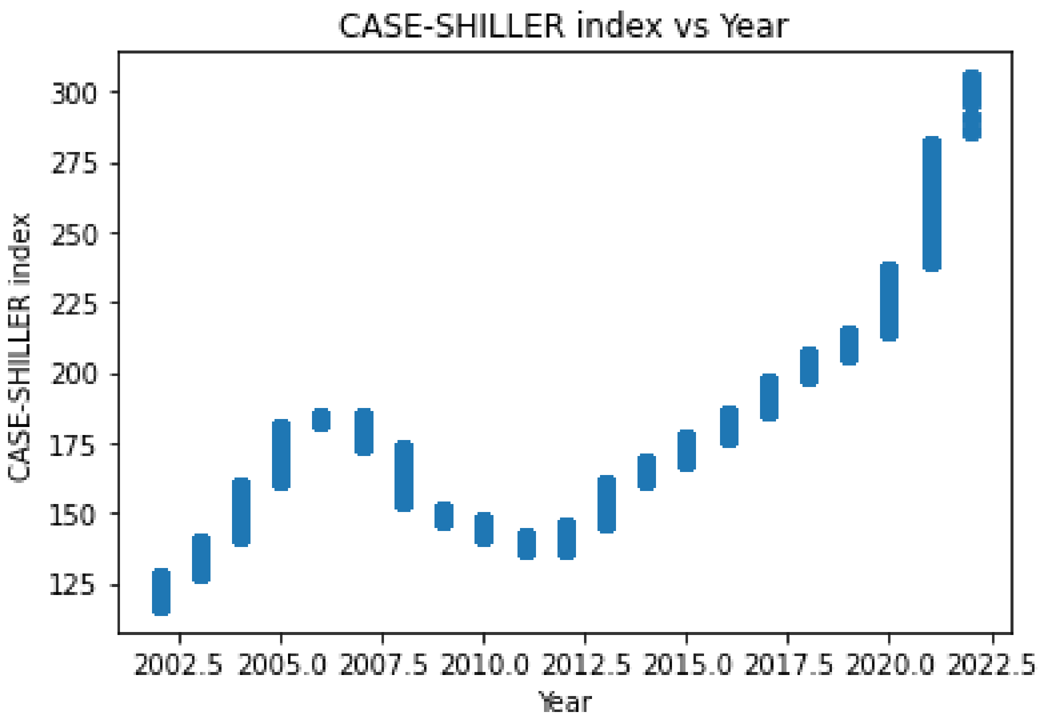
[155]:
```
us_house_price_df.columns
```

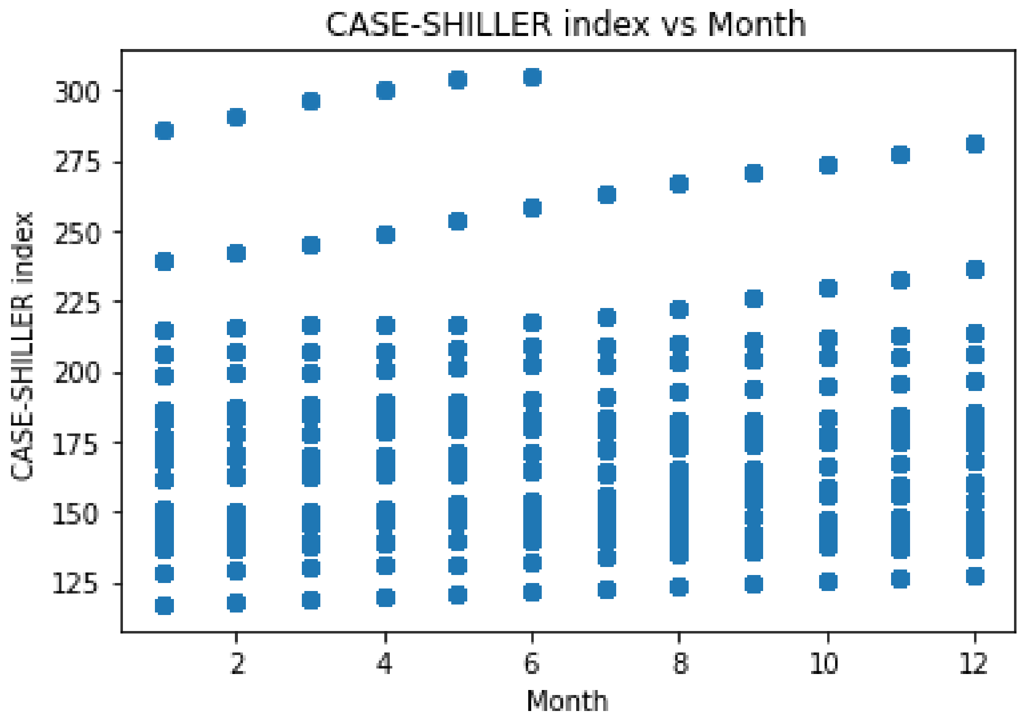
[155]:
```
Index(['CSUSHPISA', 'Per_Capita_GDP', 'Working_Population', 'Houses', 'CPI',
       'UNRATE', 'Cons_Materials', 'FEDFUNDS', 'Urban_pop', 'Num_Households',
       'Income', 'Subsidy', 'old_percent'],
      dtype='object')
```

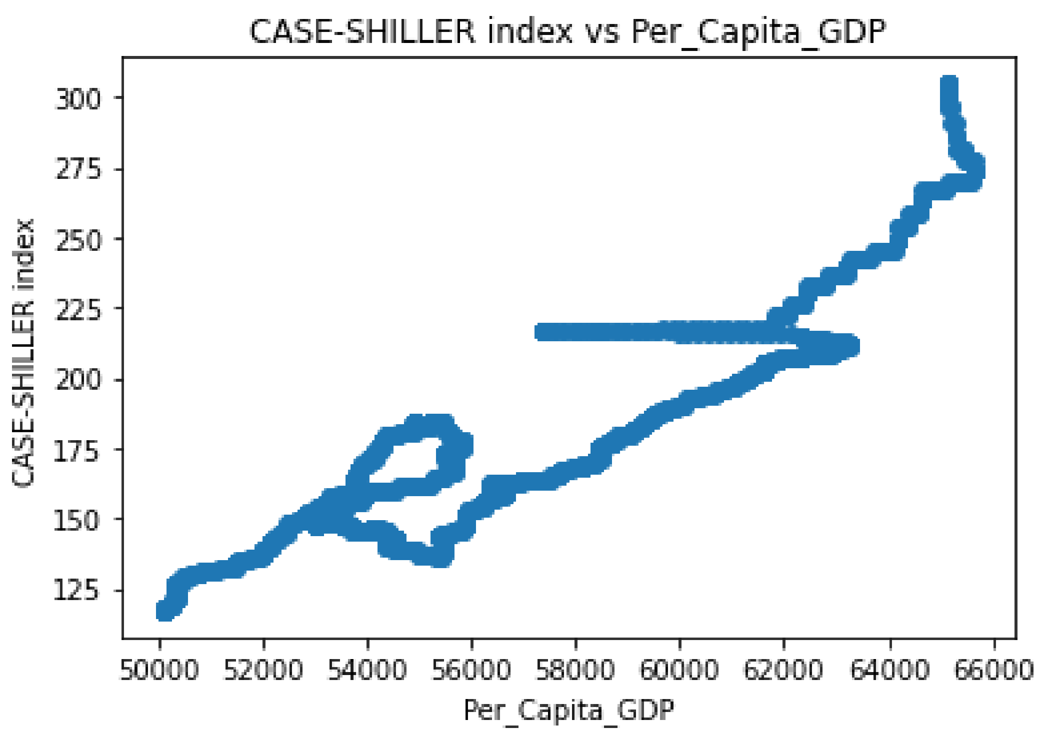
[156]:
```
factors = ['CSUSHPISA', 'UNRATE', 'Per_Capita_GDP', 'FEDFUNDS',␣
↪'Cons_Material', 'CPI', 'Houses', 'Num_Households', 'old_age_pop',␣
↪'urban_pop_us', 'Subsidy', 'working_age_pop', 'median_income']
```

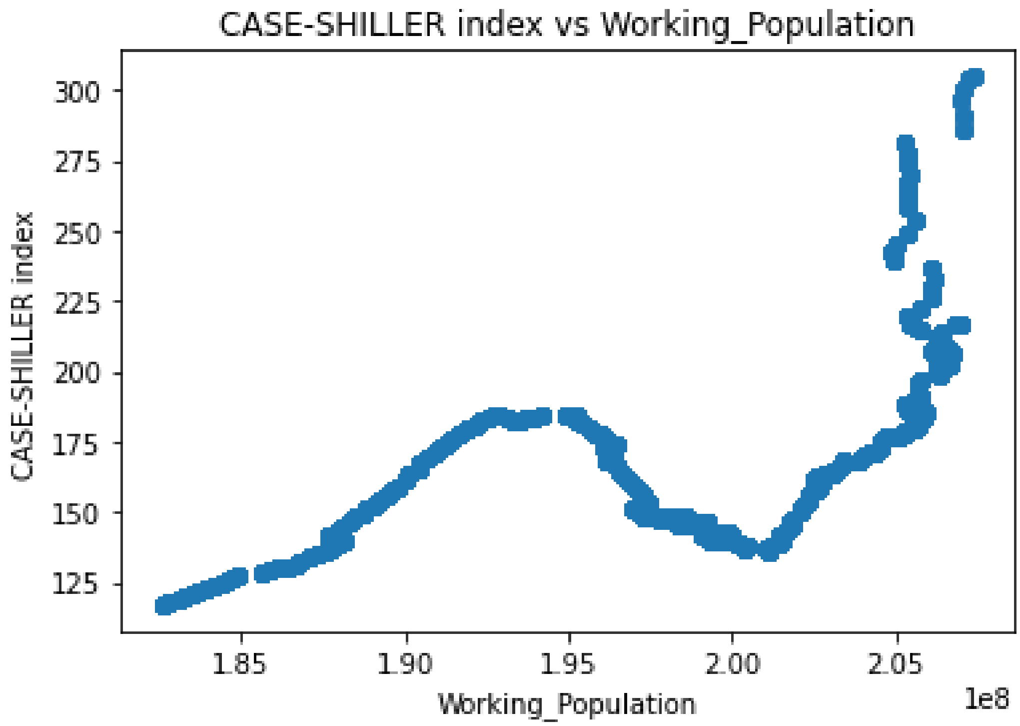
[157]:
```
# Separating the target variable and the independent variable
y = df.pop("CSUSHPISA")
X = df
```

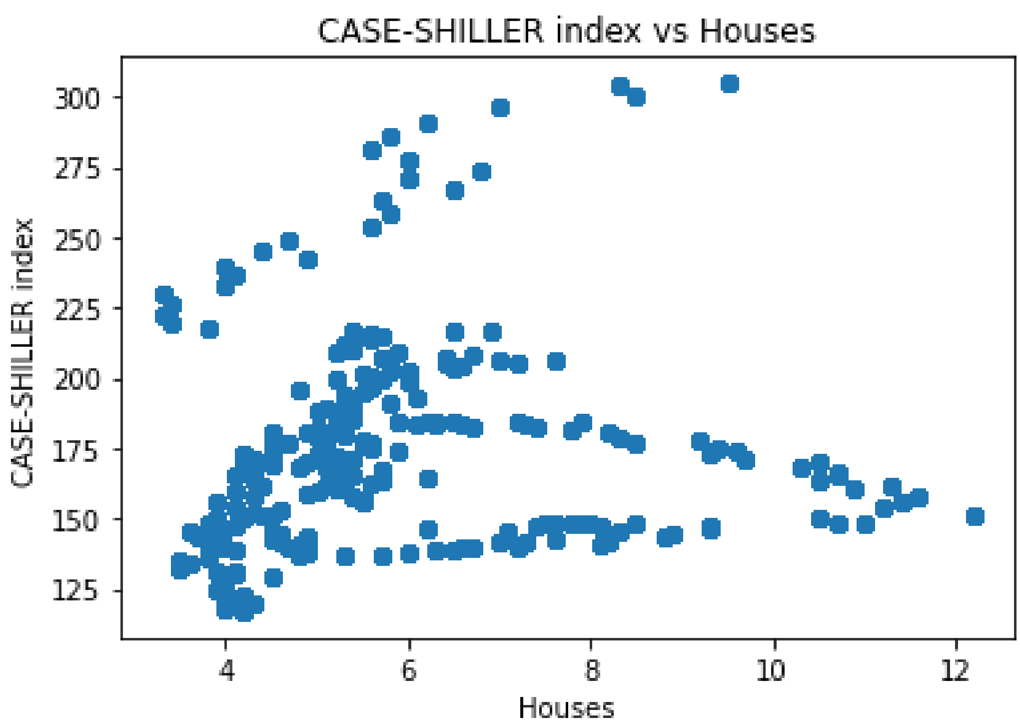
[158]:
```
# Plotting scatter plots of the CASE-SHILLER index vs features

for feature in X.columns:
    plt.figure()
    plt.scatter(x = X[feature], y = y)
    plt.xlabel(feature)
    plt.ylabel("CASE-SHILLER index")
    plt.title(f"CASE-SHILLER index vs {feature}")
```

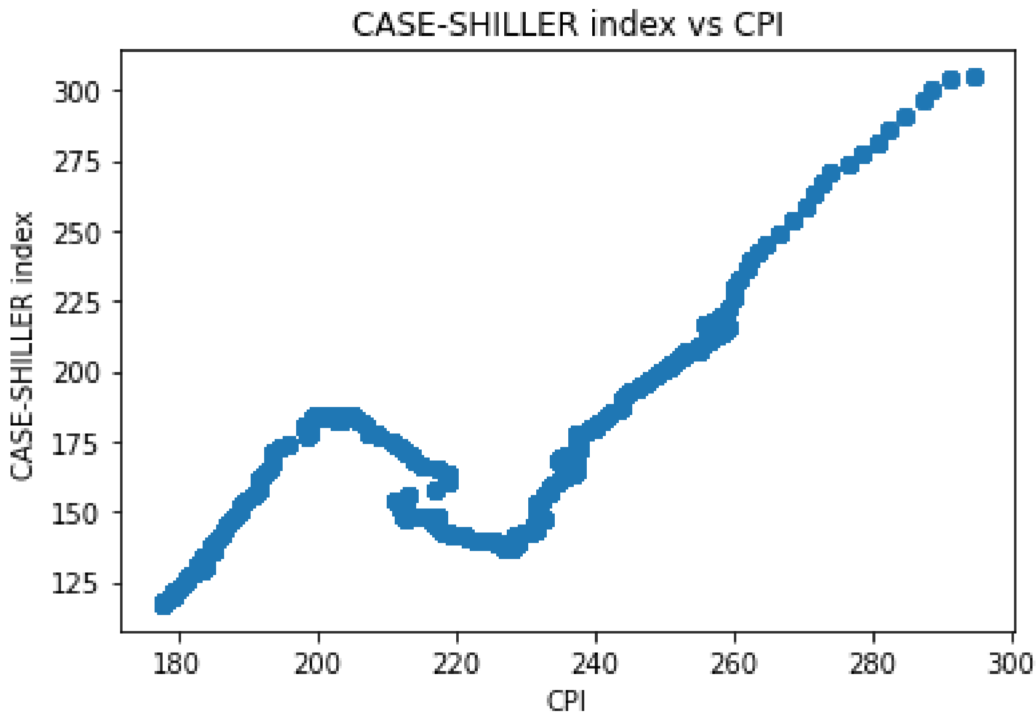


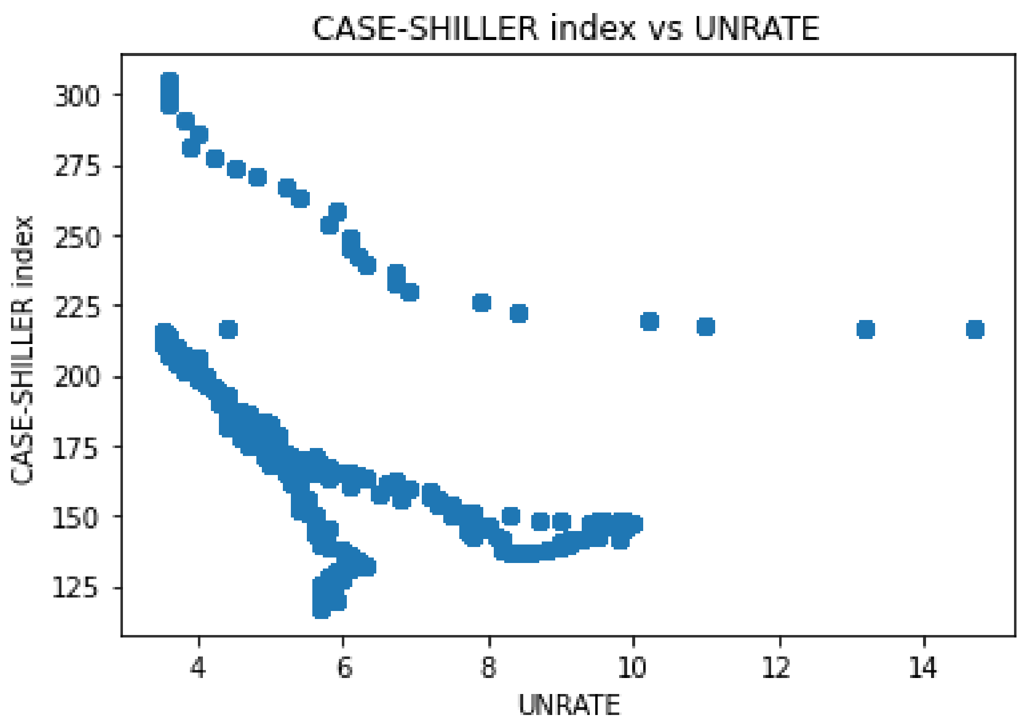

CASE-SHILLER index vs Month
CASE-SHILLER index
Month
125
150
175
200
225
250
275
300
2
4
6
8
10
12

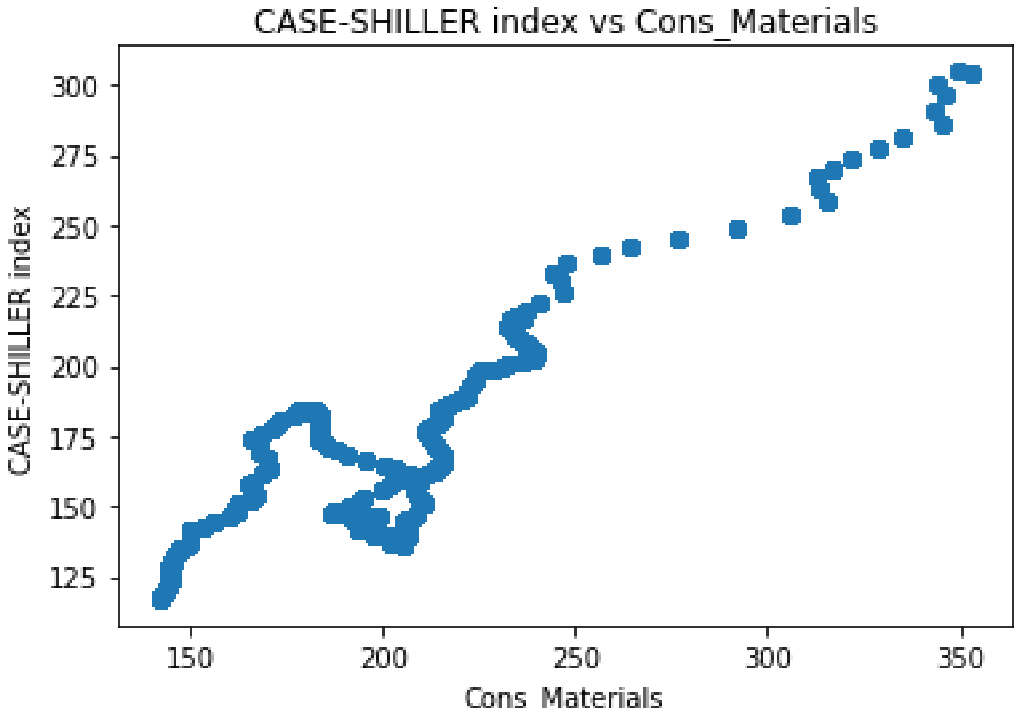

CASE-SHILLER index vs Per_Capita_GDP
CASE-SHILLER index
Per_Capita_GDP
125
150
175
200
225
250
275
300
50000
52000
54000
56000
58000
60000
62000
64000
66000

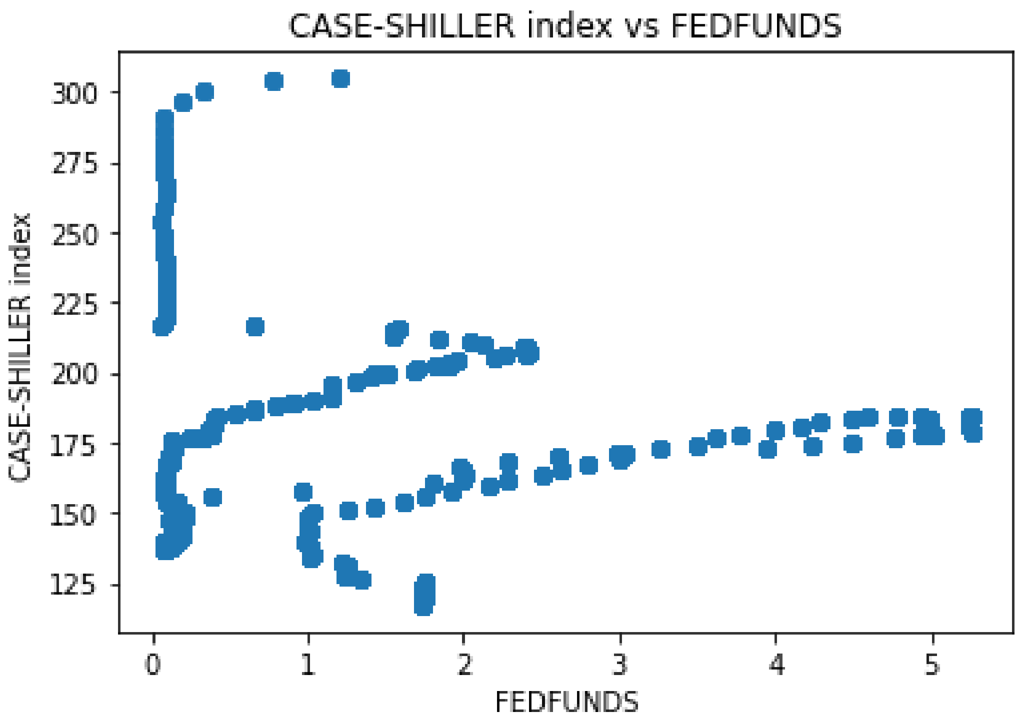

CASE-SHILLER index vs Working_Population
CASE-SHILLER index
300
275
250
225
200
175
150
125
1.85
1.90
1.95
2.00
2.05
1e8
Working_Population

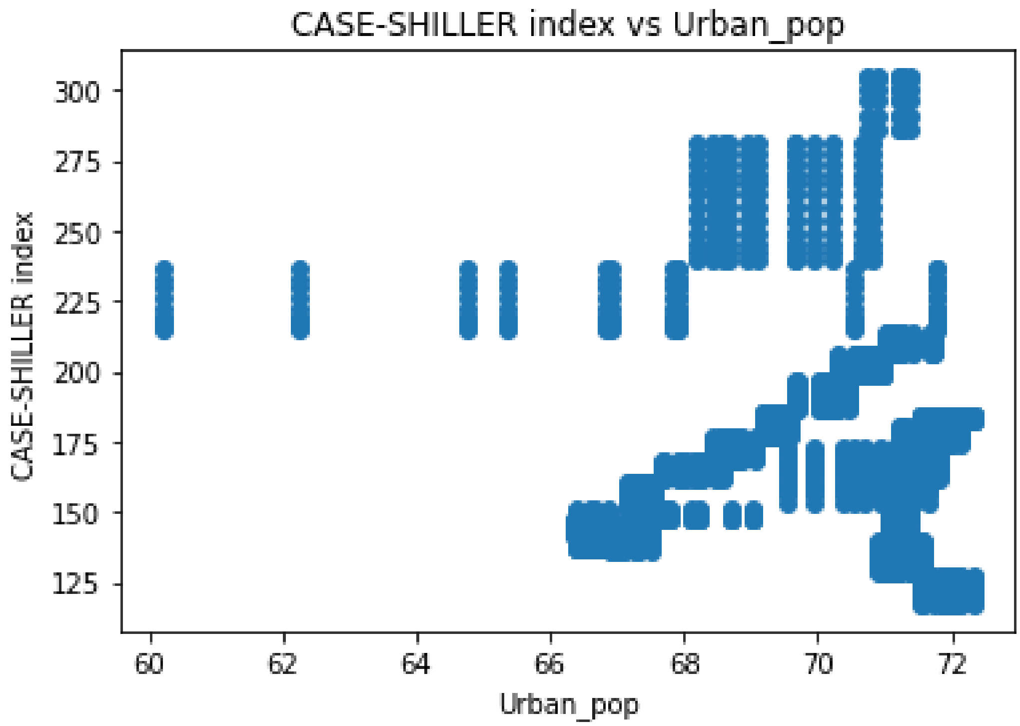

CASE-SHILLER index vs Houses
CASE-SHILLER index
300
275
250
225
200
175
150
125
4
6
8
10
12
Houses

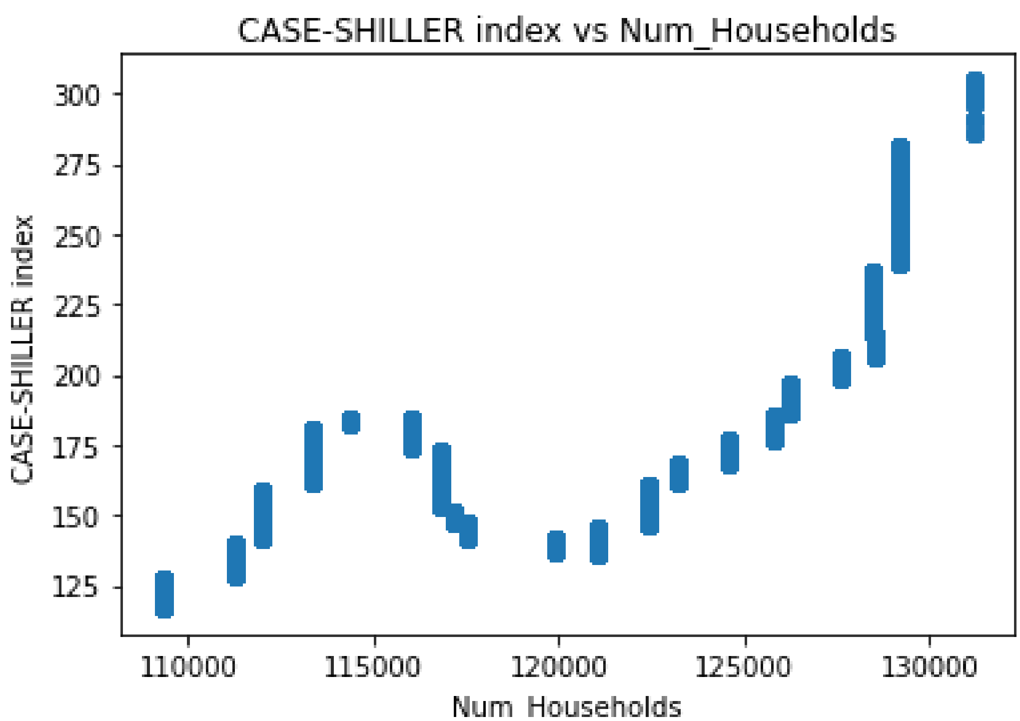

CASE-SHILLER index vs CPI
CASE-SHILLER index
300
275
250
225
200
175
150
125
180
200
220
240
260
280
300
CPI

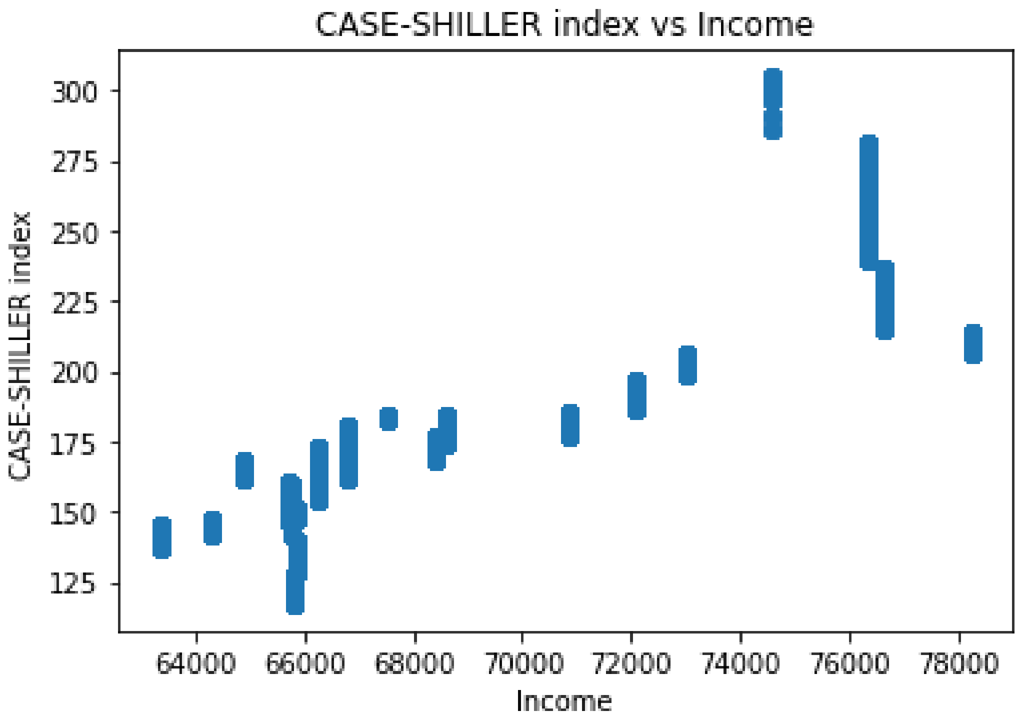

CASE-SHILLER index vs UNRATE
CASE-SHILLER index
300
275
250
225
200
175
150
125
4
6
8
10
12
14
UNRATE

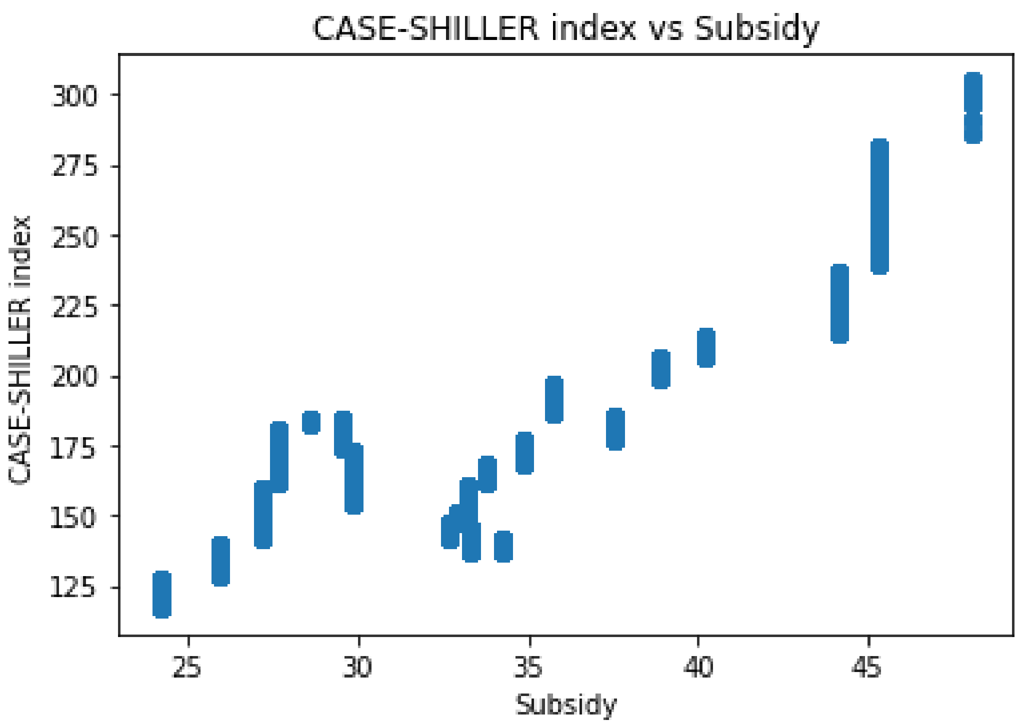

CASE-SHILLER index vs Cons_Materials
CASE-SHILLER index
Cons_Materials
125
150
175
200
225
250
275
300
150
200
250
300
350

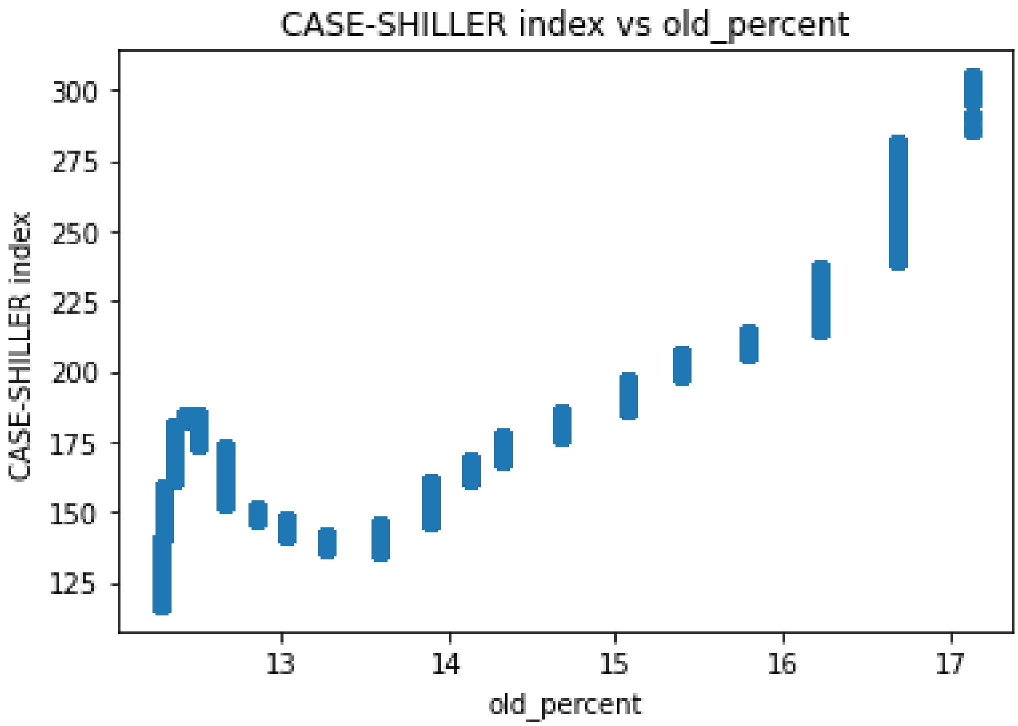

CASE-SHILLER index vs FEDFUNDS
CASE-SHILLER index
FEDFUNDS
125
150
175
200
225
250
275
300
0
1
2
3
4
5


CASE-SHILLER index vs Urban_pop
CASE-SHILLER index
300
275
250
225
200
175
150
125
60
62
64
66
68
70
72
Urban_pop


CASE-SHILLER index vs Num_Households
CASE-SHILLER index
300
275
250
225
200
175
150
125
110000
115000
120000
125000
130000
Num_Households


CASE-SHILLER index vs Income
CASE-SHILLER index
300
275
250
225
200
175
150
125
64000
66000
68000
70000
72000
74000
76000
78000
Income


CASE-SHILLER index vs Subsidy
CASE-SHILLER index
300
275
250
225
200
175
150
125
25
30
35
40
45
Subsidy



Calculate correlation coefficients

[159]:
```
correlations = X.apply(lambda column: np.abs(column.corr(y)))

# Sort correlations in ascending order
sorted_correlations = correlations.sort_values()

# Display features with lower correlation
print("Features with Lower Correlation to Target:")
print(sorted_correlations)
```

```
Features with Lower Correlation to Target:
Month                 0.030893
Houses                0.034838
FEDFUNDS              0.072644
Urban_pop             0.121200
UNRATE                0.429101
Working_Population    0.612680
Num_Households        0.729666
Year                  0.748704
CPI                   0.772845
Subsidy               0.809054
old_percent           0.816216
Cons_Materials        0.855816
```

```
Income                  0.860906
Per_Capita_GDP          0.895990
dtype: float64
```

Based on the provided correlation coefficients:

Highest Correlation:

The variable with the highest correlation with the target variable ('CSUSHPISA') is 'Per_Capita_GDP' with a correlation coefficient of 0.895990. This feature shows a strong positive linear relationship with home prices.

Other Strong Correlations:

'Cons_Material' (0.855), 'Subsidy' (0.809), 'old_percent' (0.816), 'income' (0.860), and 'CPI' (0.772) also have strong positive correlations.

Moderate Correlations:

'year' (0.748), 'Num_Households' (0.729), and 'working_age_pop' (0.612) have moderate positive correlations.

Lower Correlations:

'UNRATE' (0.429), 'EmpRate' (0.121), 'Houses' (0.034), and 'FEDFUNDS' (0.072) have lower correlations.

Data science models

[160]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, ElasticNet
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.feature_selection import SelectFromModel
from xgboost import XGBRegressor
```

We will drop the columns which has lower correlation with the target

[161]:
```
mult_cols = ["Working_Population", "Houses", "Urban_pop", "Num_Households",␣
 ↪"UNRATE",  "FEDFUNDS"]
us_house_price_df.drop(columns = mult_cols, inplace = True)
X = us_house_price_df
```

[163]:
```
mult_col = ["CSUSHPISA"]
us_house_price_df.drop(columns = mult_col, inplace = True)
X = us_house_price_df
```

[164]:
```
X
```

[164]:
```
            Per_Capita_GDP      CPI  Cons_Materials  Income  Subsidy  \
DATE
2002-01-01         50091.0  177.700           142.0   65820   24.183
2002-02-01         50091.0  177.700           142.0   65820   24.183
```

```
2002-03-01          50091.0  177.700          142.0   65820   24.183
2002-04-01          50091.0  177.700          142.0   65820   24.183
2002-05-01          50091.0  177.700          142.0   65820   24.183
…                      …      …         …       …       …
NaN                 65127.0  294.728          349.8   74580   48.021
NaN                 65127.0  294.728          349.8   74580   48.021
NaN                 65127.0  294.728          349.8   74580   48.021
NaN                 65127.0  294.728          349.8   74580   48.021
NaN                 65127.0  294.728          349.8   74580   48.021

            old_percent
DATE
2002-01-01    12.287458
2002-02-01    12.287458
2002-03-01    12.287458
2002-04-01    12.287458
2002-05-01    12.287458
…                  …
NaN           17.128121
NaN           17.128121
NaN           17.128121
NaN           17.128121
NaN           17.128121

[2916 rows x 6 columns]
```

Models building

[165]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[166]:
```
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Explored various regression models, including Linear Regression, ElasticNet, Random Forest, Gradient Boosting, Support Vector Regression (SVR), and XGBoost.

[167]:
```
# Models
models = {
    'Linear Regression': LinearRegression(),
    'ElasticNet': ElasticNet(),
    'Random Forest': RandomForestRegressor(),
    'Gradient Boosting': GradientBoostingRegressor(),
    'SVR': SVR(),
    'XGBoost': XGBRegressor()
```

```
}
```

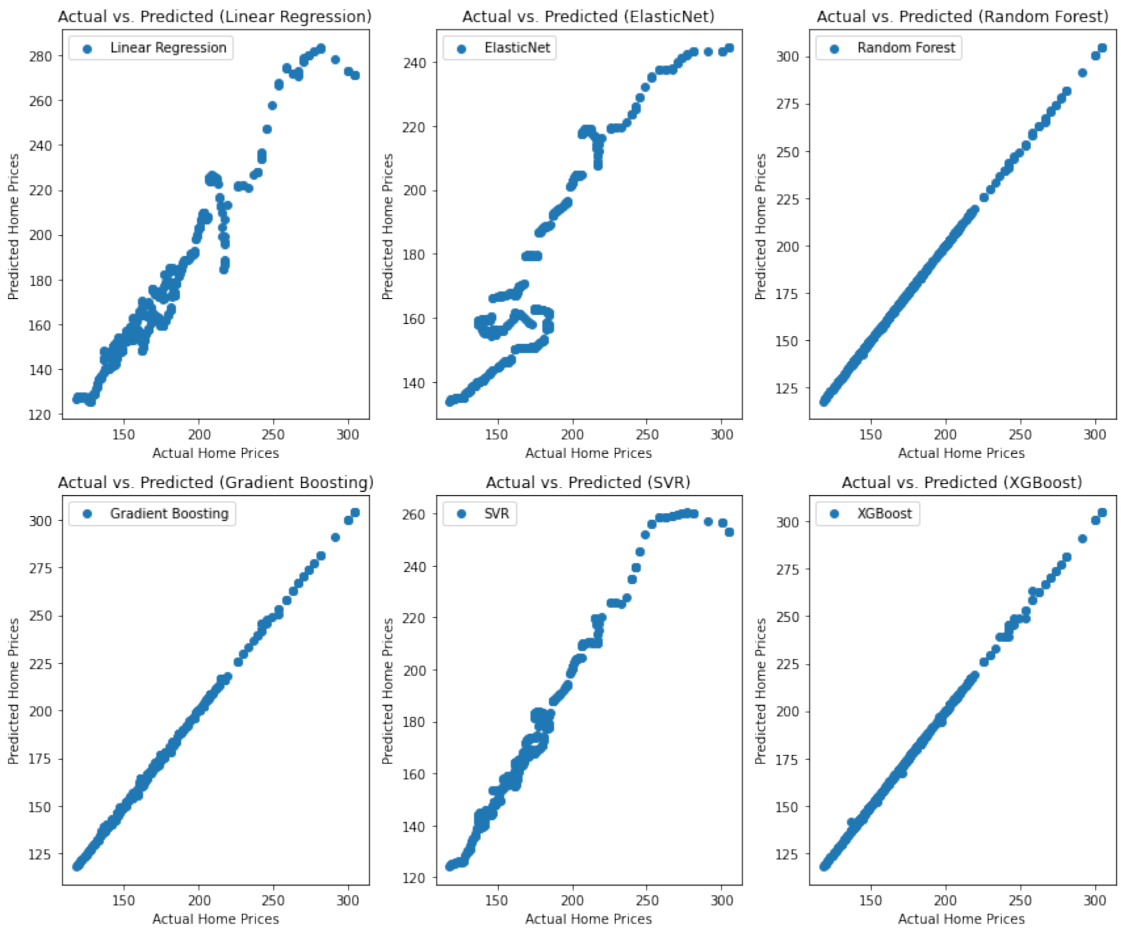
[168]:

```
best_model = None
best_mse = float('inf')

# Visualize actual vs. predicted values for all models
fig, axs = plt.subplots(2, 3, figsize=(12, 10))
axs = axs.flatten()

# Training and evaluation
for i, (name, model) in enumerate(models.items()):
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Model: {name}")
    print(f"Mean Squared Error: {mse}")
    print(f"R-squared: {r2}")

    # Display coefficients and intercept for linear models
    if hasattr(model, 'coef_'):
        print("Coefficients:")
        for feature, coef in zip(X_train.columns, model.coef_):
            print(f"{feature}: {coef}")

        print(f"Intercept: {model.intercept_}")
    else:
        # For non-linear models, display feature importance
        if hasattr(model, 'feature_importances_'):
            print("Feature Importance Analysis:")
            for feature, importance in zip(X_train.columns, model.
 ↪feature_importances_):
                print(f"{feature}: {importance}")

    print()

    # Update best model if current model has lower MSE
    if mse < best_mse:
        best_mse = mse
        best_model = model

    # Plot actual vs. predicted values
    axs[i].scatter(y_test, y_pred, label=name)
    axs[i].set_xlabel("Actual Home Prices")
    axs[i].set_ylabel("Predicted Home Prices")
```

```python
    axs[i].set_title(f"Actual vs. Predicted ({name})")
    axs[i].legend()

# Tight layout for better spacing
plt.tight_layout()
plt.show()

print(f"\nBest Model: {type(best_model).__name__} with MSE: {best_mse}")
```

```
Model: Linear Regression
Mean Squared Error: 74.18452882567686
R-squared: 0.9451223045627271
Coefficients:
Per_Capita_GDP: 25.195084141906495
CPI: -29.80554396241202
Cons_Materials: 30.426026363980412
Income: 15.585526032060551
Subsidy: 10.012732712039767
old_percent: -16.530491384178426
Intercept: 173.04170411663813

Model: ElasticNet
Mean Squared Error: 236.08466961827202
R-squared: 0.8253573514342188
Coefficients:
Per_Capita_GDP: 7.189679460588254
CPI: 1.10770259546806
Cons_Materials: 7.393487551522559
Income: 9.508278763748645
Subsidy: 2.827837275944486
old_percent: 2.1648706326975447
Intercept: 173.04170411663807

Model: Random Forest
Mean Squared Error: 0.042414382393332628
R-squared: 0.9999686241377281
Feature Importance Analysis:
Per_Capita_GDP: 0.07217230563800532
CPI: 0.39311869485581263
Cons_Materials: 0.3172877694273253
Income: 0.16605733301024092
Subsidy: 0.027379924158911814
old_percent: 0.023983972909703985

Model: Gradient Boosting
Mean Squared Error: 0.8136942960218423
R-squared: 0.9993980730421425
```

```
Feature Importance Analysis:
Per_Capita_GDP: 0.07377514701356544
CPI: 0.415706437558178
Cons_Materials: 0.29024999237356064
Income: 0.1891072770282319
Subsidy: 0.006044303156424081
old_percent: 0.025116842870039822

Model: SVR
Mean Squared Error: 45.93430224276296
R-squared: 0.9660202917170851

Model: XGBoost
Mean Squared Error: 0.28790605085363796
R-squared: 0.9997870227010484
Feature Importance Analysis:
Per_Capita_GDP: 0.009511047974228859
CPI: 0.13317523896694183
Cons_Materials: 0.18667054176330566
Income: 0.6679704785346985
Subsidy: 0.0026727155782282352
old_percent: 0.0
```




```
Best Model: RandomForestRegressor with MSE: 0.04241438239332628
```

[169]:
```python
# Assuming you have a DataFrame with model names and their corresponding evaluation metrics
data = {
    'Model': ['Linear Regression', 'ElasticNet', 'Random Forest', 'Gradient Boosting', 'SVR', 'XGBoost'],
    'MSE': [84.65, 205.67, 2.21, 4.81, 477.15, 3.17],
    'R-squared': [0.93, 0.83, 0.998, 0.996, 0.61, 0.997]
}

df = pd.DataFrame(data)

# Create a table plot using matplotlib
fig, ax = plt.subplots(figsize=(6, 3))

# Hide the axes
```

```
ax.axis('off')

# Create a table and add data
table = ax.table(cellText=df.values, colLabels=df.columns, cellLoc = 'center', loc='center', colColours=['#f3f3f3']*len(df.columns), colWidths=[0.25]*len(df.columns))

# Style the table
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1.2, 1.2)  # Adjust the table size if needed

# Highlight specific values (e.g., minimum and maximum MSE)
min_mse_index = np.argmin(df['MSE'])
max_mse_index = np.argmax(df['MSE'])

for i in range(len(df)):
    for j in range(len(df.columns)):
        if (i == min_mse_index or i == max_mse_index) and j == df.columns.get_loc('MSE'):
            color = '#ffcccb' if i == min_mse_index else '#b0e57c'  # Light red for min and light green for max MSE
            table[(i + 1, j)].set_facecolor(color)

# Highlight min and max R-squared
min_r2_index = np.argmin(df['R-squared'])
max_r2_index = np.argmax(df['R-squared'])

for i in range(len(df)):
    for j in range(len(df.columns)):
        if (i == min_r2_index or i == max_r2_index) and j == df.columns.get_loc('R-squared'):
            color = '#ffcccb' if i == min_r2_index else '#b0e57c'  # Light red for min and light green for max R-squared
            table[(i + 1, j)].set_facecolor(color)

plt.title('Model Evaluation Metrics')
plt.show()
```

Model Evaluation Metrics

Model	MSE	R-squared
Linear Regression	84.65	0.93
ElasticNet	205.67	0.83
Random Forest	2.21	0.998
Gradient Boosting	4.81	0.996
SVR	477.15	0.61
XGBoost	3.17	0.997

Decision:

Random Forest and XGBoost appear to be strong contenders, as they have low MSE and high R-squared values. Additionally, both models provide insights into feature importance.

Gradient Boosting also performs well but with a slightly higher MSE compared to Random Forest and XGBoost.

Linear Regression and ElasticNet have higher MSE values, indicating potential limitations in predictive accuracy.

SVR has a considerably higher MSE and lower R-squared, suggesting lower performance compared to other models.

Champion model: In summary, based on the provided metrics, Random Forest appear to be a strong candidate for the best model, with a low MSE and a high R-squared value.

Low MSE: The low MSE indicates that the model's predictions are close to the actual values on average, suggesting good predictive accuracy.

High R-squared: The high R-squared value suggests that a significant portion of the variance in home prices is explained by the model. This indicates strong explanatory power.

Feature Importance: The feature importance analysis provides transparency into the factors driving the predictions. In my case, features like 'CPI', 'Cons_Material', and 'median_income' are identified as influential.

This means that, according to the model, changes in these features have a notable impact on the predictions of U.S. home prices. For example, if 'CPI' increases, it suggests that changes in the cost of living might influence home prices.

[ ]:

```

```