In [258]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import plotly.graph_objects as go
import plotly.express as px
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
import statsmodels.api as sm
```

In [134]:
```
file = pd.read_csv(r"E:\assignment\final\final19.csv")
file['Date'] = pd.to_datetime(file['Date'])

file['year'] = file['Date'].dt.year
file['month'] = file['Date'].dt.month
```

In [163]:
```
file1 = file.drop(['Date'], axis=1)
```

In [256]:
```
file1
```

Out[256]:

	House_Price_Index	Interest_Rate	No_of_property_Introduced	PPI	Income	Unemployment	Working_Pop	CPI	popul
0	128.461	1.24	1654	144.400	10710.4	5.8	185635346.4	182.600	29010
1	129.355	1.26	1688	145.200	10674.0	5.9	185869692.3	183.600	29010
2	130.148	1.25	1638	145.200	10696.5	5.9	186085118.2	183.900	29010
3	130.884	1.26	1662	145.900	10752.7	6.0	186470754.0	183.200	29010
4	131.735	1.26	1733	145.800	10832.0	6.1	186649078.0	182.900	29010
...	...	...	...	...	...	...	...	...	
235	301.473	2.33	1355	342.753	16161.4	3.7	207370651.0	295.320	33328
236	299.353	2.56	1438	336.464	16184.9	3.5	207453580.5	296.539	33328
237	298.873	3.08	1348	333.796	16223.5	3.7	207431164.7	297.987	33328
238	298.269	3.78	1543	330.369	16229.6	3.6	207521914.2	298.598	33328
239	297.413	4.10	1390	326.449	16265.1	3.5	207524486.3	298.990	33328

240 rows × 27 columns

In [ ]:
```

```

Statical Tests

In [263]:
```
file1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   House_Price_Index           240 non-null    float64
 1   Interest_Rate               240 non-null    float64
 2   No_of_property_Introduced   240 non-null    int64  
 3   PPI                         240 non-null    float64
 4   Income                      240 non-null    float64
 5   Unemployment                240 non-null    float64
 6   Working_Pop                 240 non-null    float64
 7   CPI                         240 non-null    float64
 8   population                  240 non-null    int64  
 9   subsidy                     240 non-null    float64
 10  Inflation                   240 non-null    float64
 11  Annual Change               240 non-null    float64
 12  GDP                         240 non-null    int64  
 13  Mortgage_Rate               240 non-null    float64
 14  Employment Rate             240 non-null    float64
 15  ER_MALE                     240 non-null    float64
 16  ER_FEMALE                   240 non-null    float64
 17  Labour_Participation_Rate   240 non-null    float64
 18  House_Supply                240 non-null    float64
 19  personal_saving_rate        240 non-null    float64
 20  property tax                240 non-null    int64  
 21  Real_m2                     240 non-null    float64
 22  Velocity_m2                 240 non-null    float64
 23  Urban_cpi                   240 non-null    float64
 24  Government_Exp              240 non-null    float64
 25  year                        240 non-null    int64  
 26  month                       240 non-null    int64  
dtypes: float64(21), int64(6)
memory usage: 50.8 KB
```

In [356]:
```
file1.describe()
```

Out[356]:

	House_Price_Index	Interest_Rate	No_of_property_Introduced	PPI	Income	Unemployment	Working_Pop	
count	240.000000	240.000000	240.000000	240.000000	240.000000	240.000000	2.400000e+02	240.
mean	180.657238	1.301625	1201.716667	212.851467	13493.480833	6.012083	2.001512e+08	230.
std	41.254187	1.578512	423.857641	44.566715	1837.485256	2.034118	6.171480e+06	27.
min	128.461000	0.050000	520.000000	144.400000	10674.000000	3.500000	1.856353e+08	182.
25%	148.123000	0.120000	839.000000	184.275000	12115.575000	4.575000	1.962127e+08	211.
50%	172.495000	0.400000	1181.500000	207.450000	12999.350000	5.400000	2.018248e+08	231.
75%	197.457750	2.000000	1440.000000	224.525000	14820.075000	7.350000	2.055396e+08	248.
max	304.724000	5.260000	2245.000000	353.015000	20422.600000	14.700000	2.075245e+08	298.

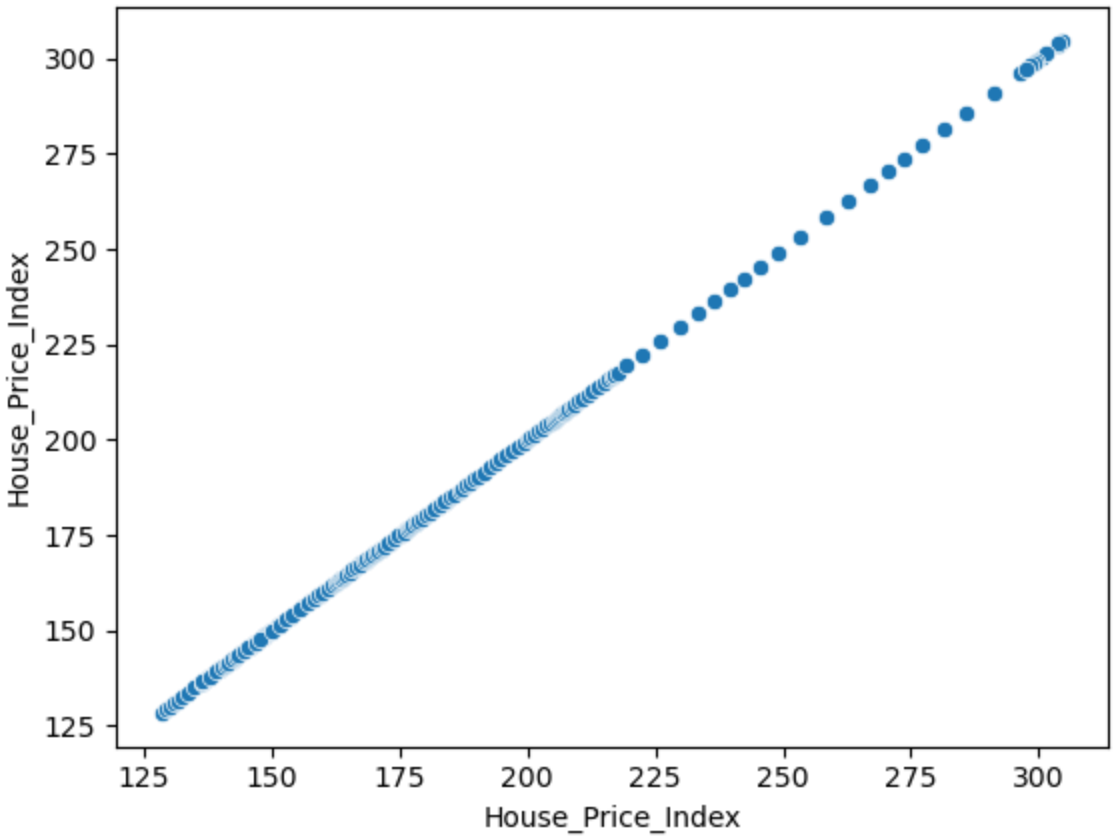
8 rows × 27 columns

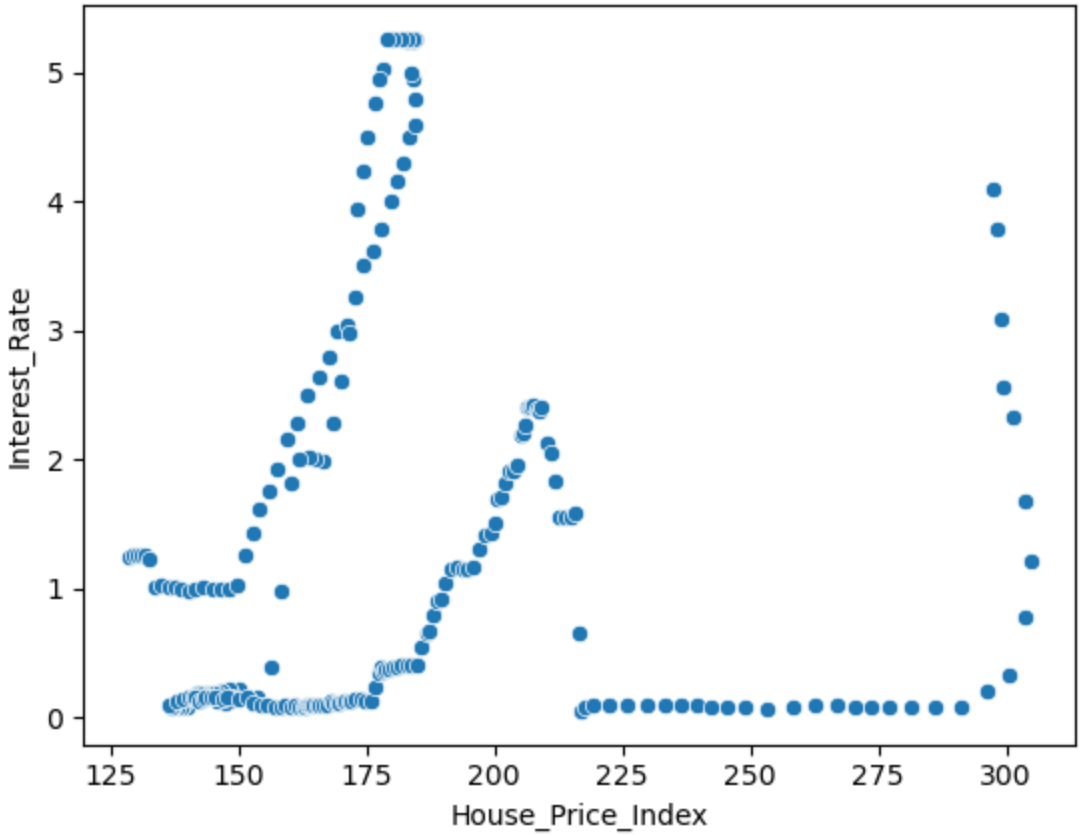
In [ ]:
```

```

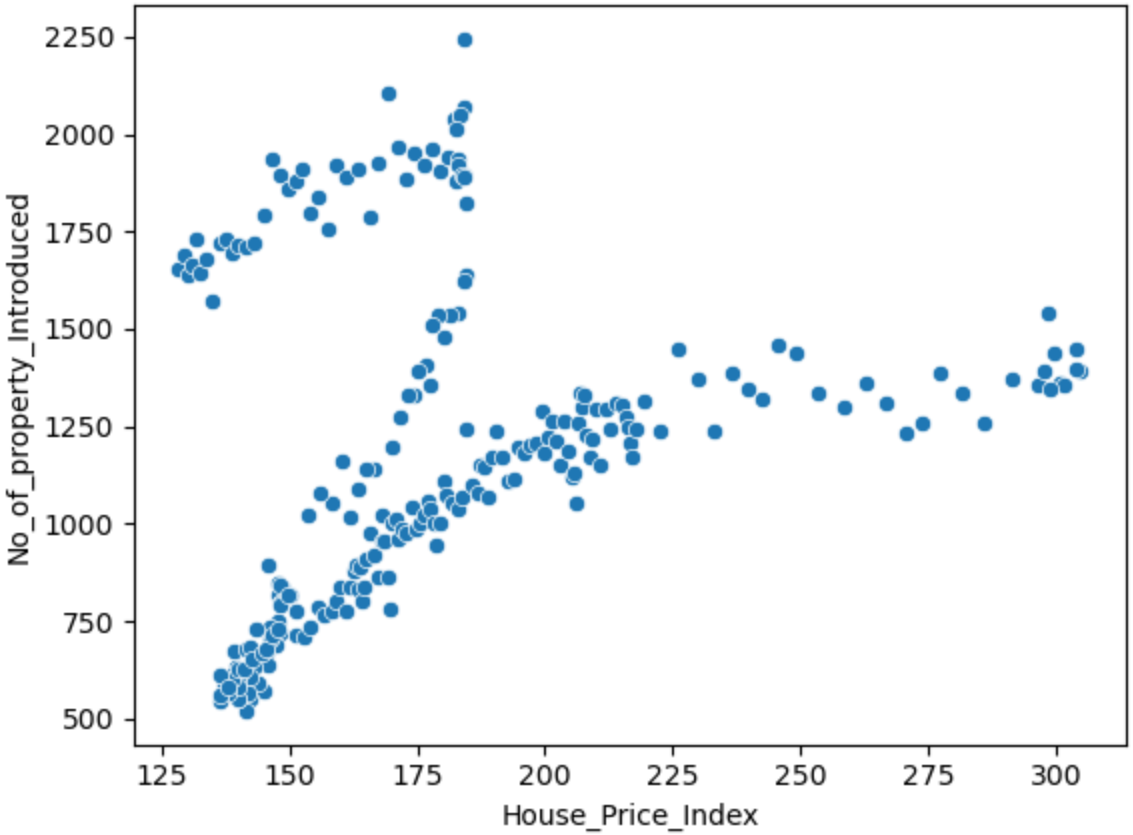
Assumption check

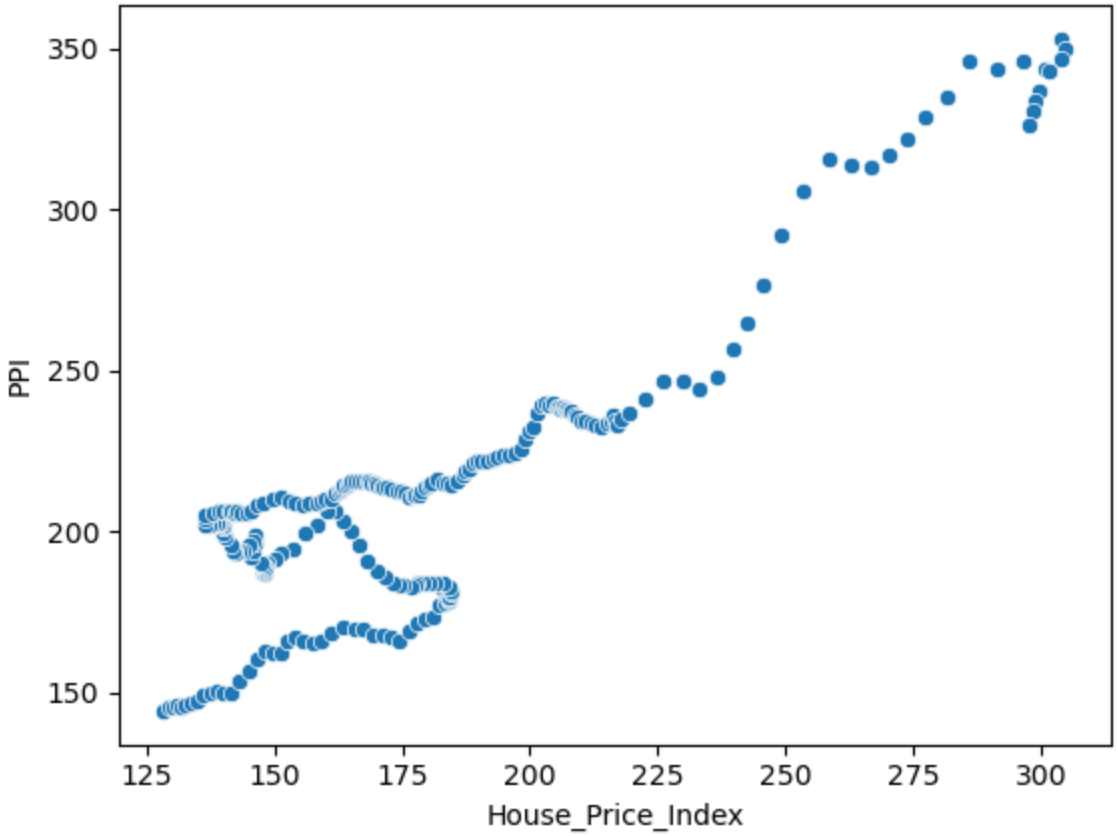
Check Whether is a Linear relationship between input and output column

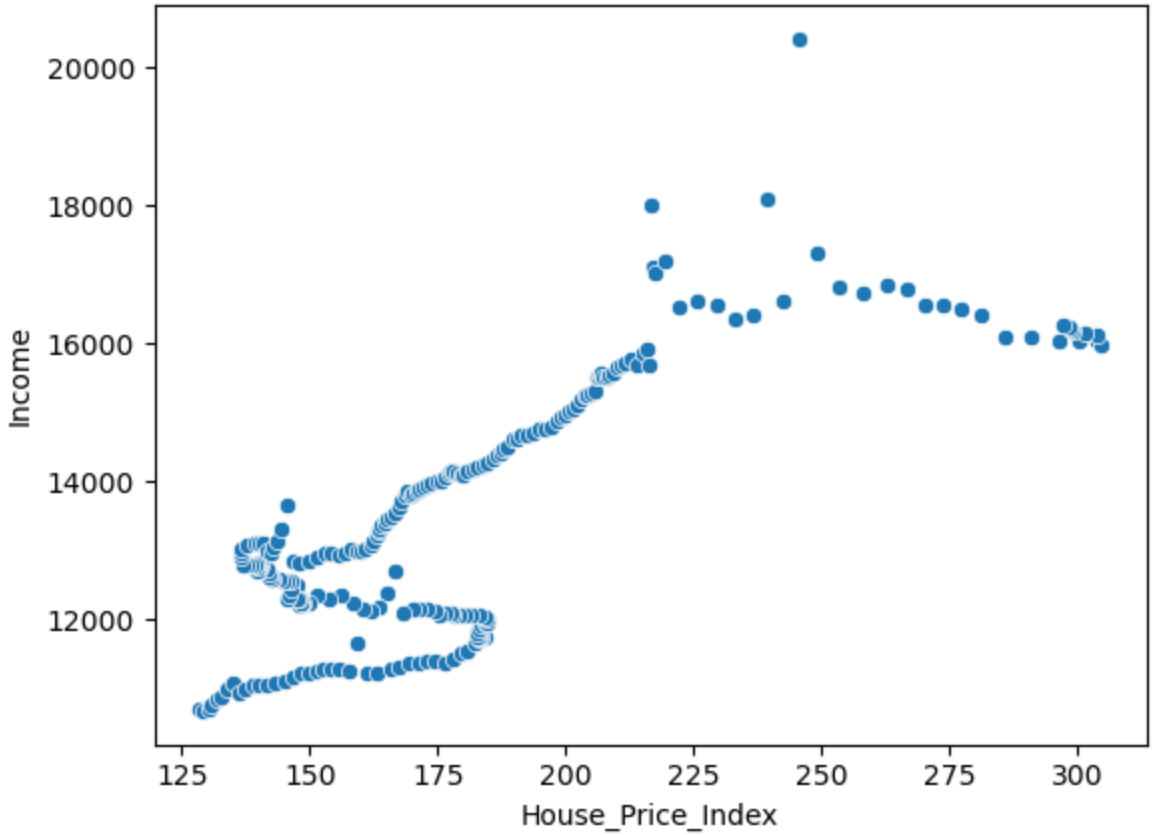
In [165]:
```
for i in file1:
    sns.scatterplot(data=file1, x='House_Price_Index', y=i)
    plt.show()
```

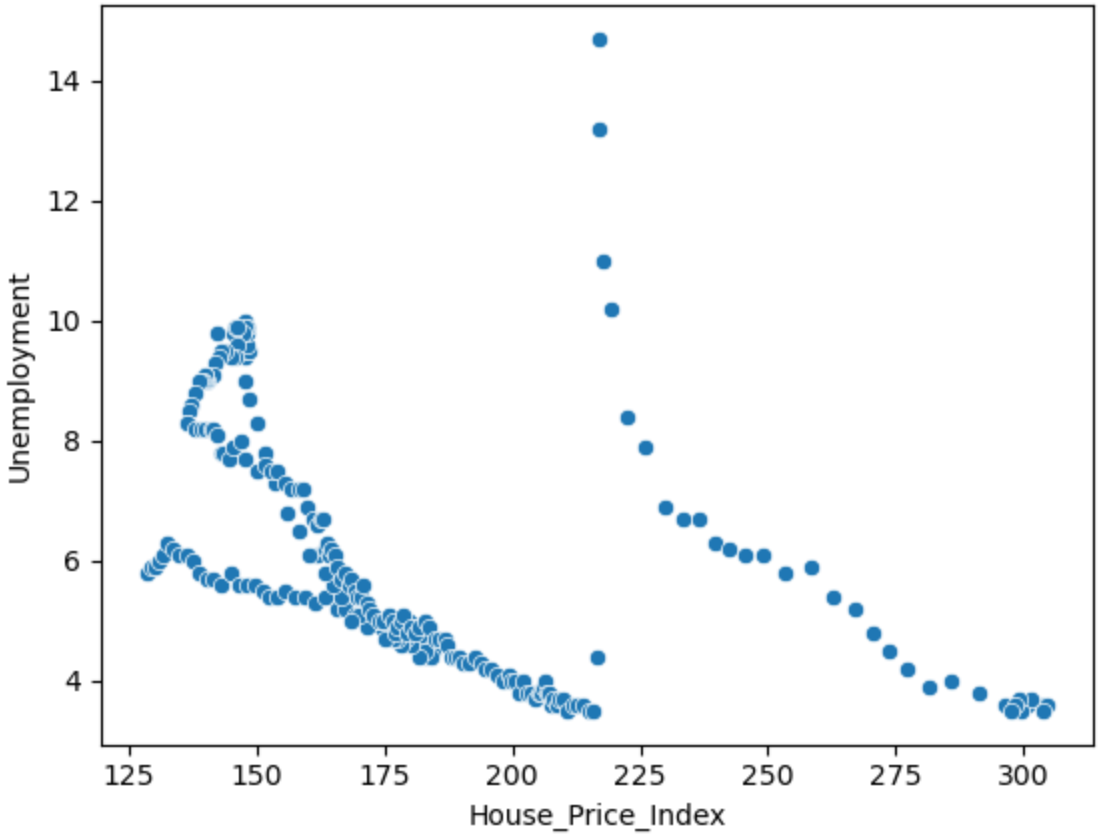


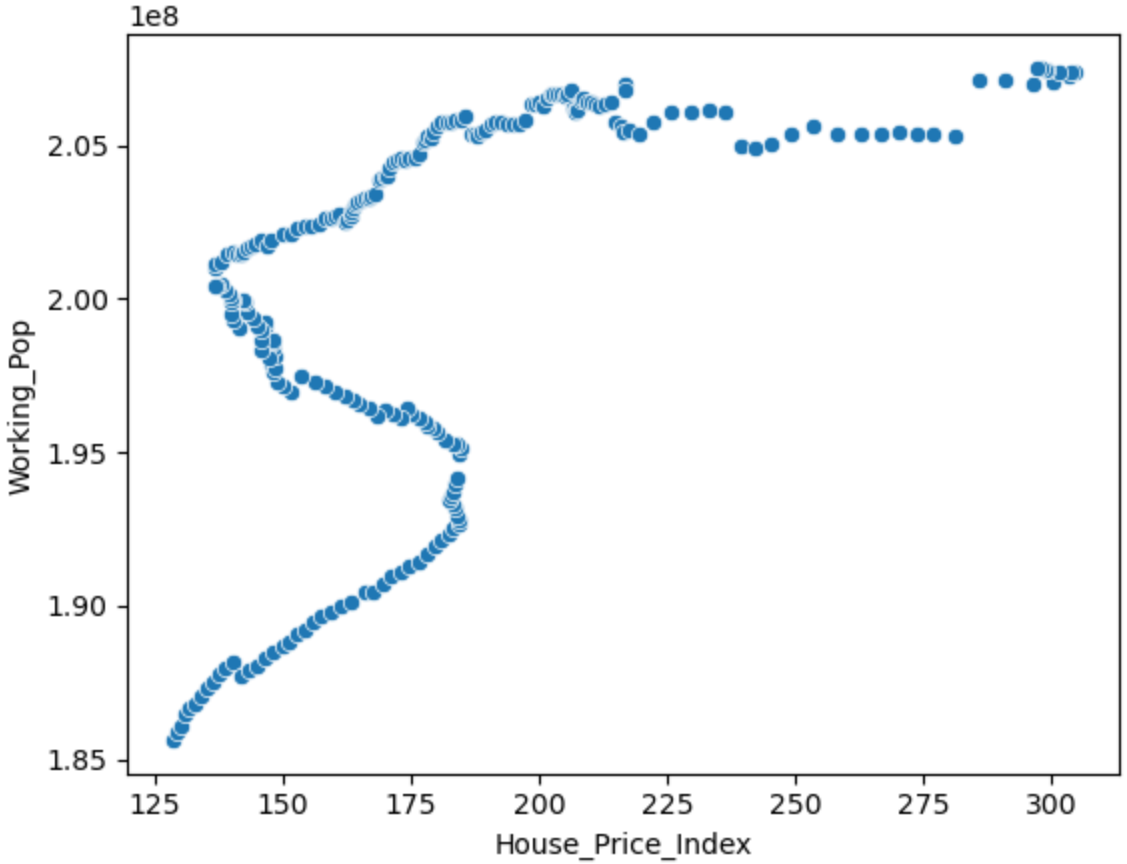

Interest_Rate
House_Price_Index
0
1
2
3
4
5
125
150
175
200
225
250
275
300

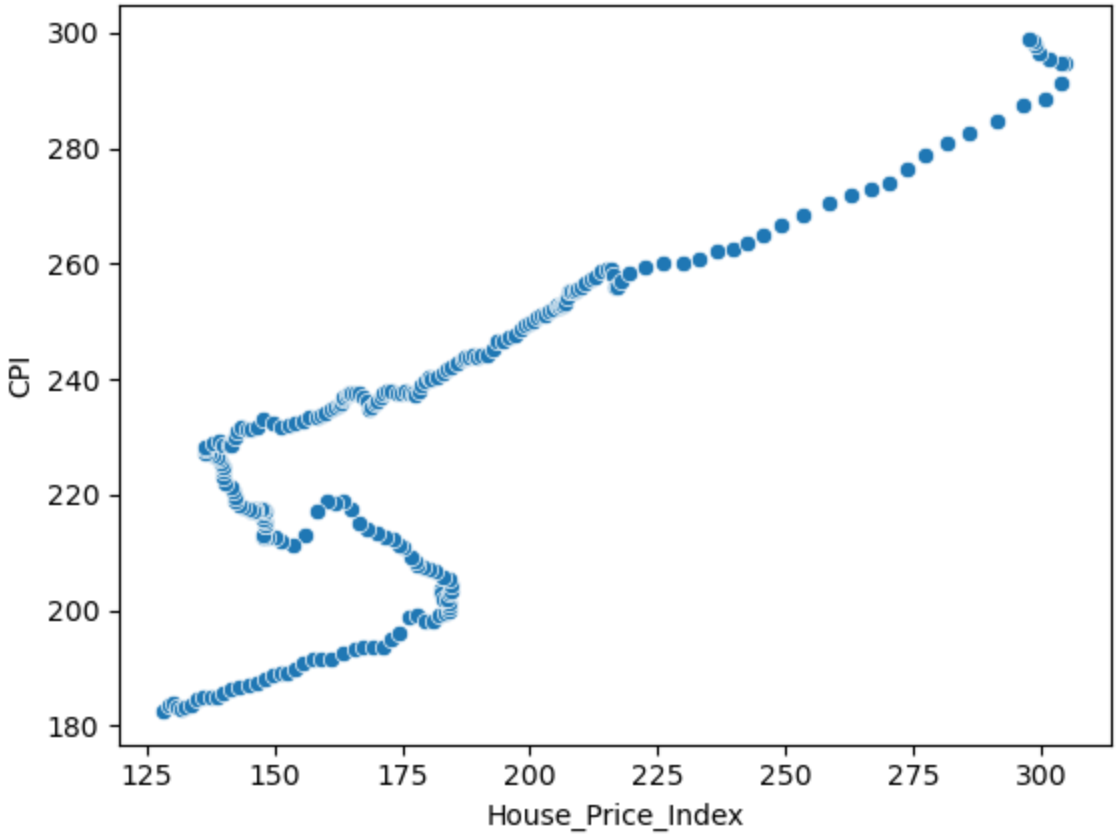

2250
2000
1750
1500
1250
1000
750
500
No_of_property_Introduced
125
150
175
200
225
250
275
300
House_Price_Index

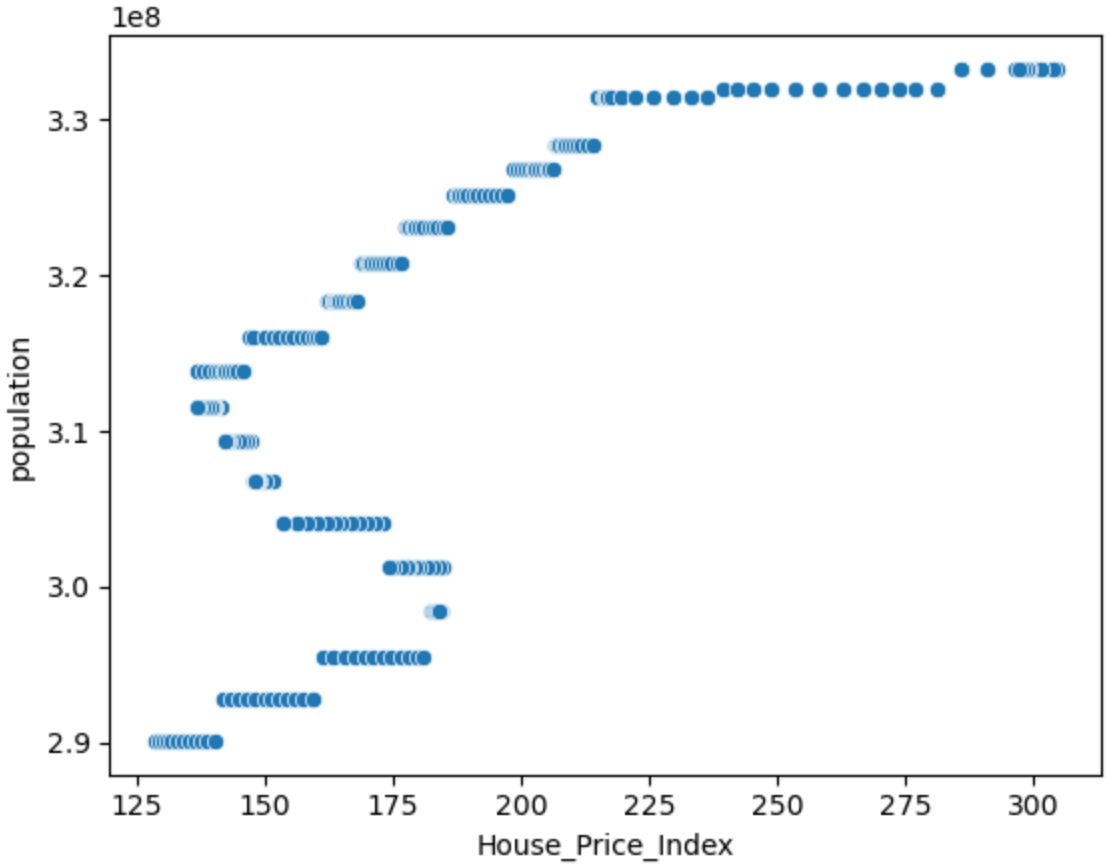

350
300
250
200
150
PPI
125
150
175
200
225
250
275
300
House_Price_Index

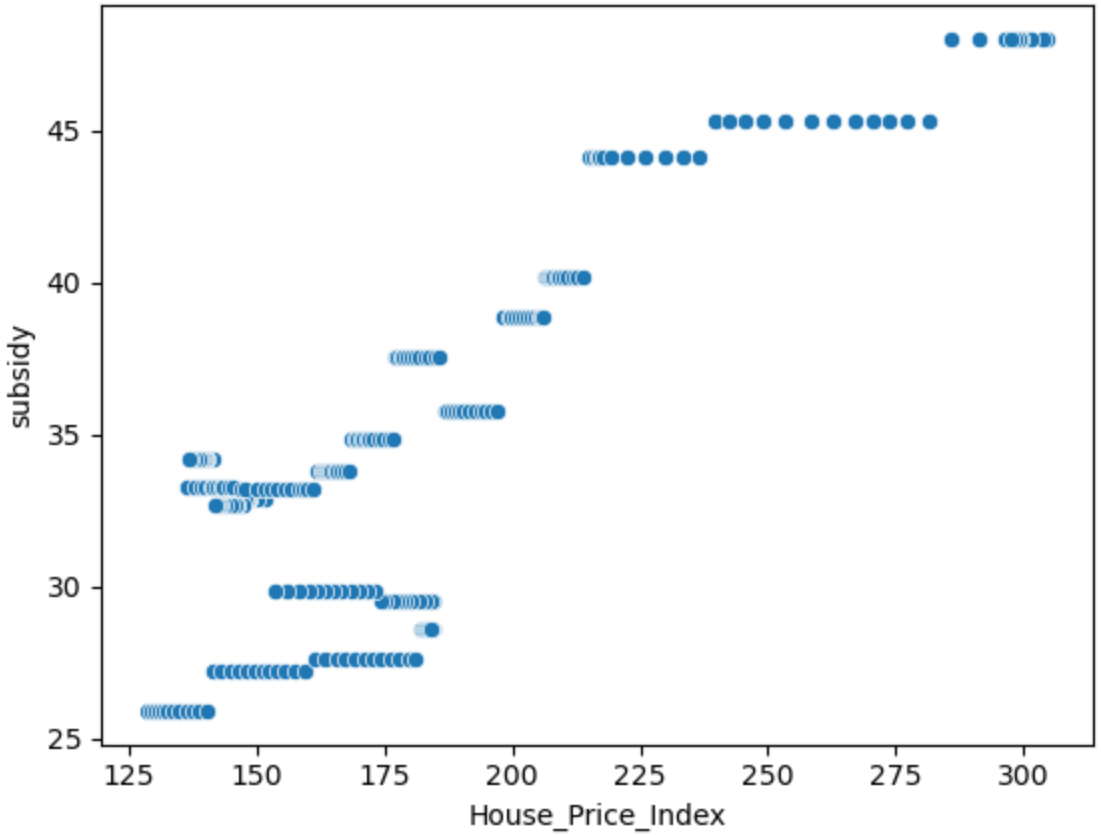

20000
18000
16000
14000
12000
Income
125
150
175
200
225
250
275
300
House_Price_Index

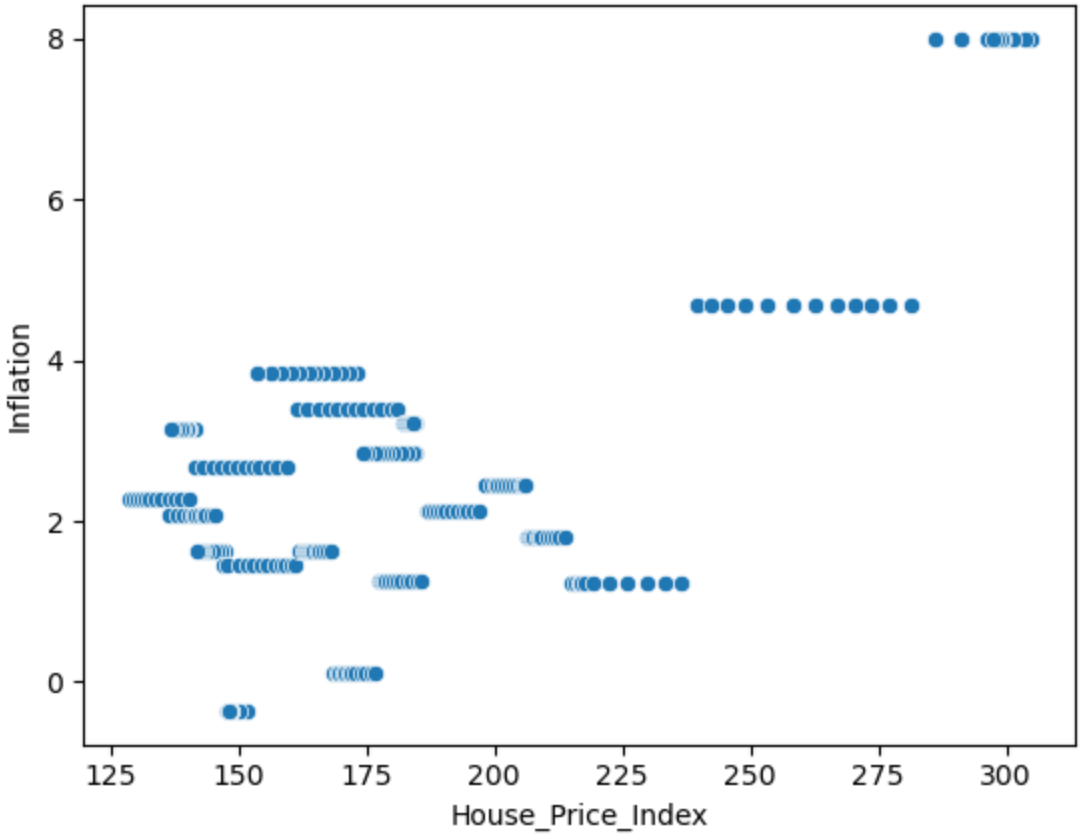

Unemployment
14
12
10
8
6
4
125
150
175
200
225
250
275
300
House_Price_Index

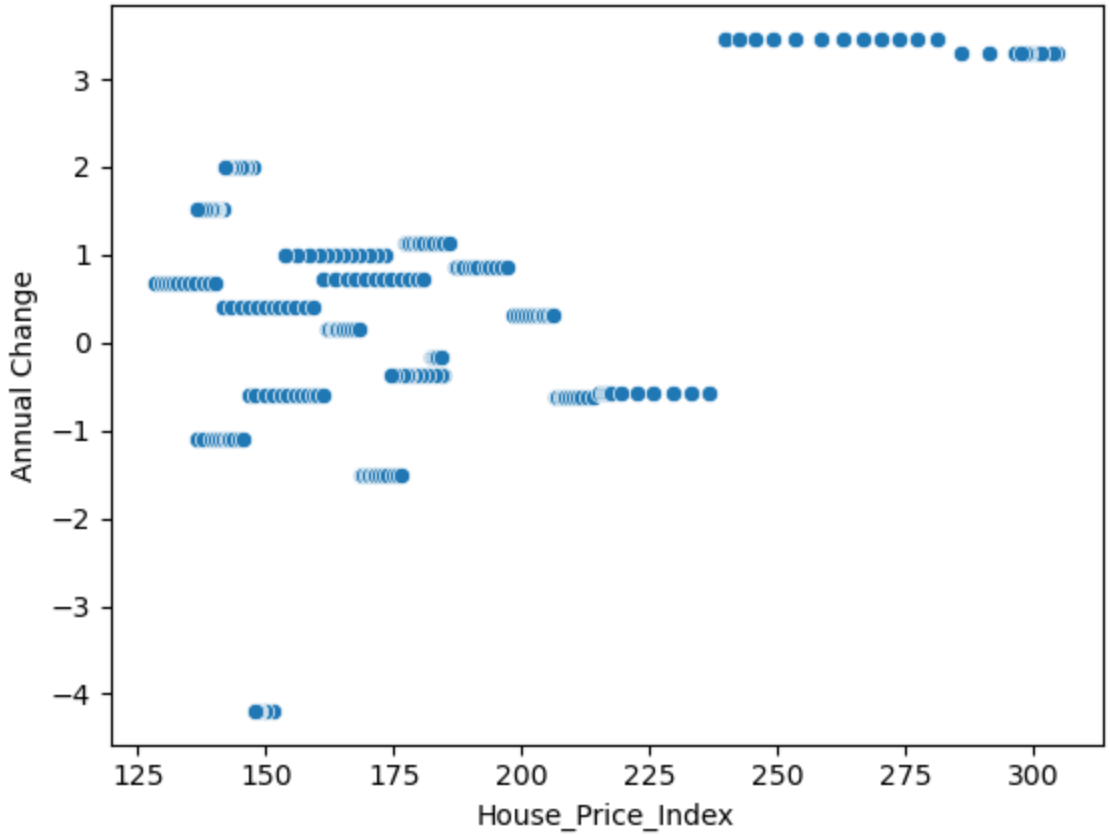

1e8
2.05
2.00
1.95
1.90
1.85
Working_Pop
125
150
175
200
225
250
275
300
House_Price_Index

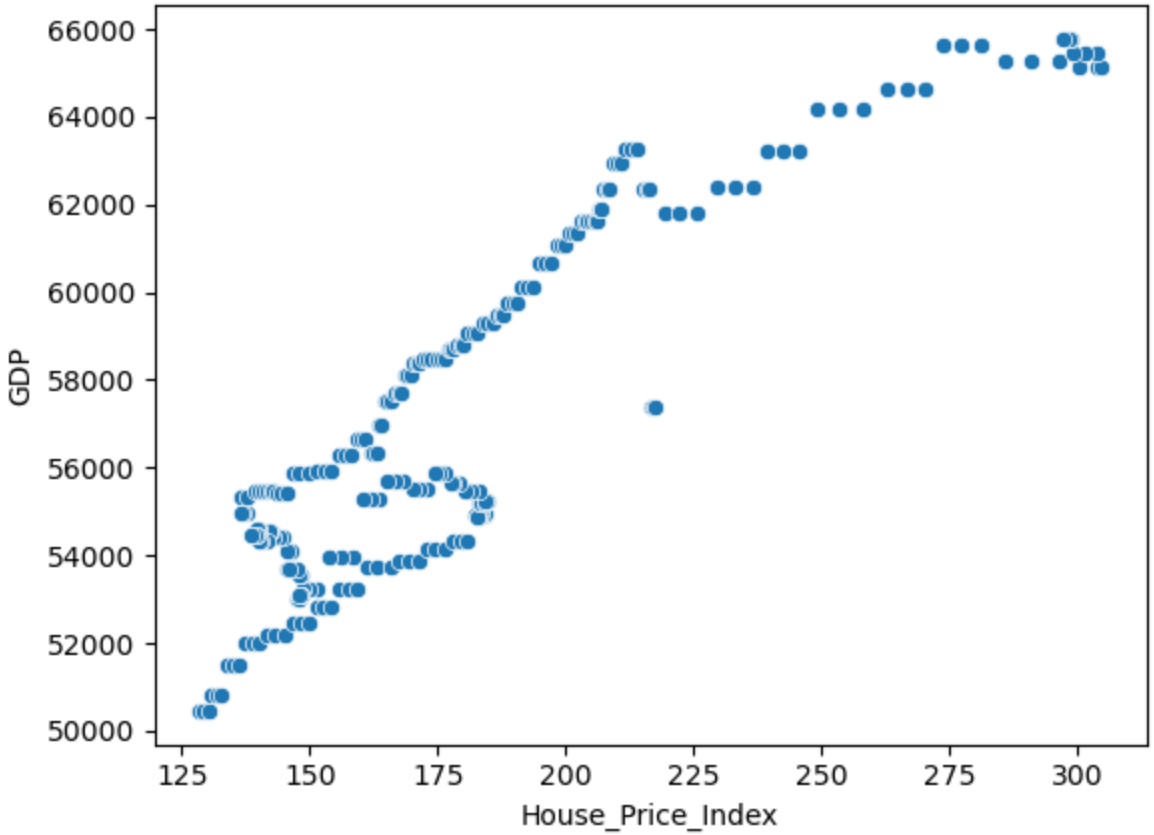

300
280
260
240
220
200
180
CPI
125
150
175
200
225
250
275
300
House_Price_Index

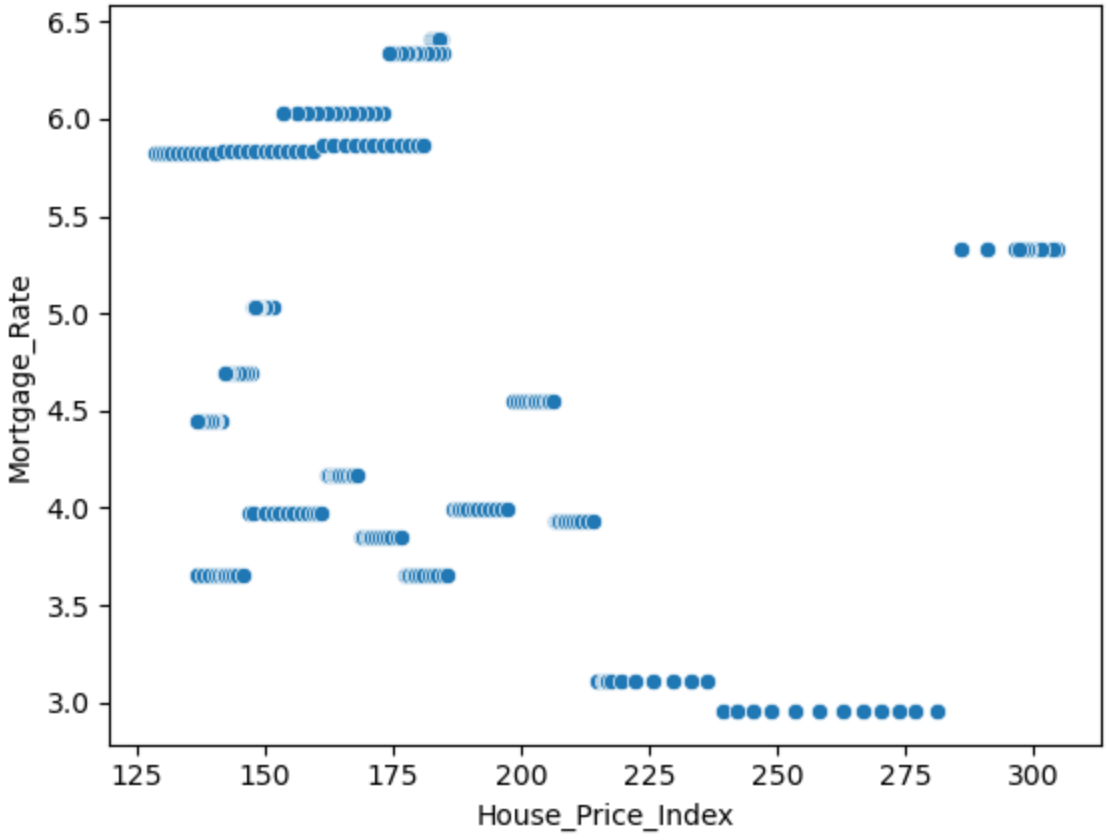

1e8
3.3
3.2
3.1
3.0
2.9
population
125
150
175
200
225
250
275
300
House_Price_Index

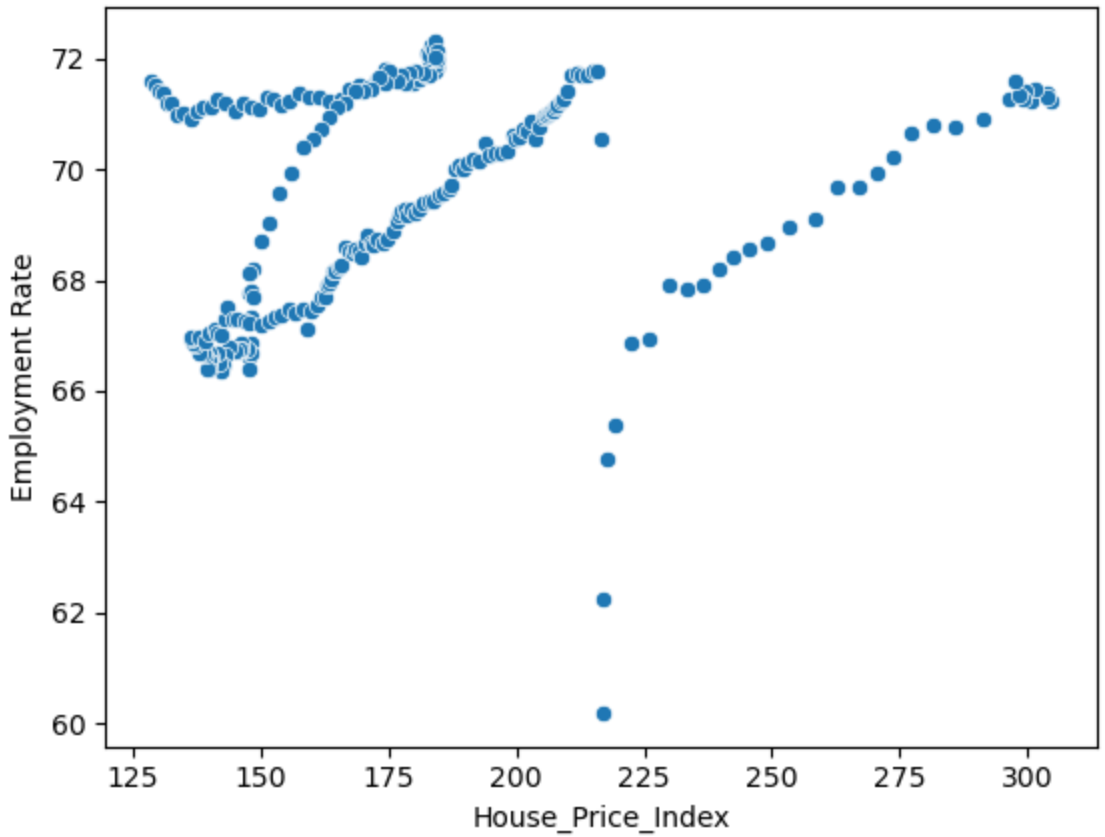

subsidy
House_Price_Index
25
30
35
40
45
125
150
175
200
225
250
275
300

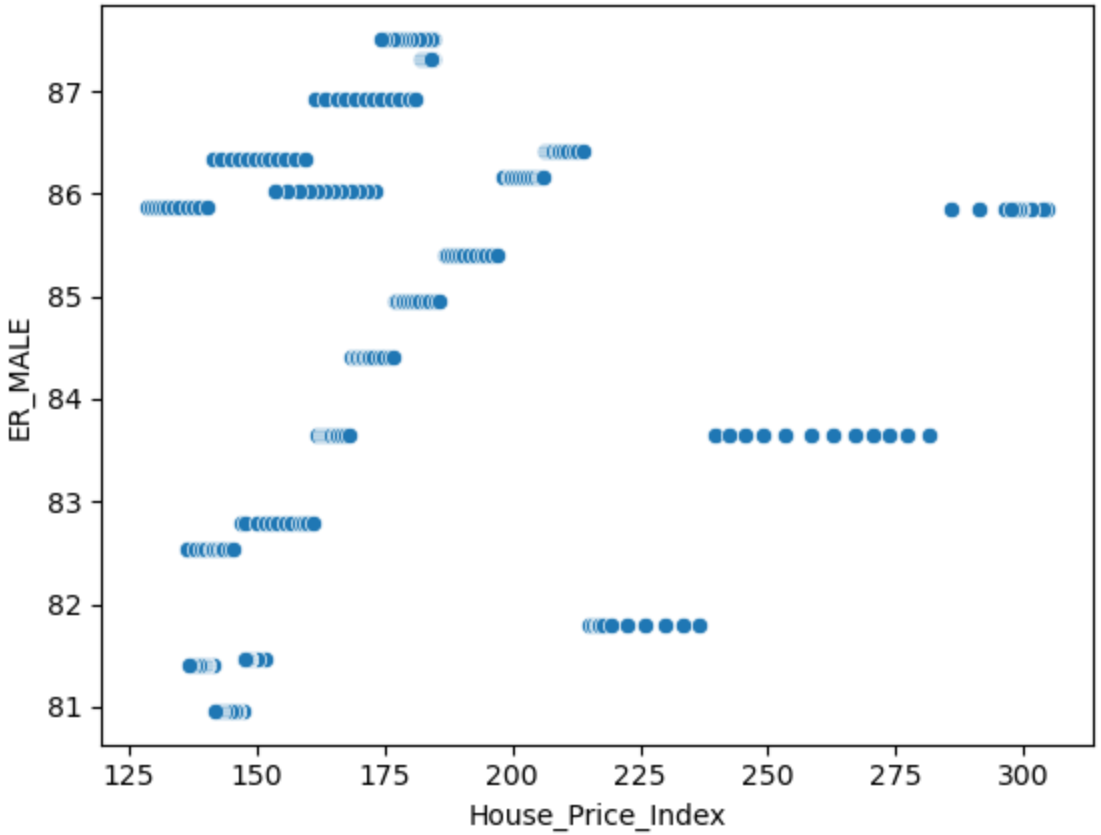

Inflation
House_Price_Index
8
6
4
2
0
125
150
175
200
225
250
275
300

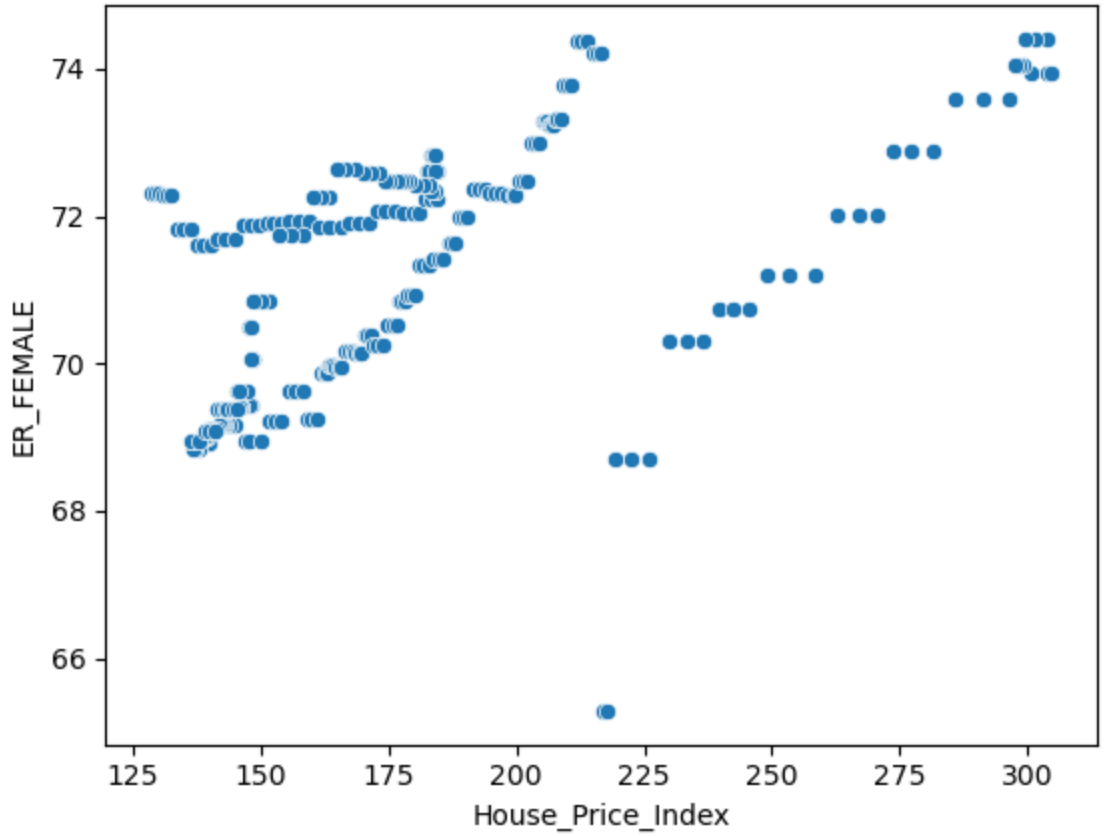

Annual Change
House_Price_Index
3
2
1
0
−1
−2
−3
−4
125
150
175
200
225
250
275
300

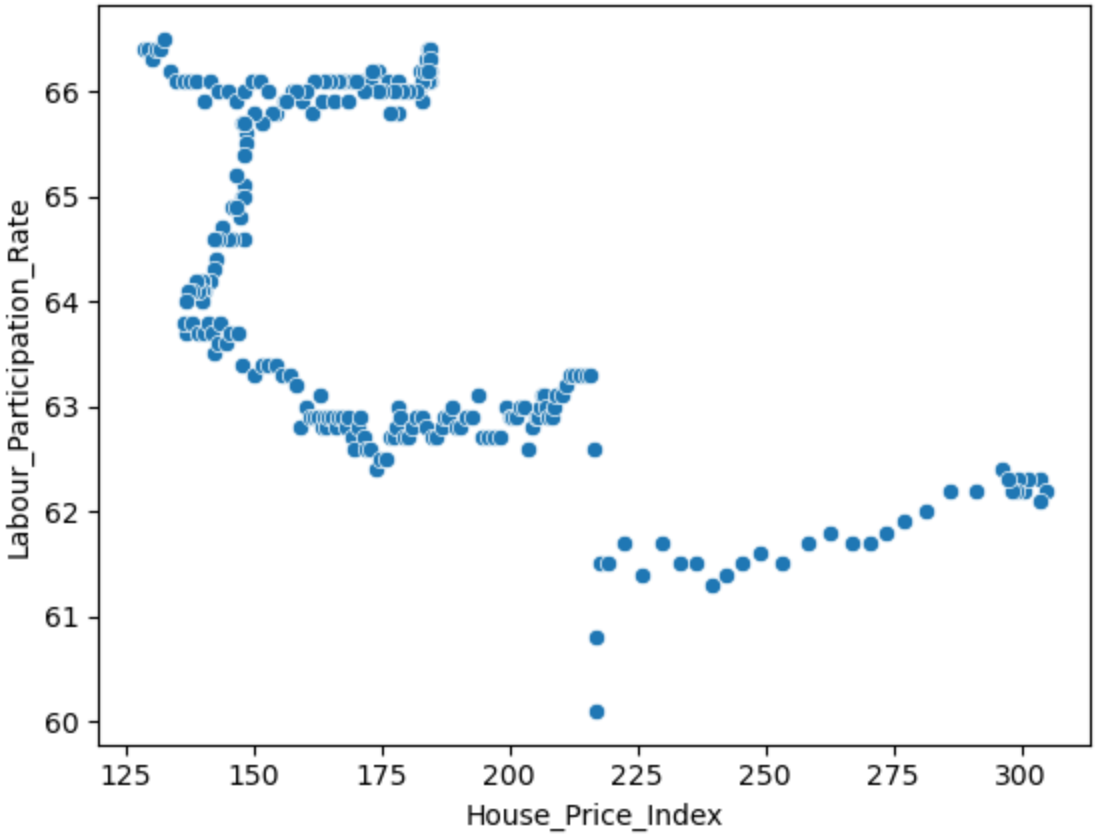

66000
64000
62000
60000
58000
56000
54000
52000
50000
GDP
125
150
175
200
225
250
275
300
House_Price_Index

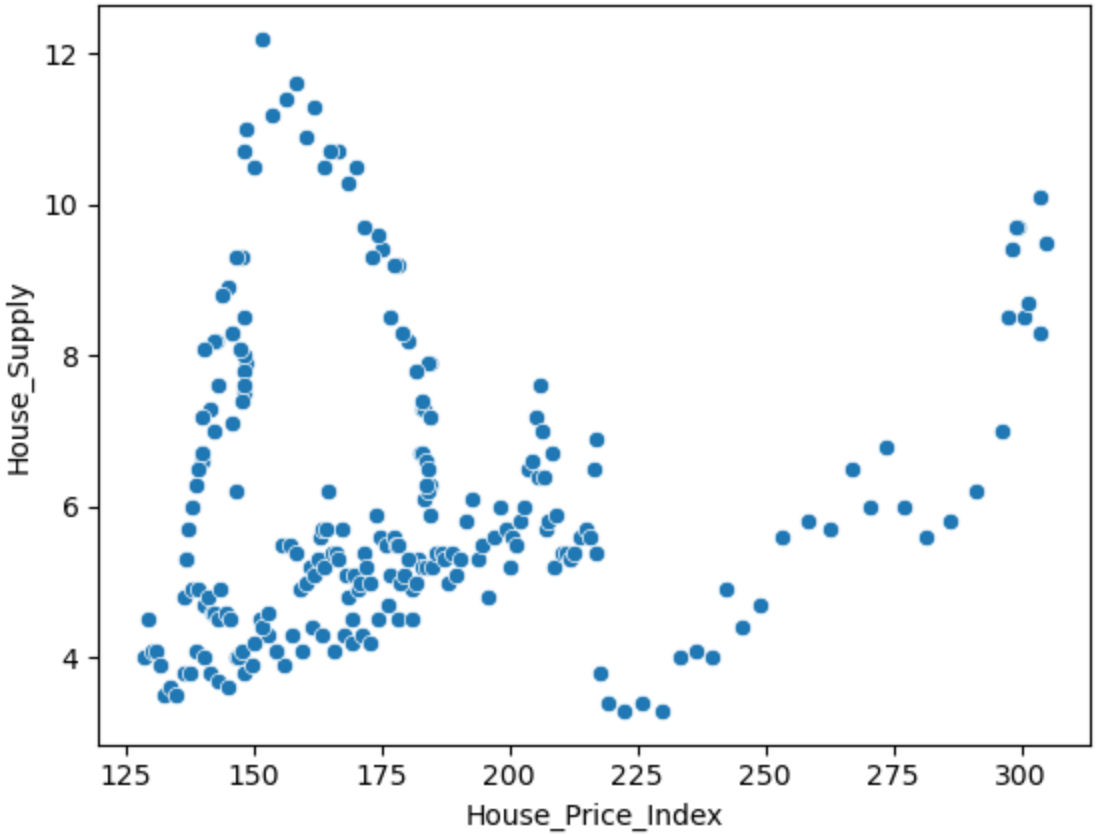

Mortgage_Rate
House_Price_Index
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
125
150
175
200
225
250
275
300

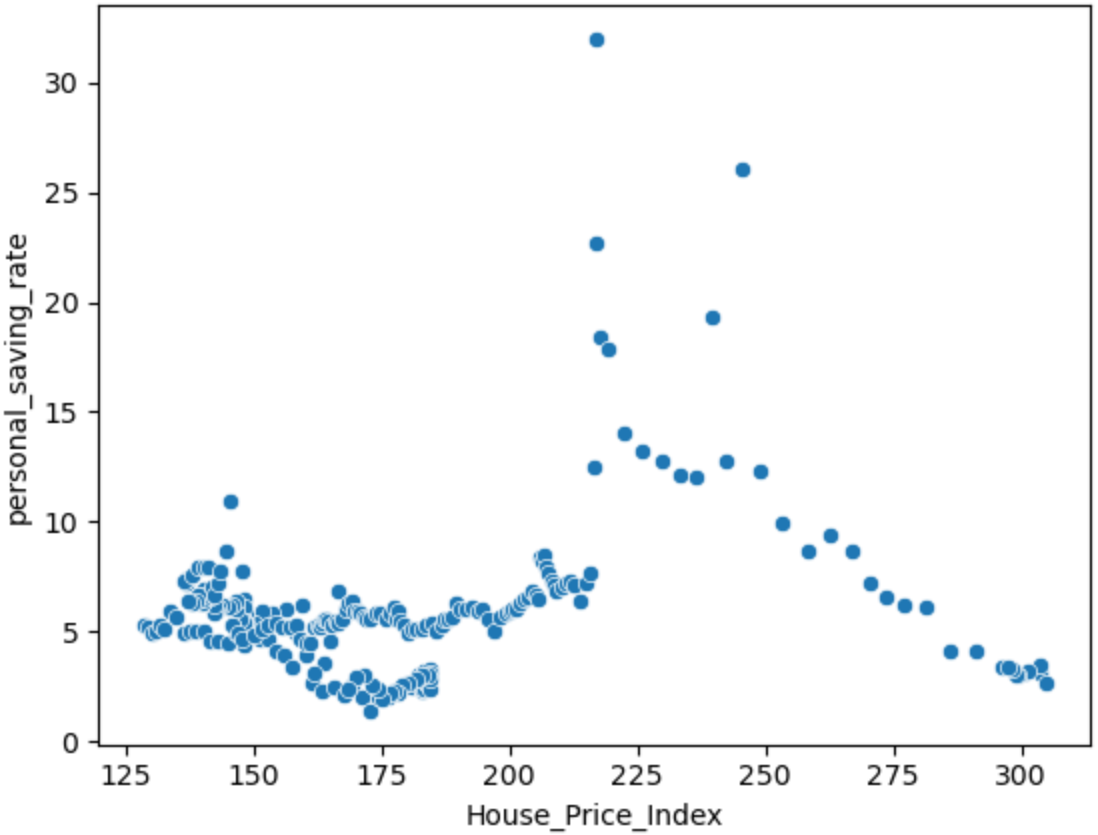

Employment Rate
72
70
68
66
64
62
60
125
150
175
200
225
250
275
300
House_Price_Index

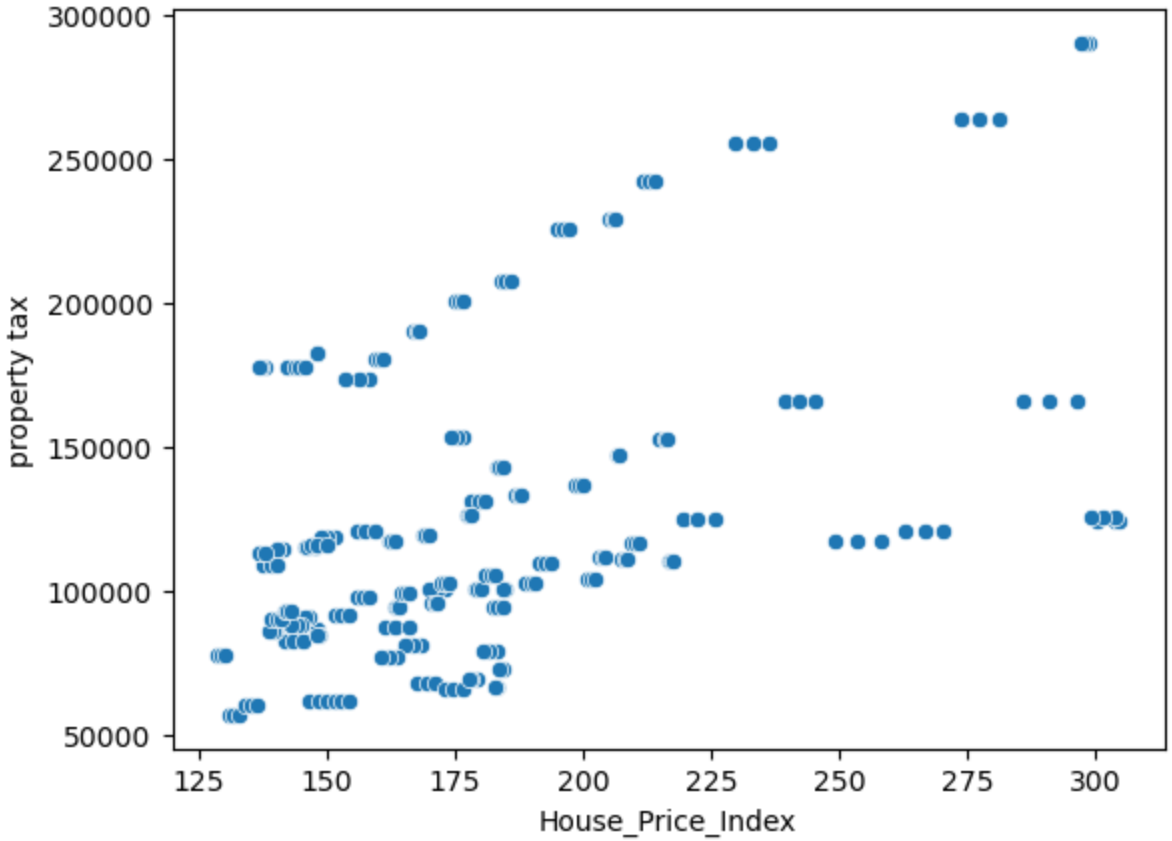

ER_MALE
87
86
85
84
83
82
81
125
150
175
200
225
250
275
300
House_Price_Index

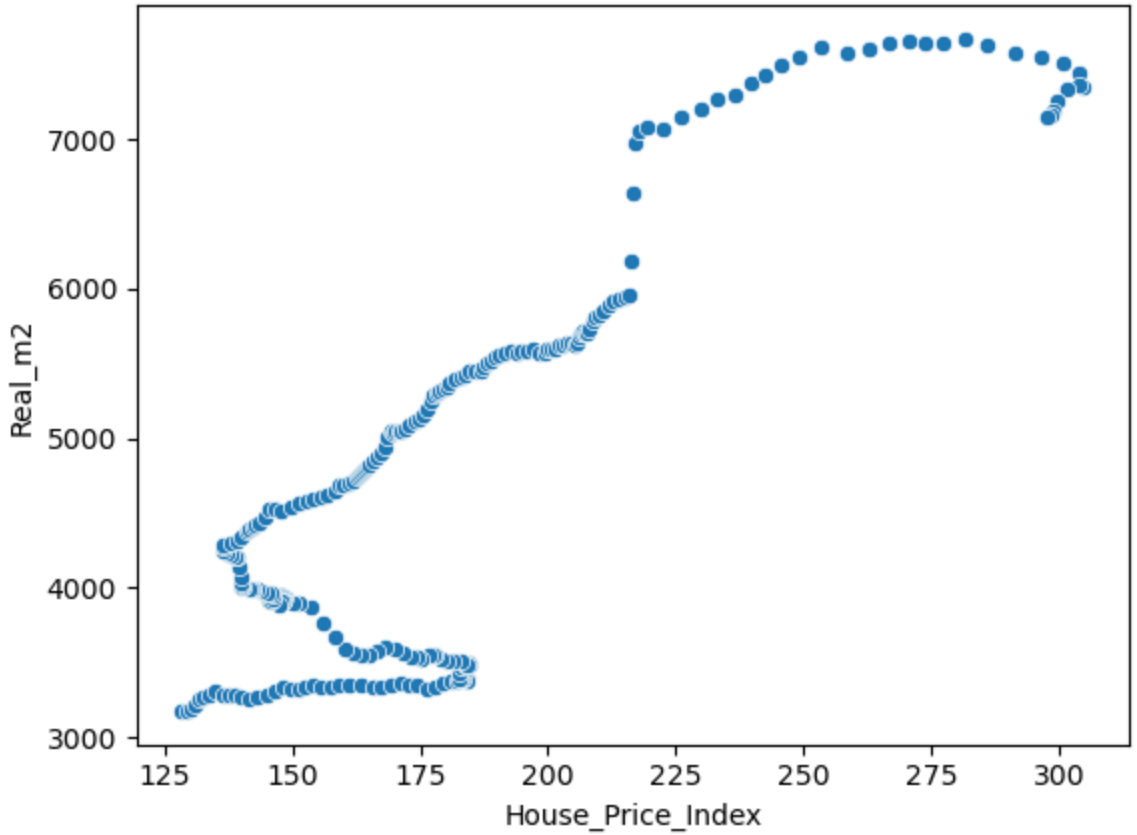

74
72
70
68
66
ER_FEMALE
125
150
175
200
225
250
275
300
House_Price_Index

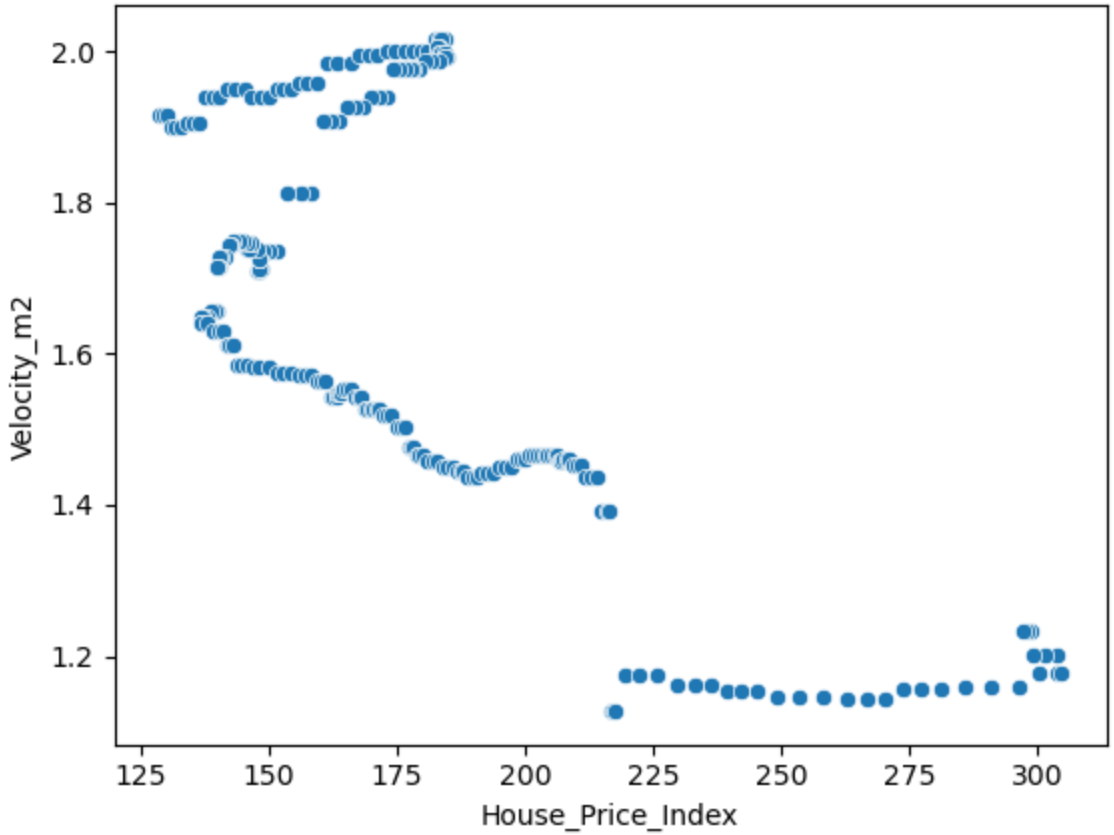

Labour_Participation_Rate
66
65
64
63
62
61
60
125
150
175
200
225
250
275
300
House_Price_Index

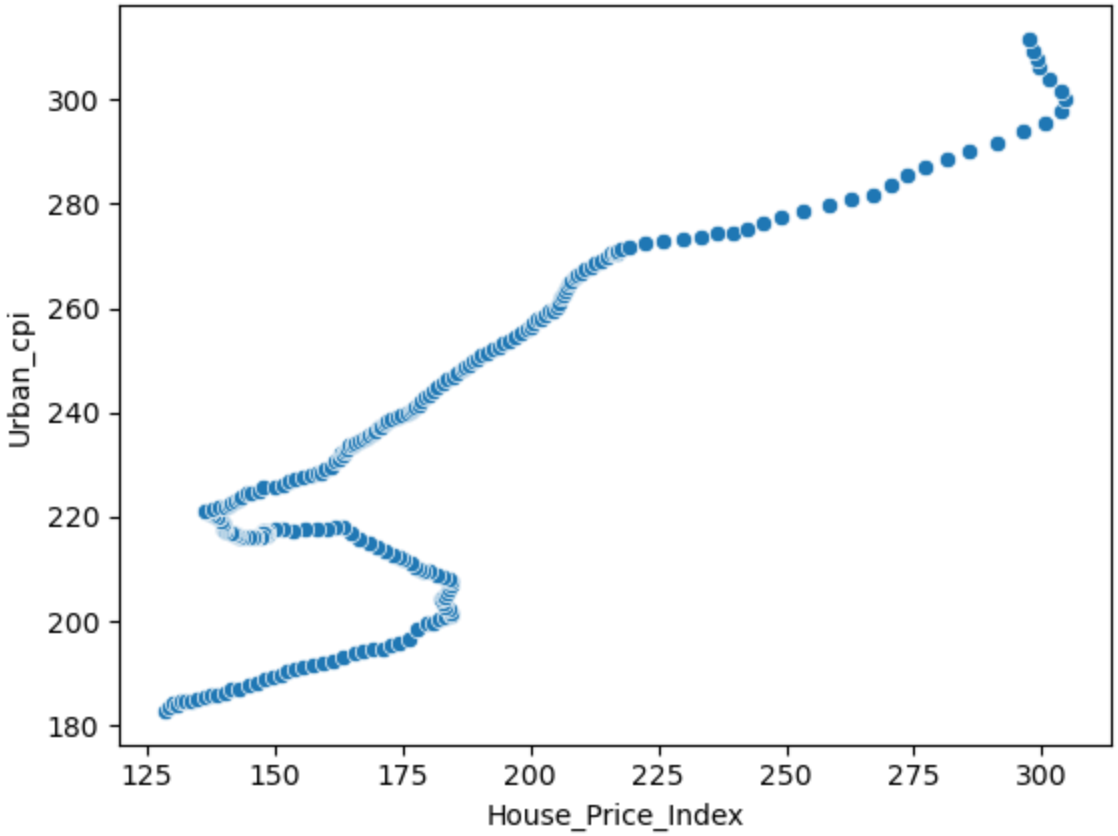

12
10
8
6
4
House_Supply
125
150
175
200
225
250
275
300
House_Price_Index

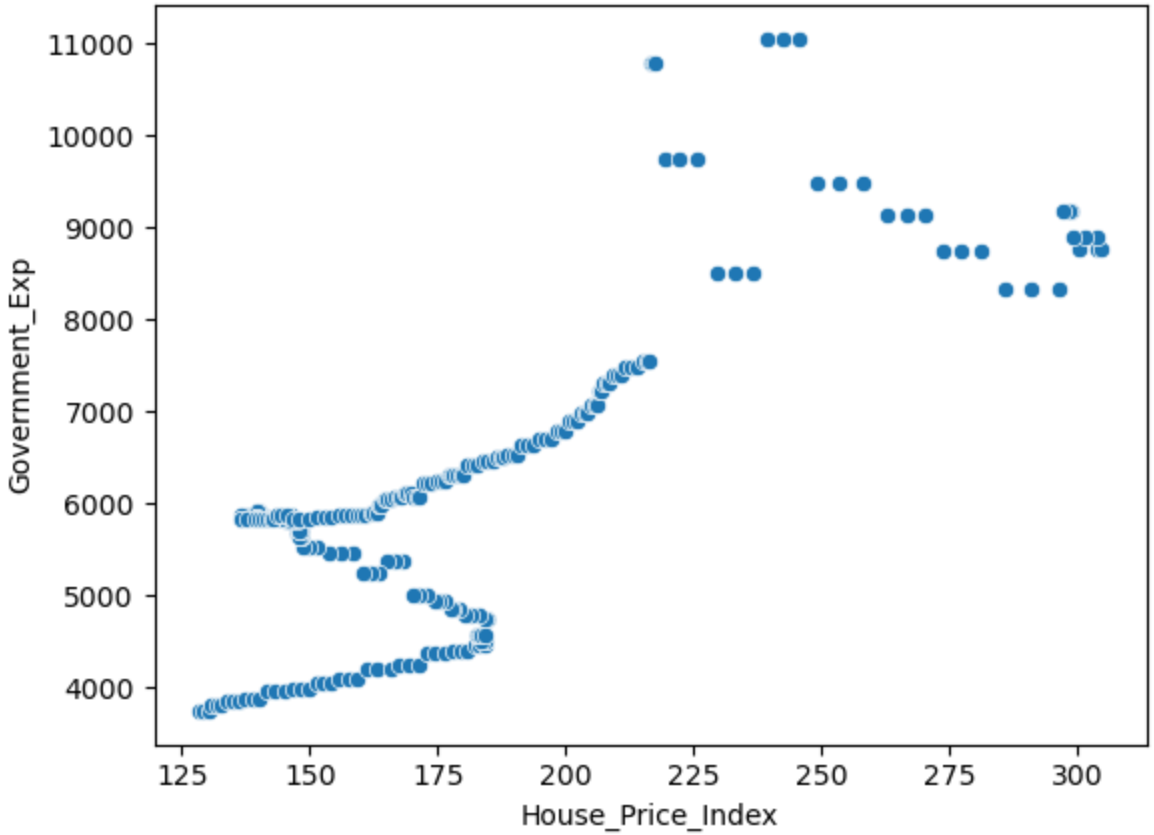

personal_saving_rate
House_Price_Index
0
5
10
15
20
25
30
125
150
175
200
225
250
275
300

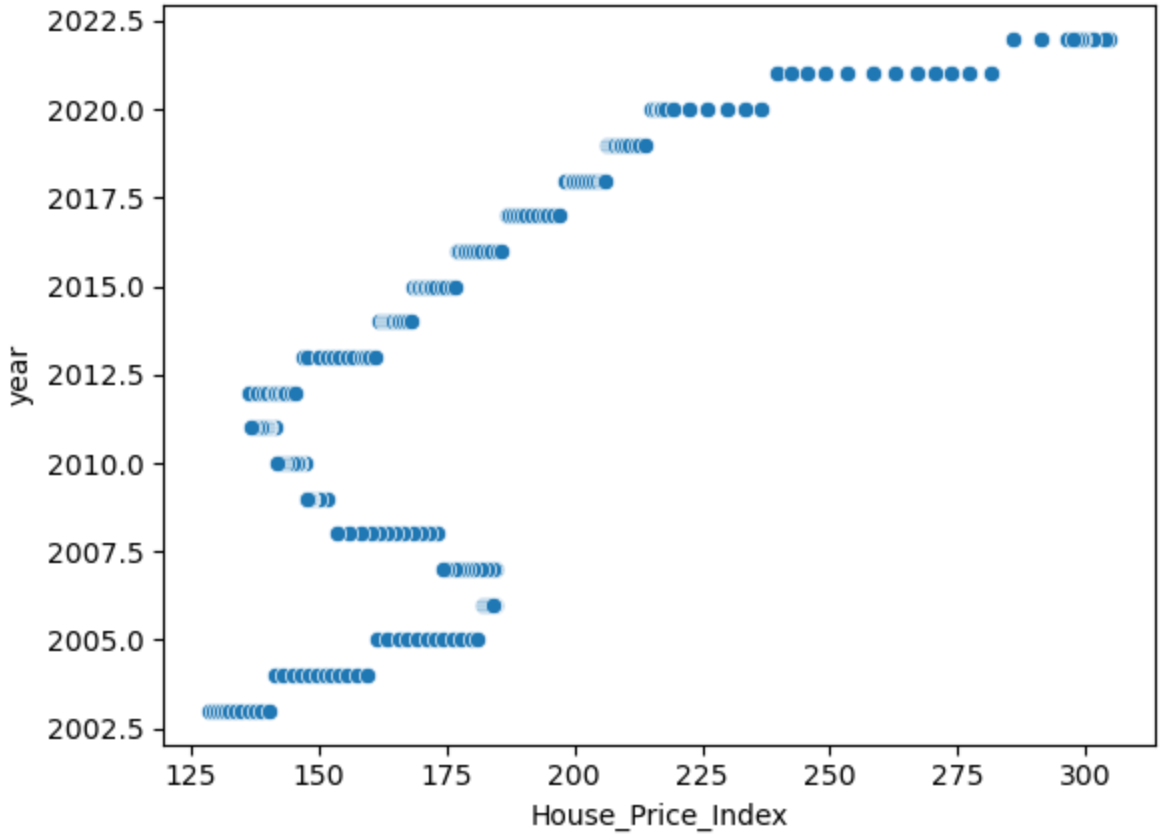

300000
250000
200000
150000
100000
50000
property tax
125
150
175
200
225
250
275
300
House_Price_Index

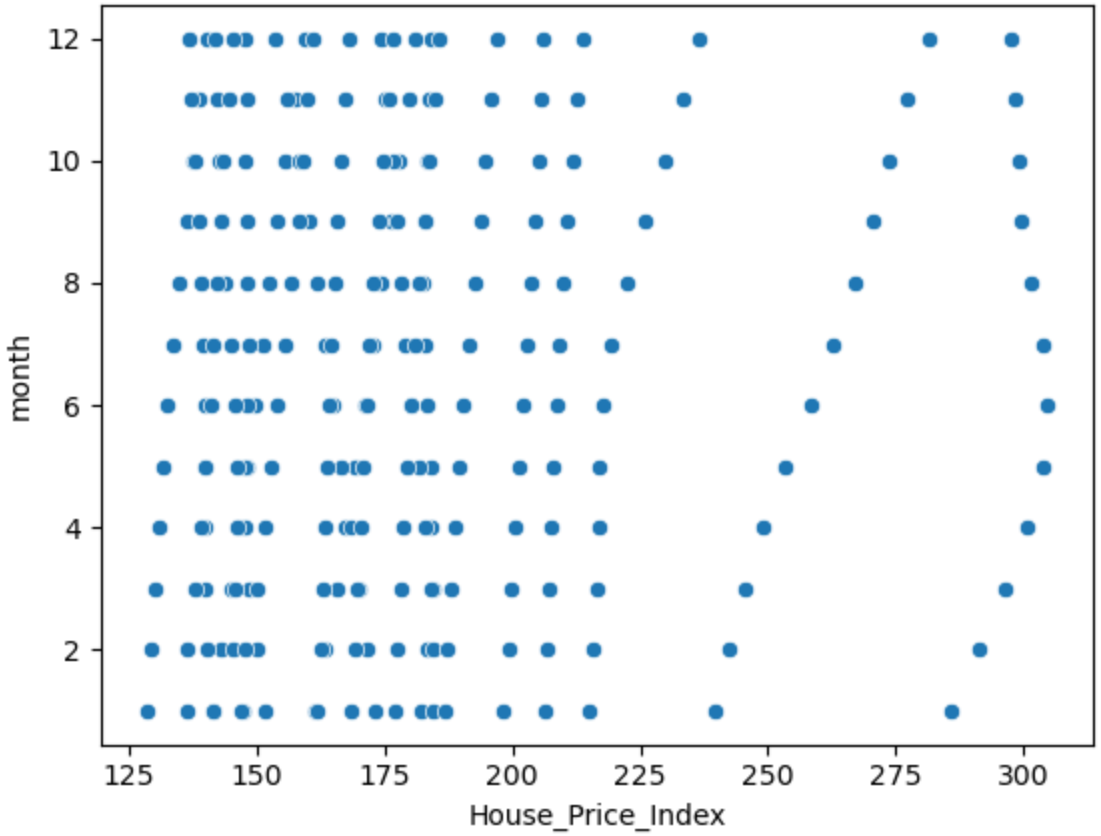

7000
6000
5000
4000
3000
Real_m2
125
150
175
200
225
250
275
300
House_Price_Index


2.0
1.8
1.6
1.4
1.2
Velocity_m2
125
150
175
200
225
250
275
300
House_Price_Index


Urban_cpi
300
280
260
240
220
200
180
125
150
175
200
225
250
275
300
House_Price_Index


Government_Exp
House_Price_Index
11000
10000
9000
8000
7000
6000
5000
4000
125
150
175
200
225
250
275
300


2022.5
2020.0
2017.5
2015.0
2012.5
2010.0
2007.5
2005.0
2002.5
year
125
150
175
200
225
250
275
300
House_Price_Index




```
# as per first assumption there is sort of linear relationship between input and output column, in some
columns there is no linear relationship
```

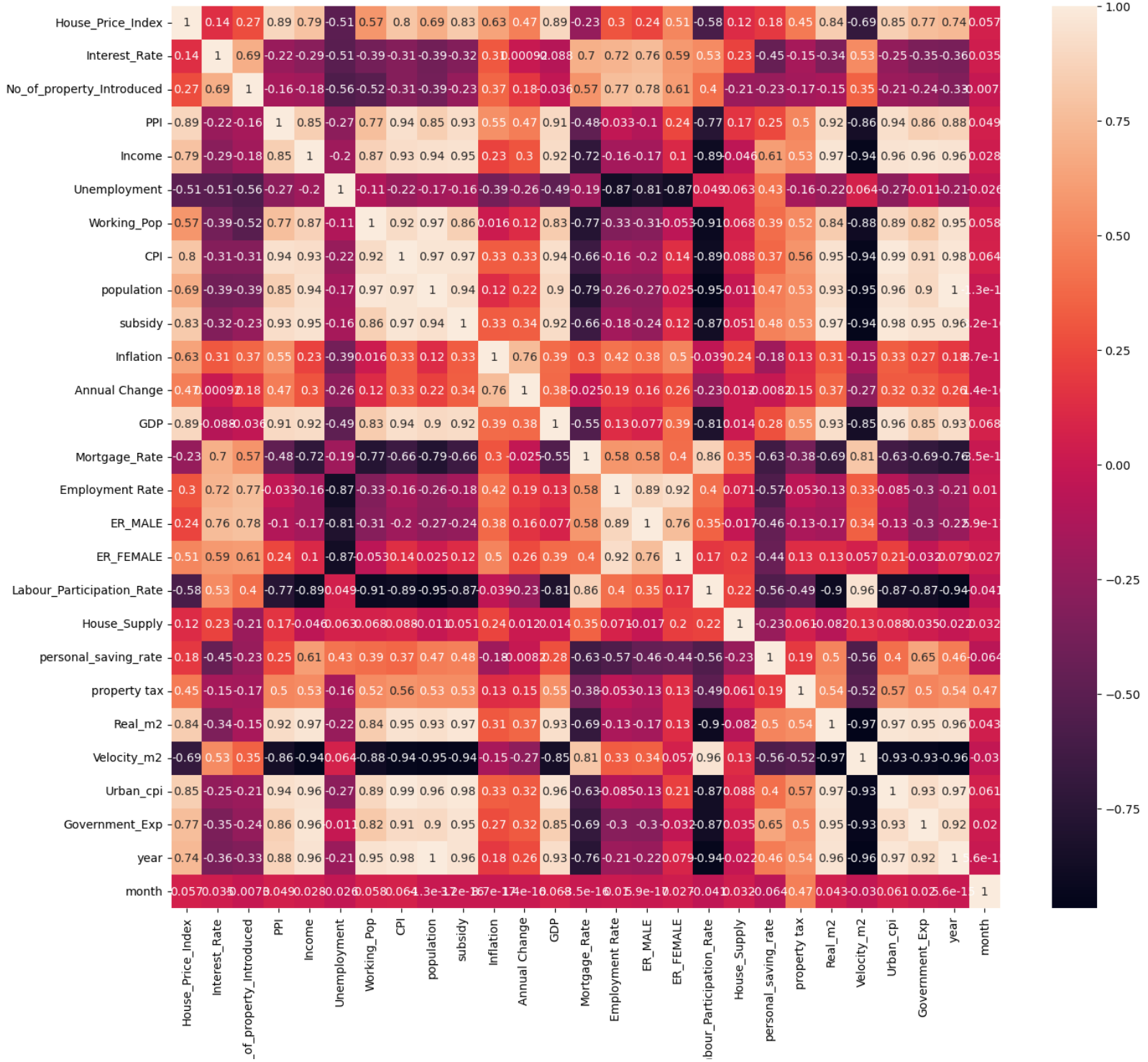
In [ ]:

Multicollinearity check

In [168]:
```
plt.figure(figsize=(16,14))
sns.heatmap(file.corr(), annot=True)
```

```
C:\Users\ravin\AppData\Local\Temp\ipykernel_14800\738317289.py:2: FutureWarning: The default value of nu
meric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only v
alid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(file.corr(), annot=True)
```

Out[168]:
```
<Axes: >
```


House_Price_Index
Interest_Rate
No_of_property_Introduced
PPI
Income
Unemployment
Working_Pop
CPI
population
subsidy
Inflation
Annual Change
GDP
Mortgage_Rate
Employment Rate
ER_MALE
ER_FEMALE
Labour_Participation_Rate
House_Supply
personal_saving_rate
property tax
Real_m2
Velocity_m2
Urban_cpi
Government_Exp
year
month
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

```
# There is problem of MUlticollinearity so we have to do feature selection
```

In [201]:
```
cor = file1.corr()
```

In [202]:
```
threshold = 0.85


selected_features = set()

for i in range(len(cor.columns)):
    for j in range(i):
        if abs(cor.iloc[i, j]) > threshold:
            colname = cor.columns[i]
            selected_features.add(colname)


selected_features_list = list(selected_features)
```

In [207]:
```
columns_not_selected = file1.columns[~file1.columns.isin(selected_features_list)]

data = file1[columns_not_selected]
```

In [208]:
```
data
```

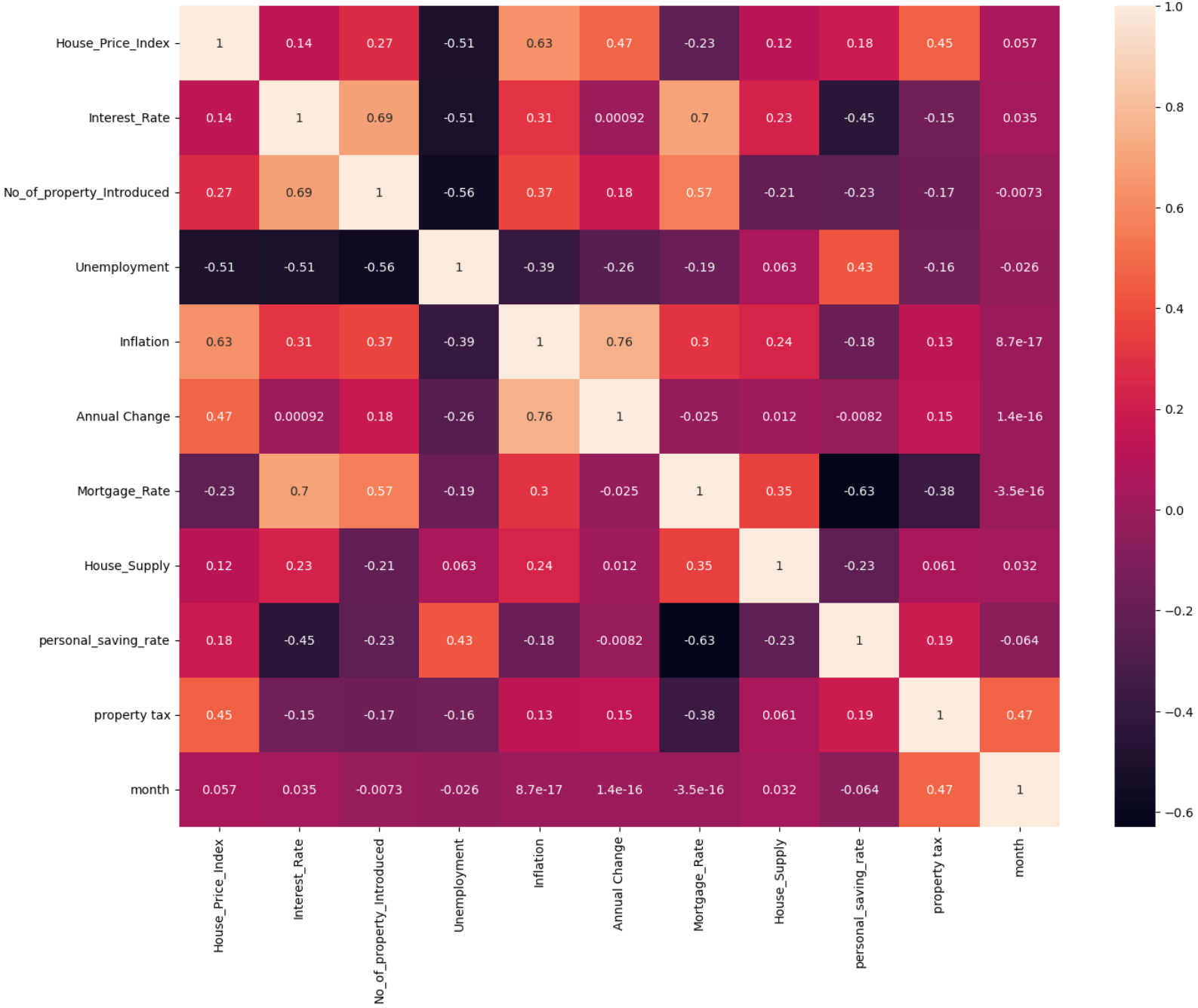
Out[208]:

	House_Price_Index	Interest_Rate	No_of_property_Introduced	Unemployment	Inflation	Annual Change	Mortgage_Rate	House_Suppl
0	128.461	1.24	1654	5.8	2.2701	0.68	5.82698	4.
1	129.355	1.26	1688	5.9	2.2701	0.68	5.82698	4.
2	130.148	1.25	1638	5.9	2.2701	0.68	5.82698	4.
3	130.884	1.26	1662	6.0	2.2701	0.68	5.82698	4.
4	131.735	1.26	1733	6.1	2.2701	0.68	5.82698	3.
...	...	...	...	...	...	...	...	.
235	301.473	2.33	1355	3.7	8.0028	3.30	5.32750	8.
236	299.353	2.56	1438	3.5	8.0028	3.30	5.32750	9.
237	298.873	3.08	1348	3.7	8.0028	3.30	5.32750	9.
238	298.269	3.78	1543	3.6	8.0028	3.30	5.32750	9.
239	297.413	4.10	1390	3.5	8.0028	3.30	5.32750	8.

240 rows × 11 columns

In [209]:
```
plt.figure(figsize=(16,12))
sns.heatmap(data.corr(), annot=True)
```

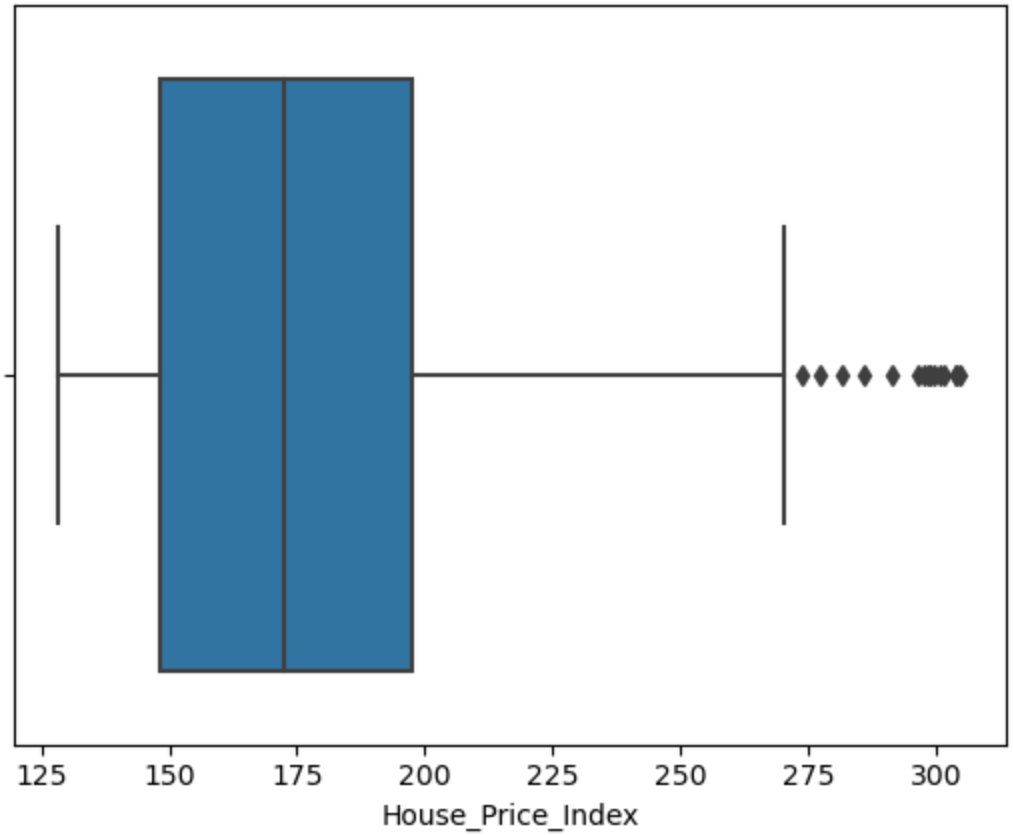
Out[209]: <Axes: >

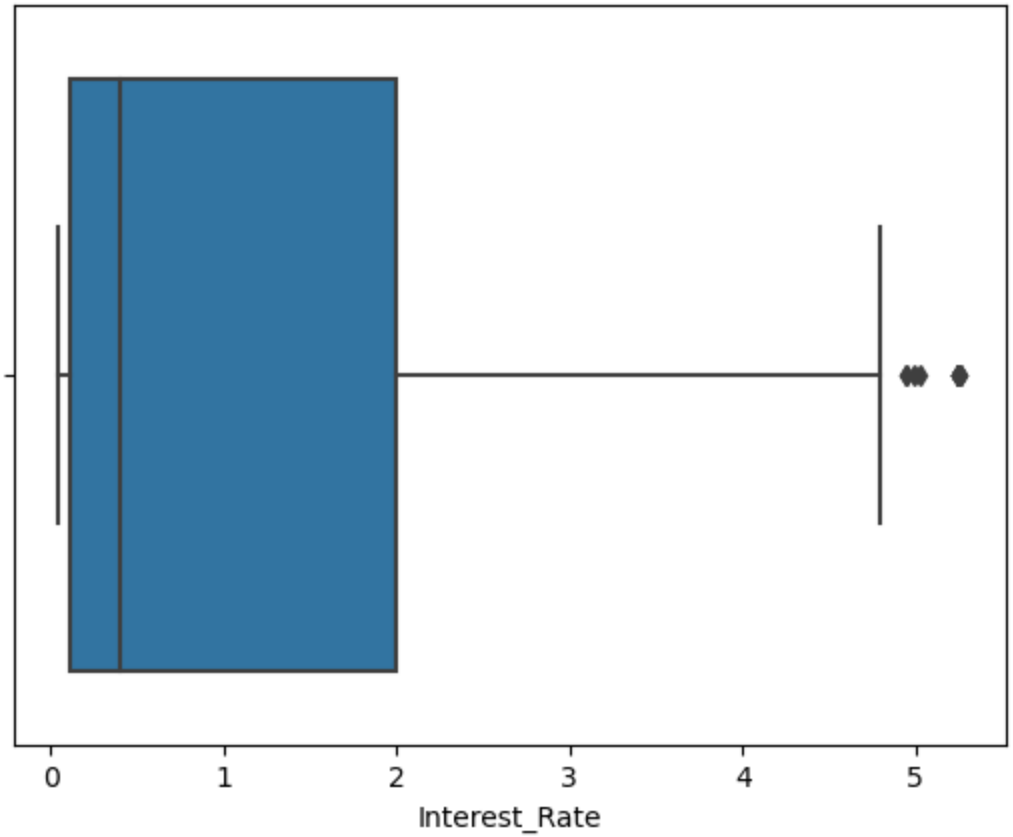

House_Price_Index 1 0.14 0.27 -0.51 0.63 0.47 -0.23 0.12 0.18 0.45 0.057
Interest_Rate 0.14 1 0.69 -0.51 0.31 0.00092 0.7 0.23 -0.45 -0.15 0.035
No_of_property_Introduced 0.27 0.69 1 -0.56 0.37 0.18 0.57 -0.21 -0.23 -0.17 -0.0073
Unemployment -0.51 -0.51 -0.56 1 -0.39 -0.26 -0.19 0.063 0.43 -0.16 -0.026
Inflation 0.63 0.31 0.37 -0.39 1 0.76 0.3 0.24 -0.18 0.13 8.7e-17
Annual Change 0.47 0.00092 0.18 -0.26 0.76 1 -0.025 0.012 -0.0082 0.15 1.4e-16
Mortgage_Rate -0.23 0.7 0.57 -0.19 0.3 -0.025 1 0.35 -0.63 -0.38 -3.5e-16
House_Supply 0.12 0.23 -0.21 0.063 0.24 0.012 0.35 1 -0.23 0.061 0.032
personal_saving_rate 0.18 -0.45 -0.23 0.43 -0.18 -0.0082 -0.63 -0.23 1 0.19 -0.064
property tax 0.45 -0.15 -0.17 -0.16 0.13 0.15 -0.38 0.061 0.19 1 0.47
month 0.057 0.035 -0.0073 -0.026 8.7e-17 1.4e-16 -3.5e-16 0.032 -0.064 0.47 1
House_Price_Index
Interest_Rate
No_of_property_Introduced
Unemployment
Inflation
Annual Change
Mortgage_Rate
House_Supply
personal_saving_rate
property tax
month
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

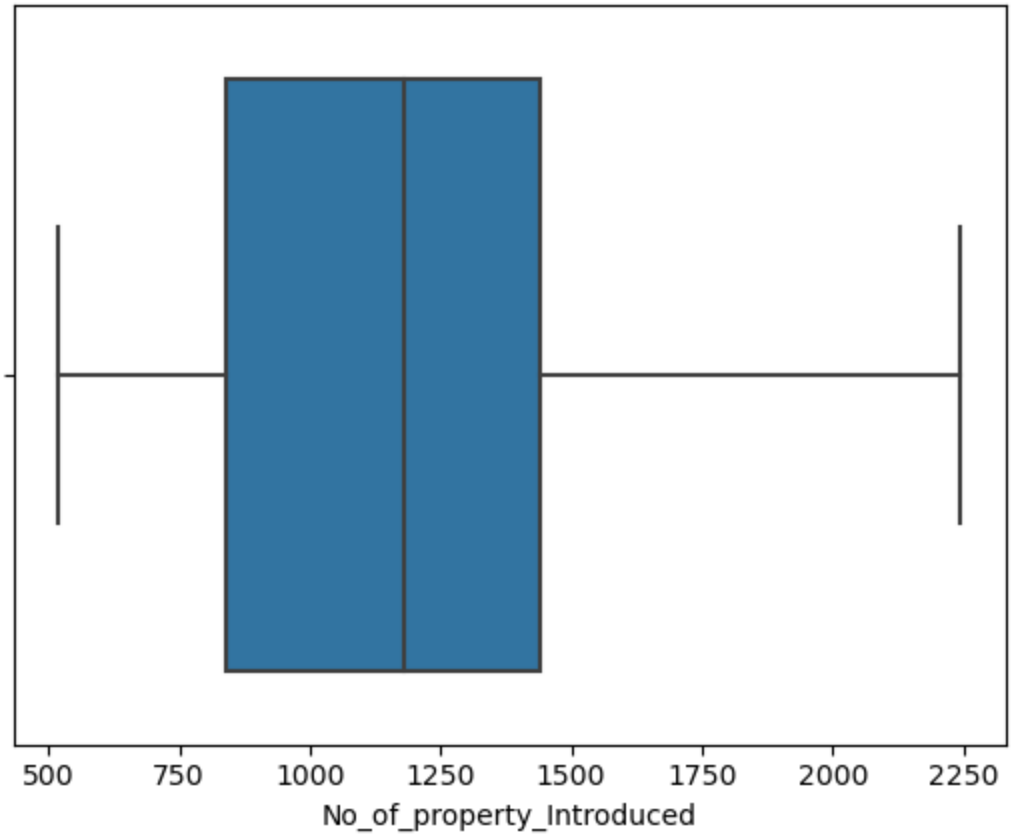
In [ ]:

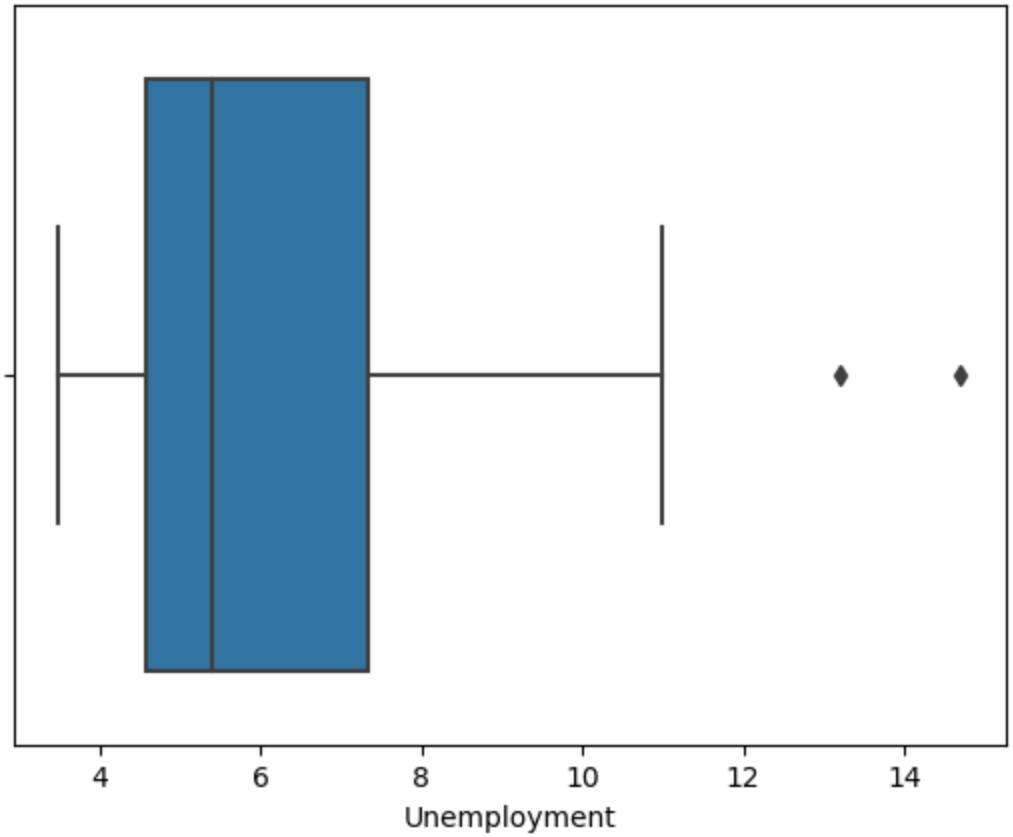
Outliers detection and removal

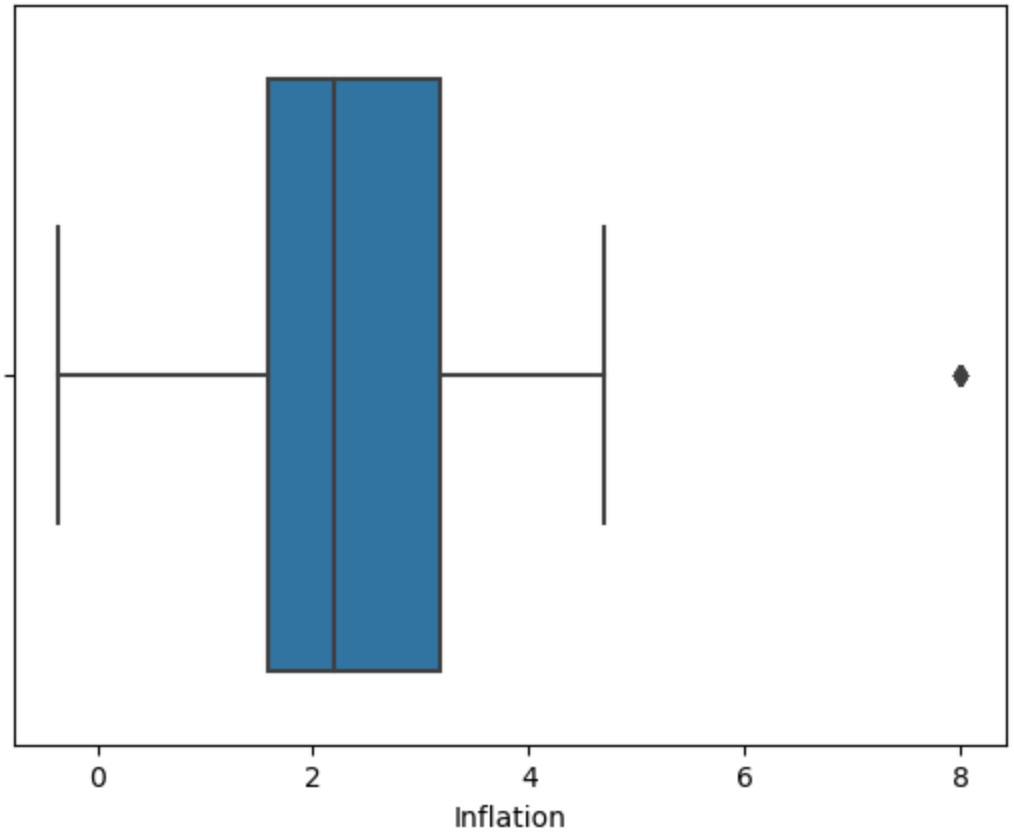
In [262]:
```
for i in data:
    sns.boxplot(data=data, x=i)
    plt.show()
```

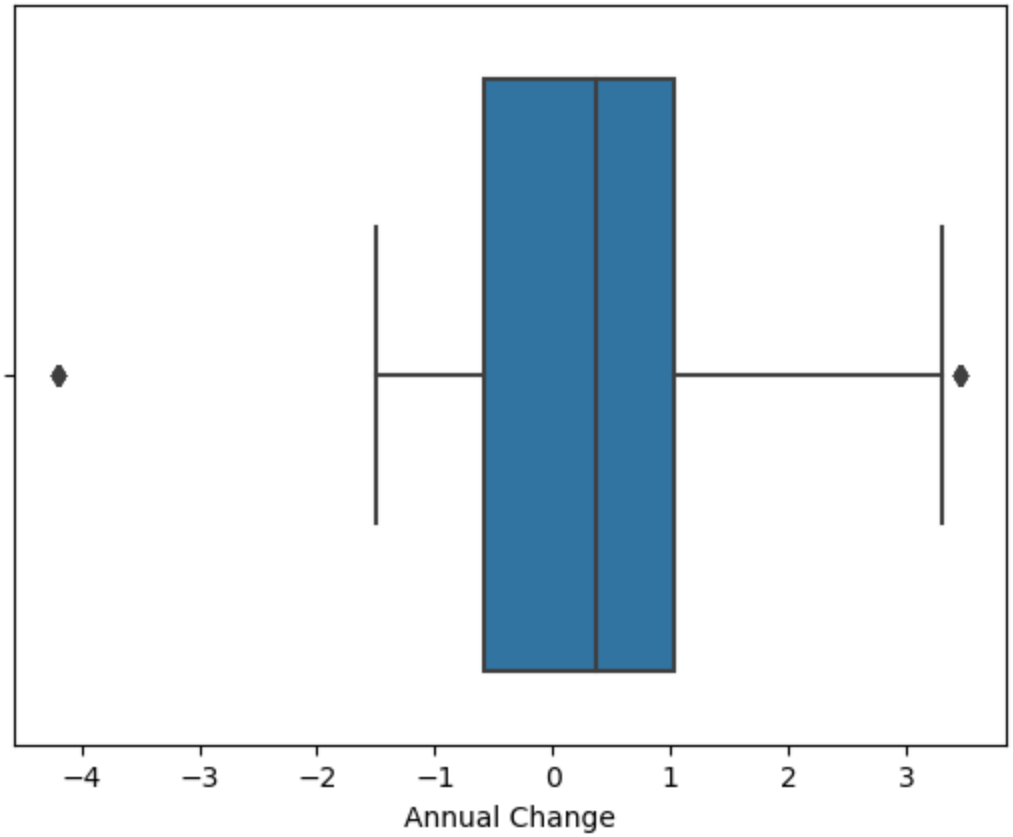


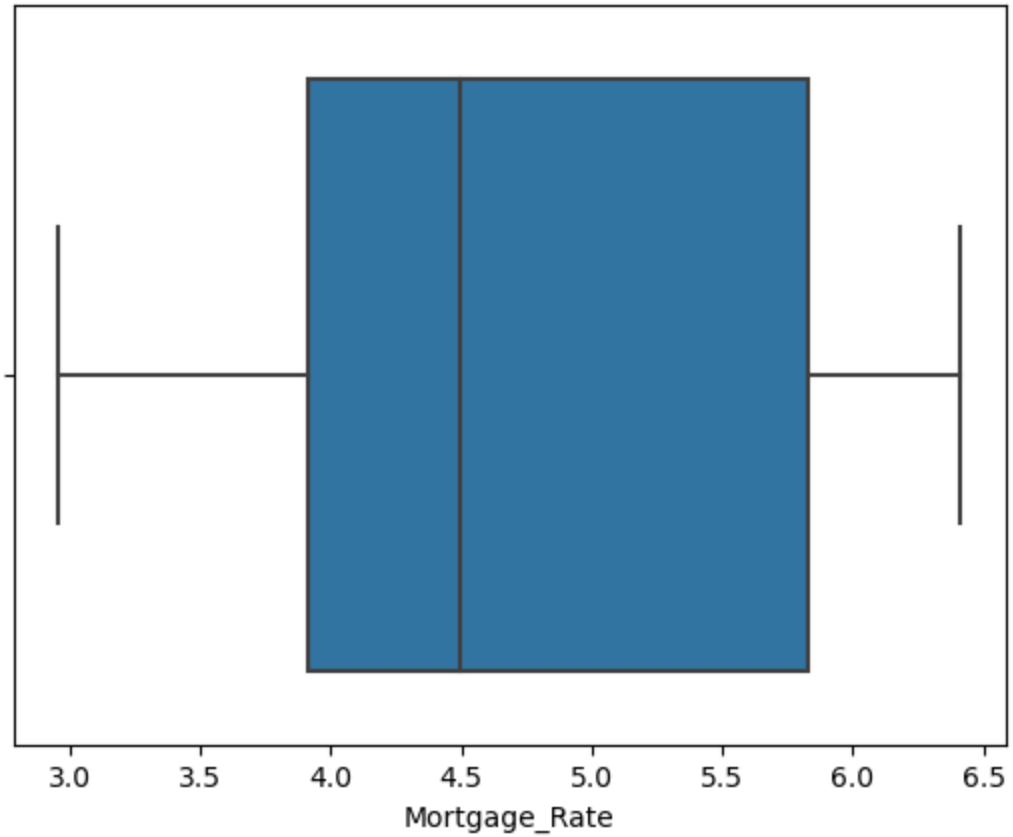

0
1
2
3
4
5
Interest_Rate

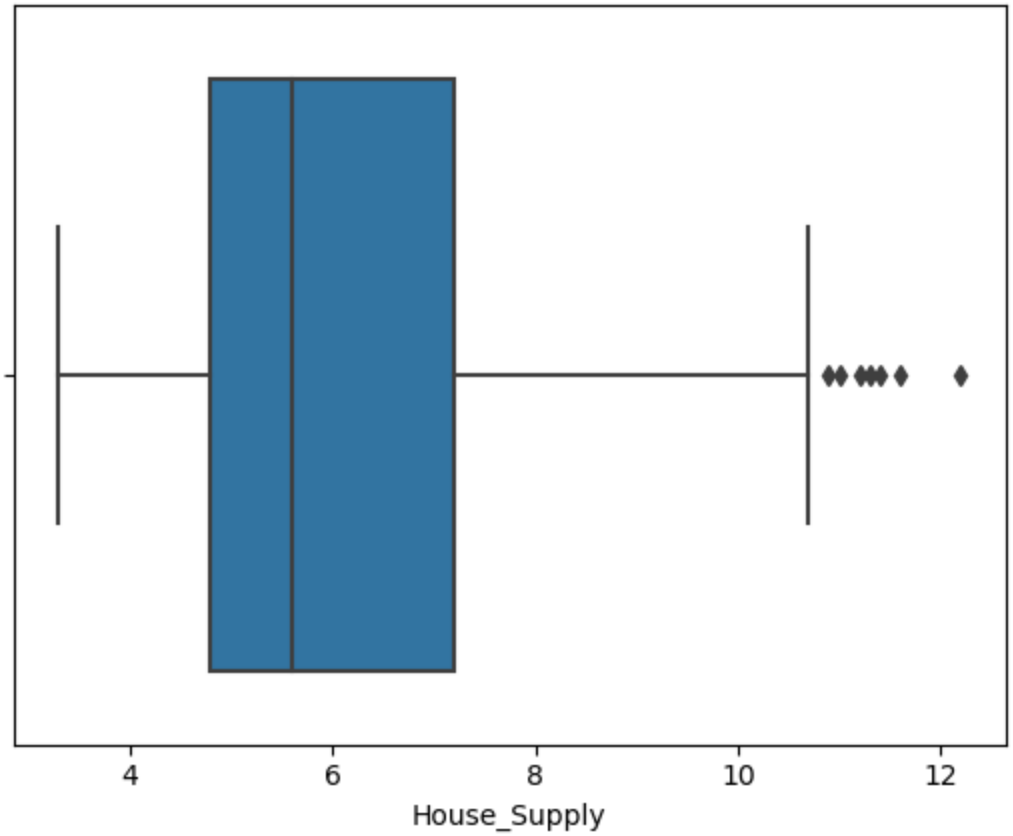

500
750
1000
1250
1500
1750
2000
2250
No_of_property_Introduced

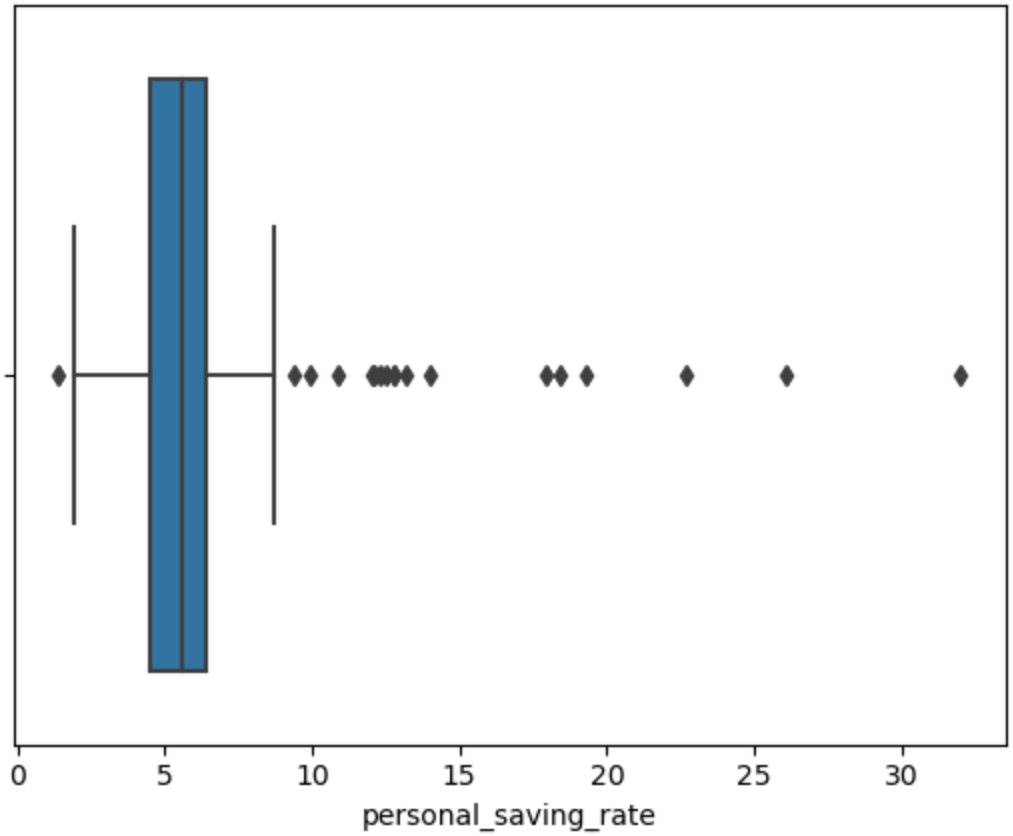

4
6
8
10
12
14
Unemployment

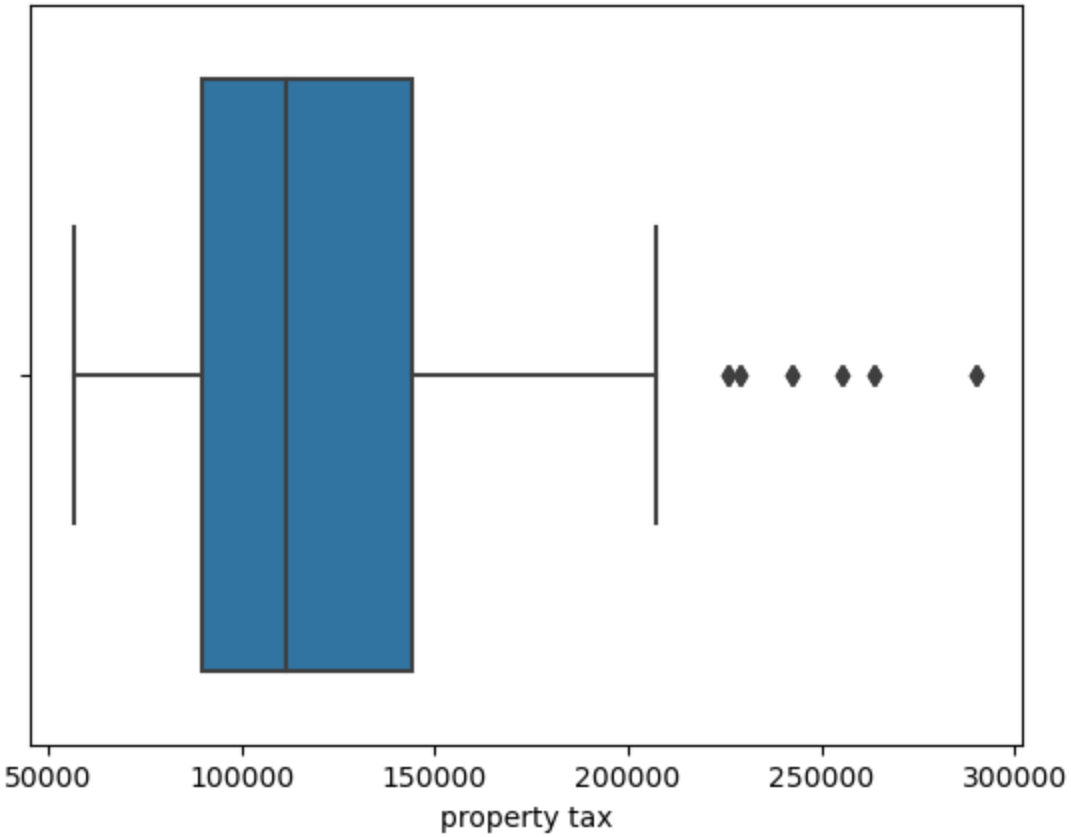

0
2
4
6
8
Inflation

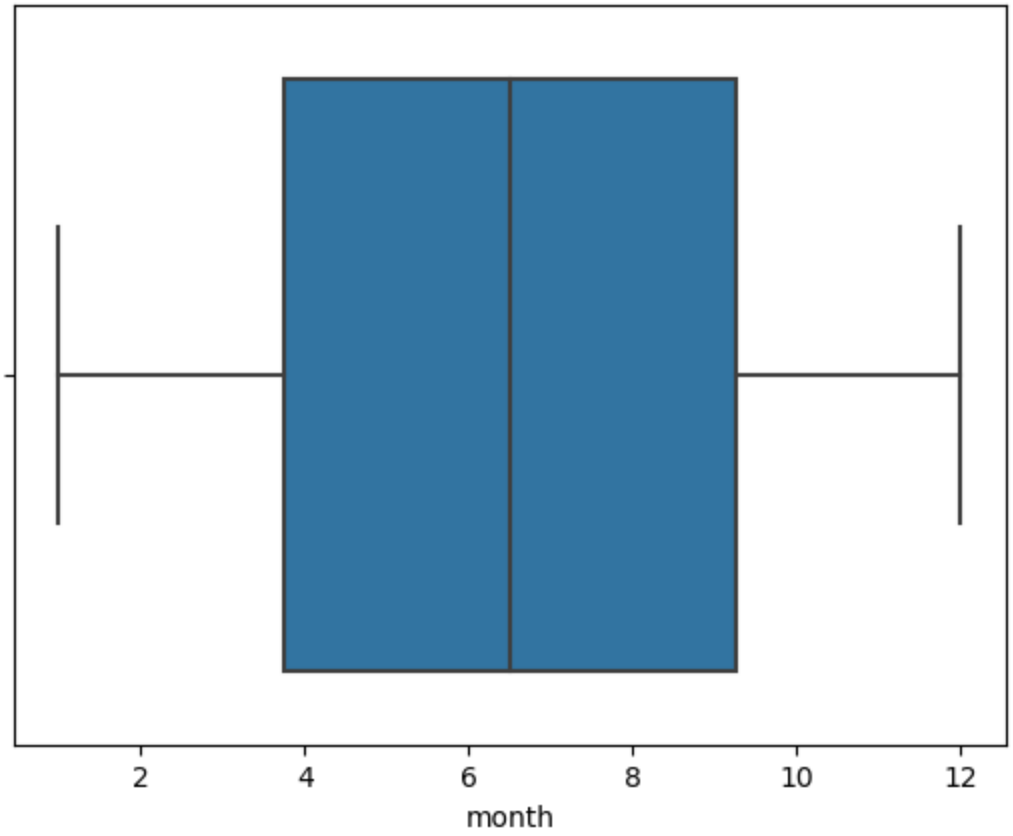

−4
−3
−2
−1
0
1
2
3
Annual Change


3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
Mortgage_Rate


4
6
8
10
12
House_Supply


0
5
10
15
20
25
30
personal_saving_rate


50000
100000
150000
200000
250000
300000
property tax



In [359]:
```
data1 = data[data['House_Price_Index'] <= 280]
data0 = data1[data1['personal_saving_rate'] <= 15]
data2 = data0.reset_index().drop(['index'], axis=1)
```

In [360]:
```
data2
```

Out[360]:

	House_Price_Index	Interest_Rate	No_of_property_Introduced	Unemployment	Inflation	Annual Change	Mortgage_Rate	House_Suppl
0	128.461	1.24	1654	5.8	2.2701	0.68	5.82698	4.
1	129.355	1.26	1688	5.9	2.2701	0.68	5.82698	4.
2	130.148	1.25	1638	5.9	2.2701	0.68	5.82698	4.
3	130.884	1.26	1662	6.0	2.2701	0.68	5.82698	4.
4	131.735	1.26	1733	6.1	2.2701	0.68	5.82698	3.
...	...	...	...	...	...	...	...	.
216	262.820	0.10	1361	5.4	4.6979	3.46	2.95769	5.
217	266.845	0.09	1312	5.2	4.6979	3.46	2.95769	6.
218	270.377	0.08	1232	4.8	4.6979	3.46	2.95769	6.
219	273.725	0.08	1259	4.5	4.6979	3.46	2.95769	6.
220	277.210	0.08	1389	4.2	4.6979	3.46	2.95769	6.

221 rows × 11 columns

scaling

In [361]:
```
x = data2.drop(['House_Price_Index'], axis=1)
y = data2['House_Price_Index']
```

In [362]:
```
from sklearn.preprocessing import RobustScaler

rs = RobustScaler()

scaled = pd.DataFrame(rs.fit_transform(x), columns=column)
```

In [363]:
```
scaled
```

Out[363]:

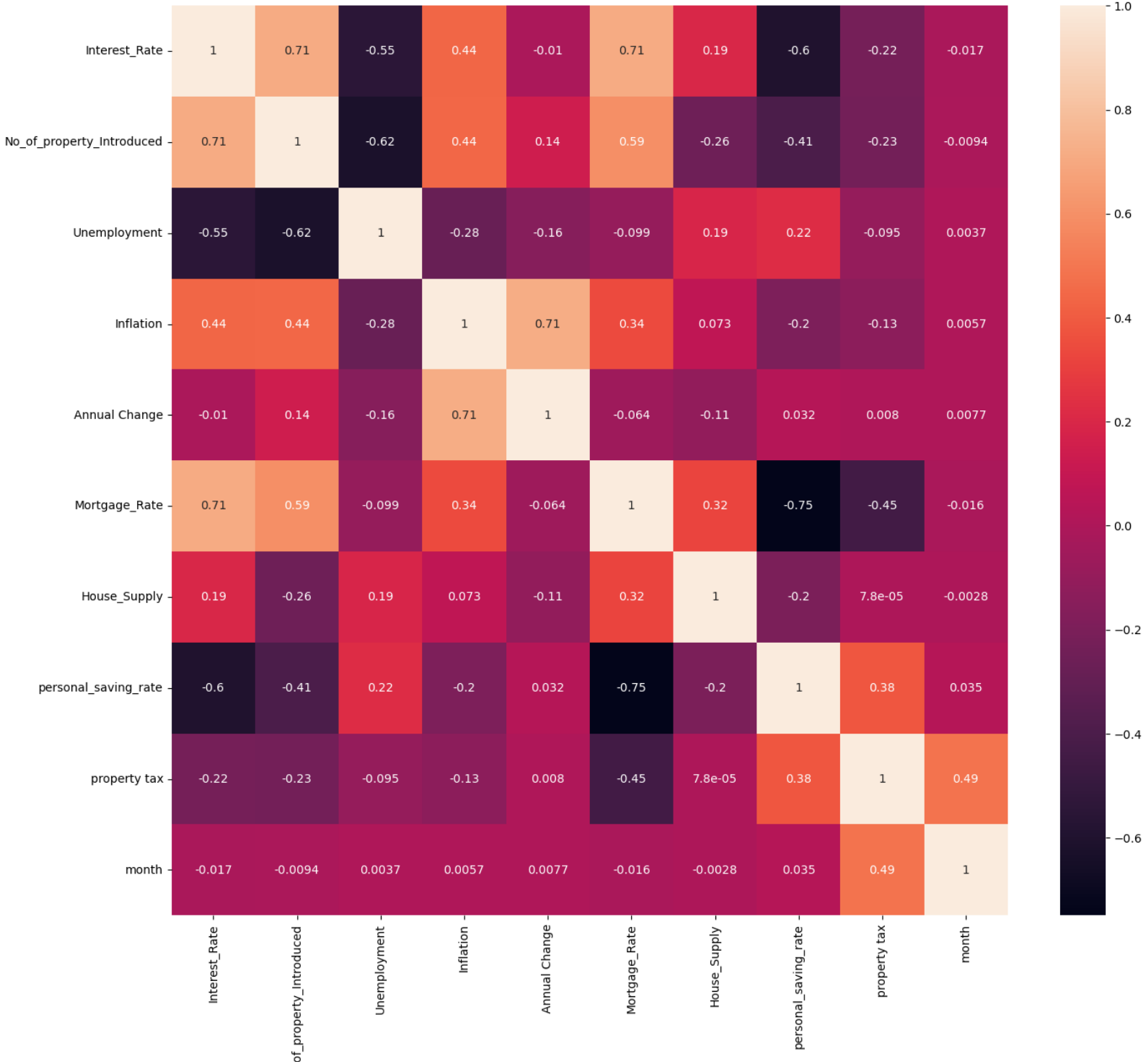
	Interest_Rate	No_of_property_Introduced	Unemployment	Inflation	Annual Change	Mortgage_Rate	House_Supply	personal_saving
0	0.441489	0.764706	0.107143	0.082742	0.232704	0.724523	-0.75	-0.17
1	0.452128	0.815988	0.142857	0.082742	0.232704	0.724523	-0.50	-0.23
2	0.446809	0.740573	0.142857	0.082742	0.232704	0.724523	-0.70	-0.4
3	0.452128	0.776772	0.178571	0.082742	0.232704	0.724523	-0.70	-0.35
4	0.452128	0.883861	0.214286	0.082742	0.232704	0.724523	-0.80	-0.17
...	...	...	...	...	...	...	...	
216	-0.164894	0.322775	-0.035714	1.517612	1.981132	-0.782885	0.10	2.23
217	-0.170213	0.248869	-0.107143	1.517612	1.981132	-0.782885	0.50	1.82
218	-0.175532	0.128205	-0.250000	1.517612	1.981132	-0.782885	0.25	0.9
219	-0.175532	0.168929	-0.357143	1.517612	1.981132	-0.782885	0.65	0.58
220	-0.175532	0.365008	-0.464286	1.517612	1.981132	-0.782885	0.25	0.35

221 rows × 10 columns

Correlation

In [364]:
```
plt.figure(figsize=(16,14))
sns.heatmap(scaled.corr(), annot=True)
```

Out[364]: <Axes: >


Interest_Rate
No_of_property_Introduced
Unemployment
Inflation
Annual Change
Mortgage_Rate
House_Supply
personal_saving_rate
property tax
month
1 0.71 -0.55 0.44 -0.01 0.71 0.19 -0.6 -0.22 -0.017
0.71 1 -0.62 0.44 0.14 0.59 -0.26 -0.41 -0.23 -0.0094
-0.55 -0.62 1 -0.28 -0.16 -0.099 0.19 0.22 -0.095 0.0037
0.44 0.44 -0.28 1 0.71 0.34 0.073 -0.2 -0.13 0.0057
-0.01 0.14 -0.16 0.71 1 -0.064 -0.11 0.032 0.008 0.0077
0.71 0.59 -0.099 0.34 -0.064 1 0.32 -0.75 -0.45 -0.016
0.19 -0.26 0.19 0.073 -0.11 0.32 1 -0.2 7.8e-05 -0.0028
-0.6 -0.41 0.22 -0.2 0.032 -0.75 -0.2 1 0.38 0.035
-0.22 -0.23 -0.095 -0.13 0.008 -0.45 7.8e-05 0.38 1 0.49
-0.017 -0.0094 0.0037 0.0057 0.0077 -0.016 -0.0028 0.035 0.49 1
Interest_Rate
of_property_Introduced
Unemployment
Inflation
Annual Change
Mortgage_Rate
House_Supply
personal_saving_rate
property tax
month
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [365]:
```
scaled.describe()
```

Out[365]:

erest_Rate	No_of_property_Introduced	Unemployment	Inflation	Annual Change	Mortgage_Rate	House_Supply	personal_saving_rate
221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000	221.000000
0.483946	0.062768	0.191176	0.023648	-0.111016	0.132342	0.277602	0.003460
0.848278	0.661469	0.658101	0.692696	0.945216	0.552328	0.954122	1.286464
-0.186170	-0.945701	-0.714286	-1.469031	-2.830189	-0.782885	-1.100000	-2.470588
-0.154255	-0.496229	-0.285714	-0.393203	-0.572327	-0.269042	-0.350000	-0.529412
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.845745	0.503771	0.714286	0.606797	0.427673	0.730958	0.650000	0.470588
2.579787	1.656109	1.607143	1.517612	1.981132	1.032535	3.350000	4.941176

In [ ]:
```

```

Train Test Split

In [366]:
```
from sklearn.model_selection import train_test_split
```

In [367]:
```
x_train, x_test, y_train, y_test = train_test_split(scaled, y, test_size=0.20, random_state=24)
```

In [368]:
```
x_train.shape
```

Out[368]: (176, 10)

In [369]:
```
y_train.shape
```

Out[369]: (176,)

Model

In [370]:
```
from sklearn.linear_model import LinearRegression
```

In [371]:
```
lr = LinearRegression()
```

In [372]:
```
lr.fit(x_train, y_train)
```

Out[372]:
```
LinearRegression
LinearRegression()
```

In [373]:
```
train_pred = lr.predict(x_train)
```

In [374]:
```
test_pred = lr.predict(x_test)
```

In [ ]:
```

```

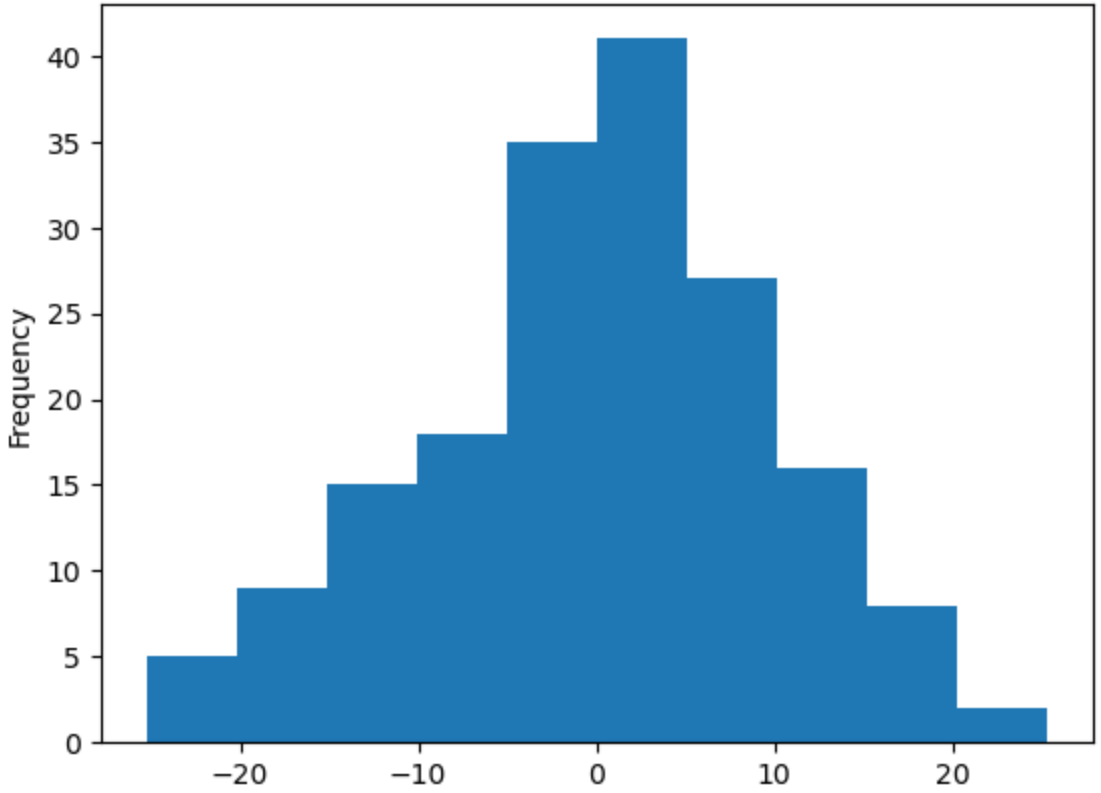
Assumption (residuals normal distribution)

In [375]:
```
from scipy.stats import shapiro
import numpy as np
```

In [376]:
```
res = train_pred - y_train
```

In [377]:
```
res.plot(kind='hist')
```

Out[377]:
```
<Axes: ylabel='Frequency'>
```



In [378]:
```
statistic, p_value = shapiro(res)

alpha = 0.05
if p_value > alpha:
    print("Gaussian (fail to reject H0)")
else:
    print("not look Gaussian (reject H0)")
```

```
Gaussian (fail to reject H0)
```

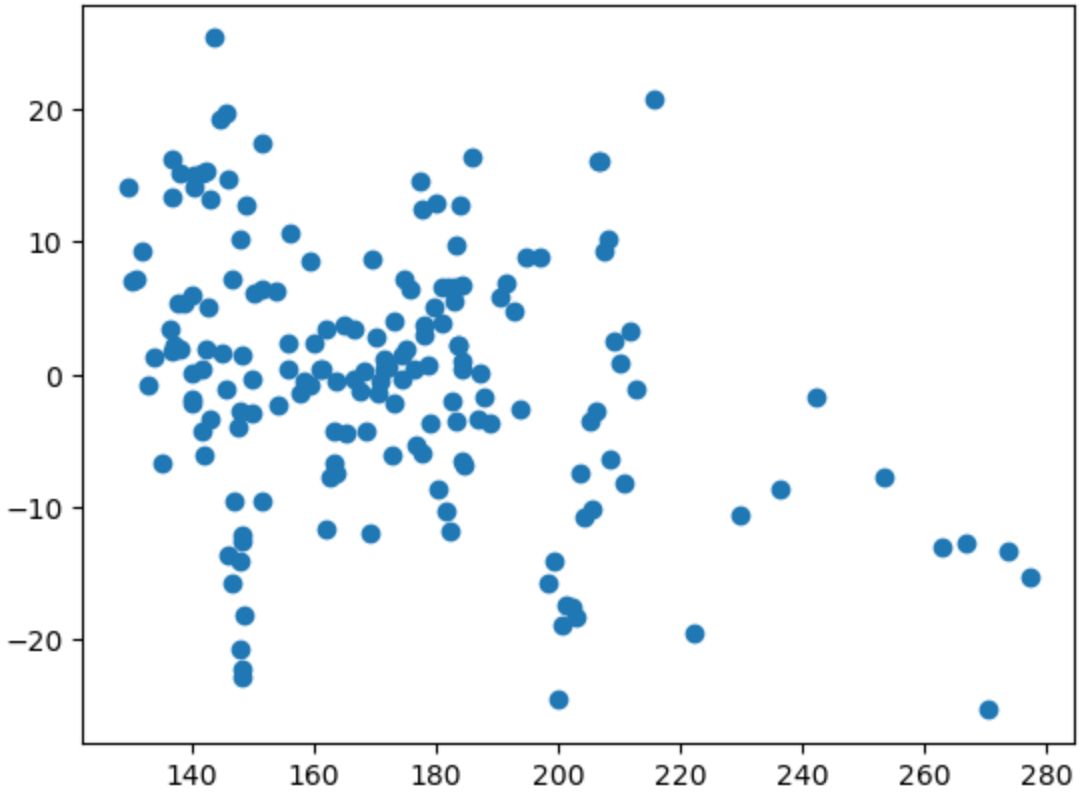
In [ ]:
```

```

Assumption Homosedasticity

In [379]:
```
sd = pd.DataFrame()

sd['dep'] = y_train
sd['resid'] = res
```

In [380]:
```
plt.scatter(sd['dep'], sd['resid'])
```

Out[380]:
```
<matplotlib.collections.PathCollection at 0x21289799010>
```



Assumption autocorrelation(Durbin-Watson test)

In [381]:
```
durbin_ = sm.stats.durbin_watson(res)
durbin_
```

Out[381]:
```
1.989750055567387
```

```
# it is very close to 2 which means there is no problem of autocorrealtion
```

In [ ]:

```
```

Accuracy check----------

In [382]:

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

r2_score

In [383]:

```
acc = r2_score(train_pred, y_train)
acc
```

Out[383]: 0.8832061229329274

In [384]:

```
acct = r2_score(test_pred, y_test)
acct
```

Out[384]: 0.8905504472707525

mean_squared_error

In [385]:

```
accm = mean_squared_error(train_pred, y_train)
accm
```

Out[385]: 95.07169892546003

In [386]:
```
acctm = mean_squared_error(test_pred, y_test)
acctm
```

Out[386]: 103.50325647999124

In [ ]:
```

```

mean_absolute_error

In [387]:
```
accma = mean_absolute_error(train_pred, y_train)
accma
```

Out[387]: 7.5371449361832985

In [388]:
```
acca = mean_absolute_error(test_pred, y_test)
acca
```

Out[388]: 8.043409646658155

In [ ]:
```

```

Cross Validation

In [389]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

In [390]:
```
num_folds = 5

kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)

cross_val_results = cross_val_score(lr,scaled, y, cv=kf)
```

In [391]:
```
cross_val_results
```

Out[391]:
```
array([0.88420572, 0.86002032, 0.85658903, 0.81263493, 0.90882697])
```

In [392]:
```
cross_val_results.mean()
```

Out[392]:
```
0.8644553958662909
```

In [ ]:

In [ ]:

Gradient Boosting

In [393]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [394]:
```
gbr = GradientBoostingRegressor(learning_rate=0.1)
```

In [395]:
```
gbr.fit(x_train, y_train)
```

Out[395]:
```
GradientBoostingRegressor
GradientBoostingRegressor()
```

In [396]:
```
predict_train = gbr.predict(x_train)
predict_test = gbr.predict(x_test)
```

In [ ]:
```

```

Accuracy check

In [ ]:
```

```

R2_score

In [397]:
```
accg = r2_score(predict_train, y_train)
acctg = r2_score(predict_test, y_test)
print('train',accg, 'test', acctg)
```

```
train 0.9987196049313991 test 0.9758182514399932
```

In [ ]:
```

```

Mean_squred_error

In [398]:
```
accm = mean_squared_error(predict_train, y_train)
accm = mean_squared_error(predict_test, y_test)
print('train',accm, 'test', accm)
```

```
train 23.64170819535334 test 23.64170819535334
```

In [ ]:

mean_absolute_error

In [399]:
```
accma = mean_absolute_error(predict_train, y_train)
accma = mean_absolute_error(predict_test, y_test)
print('train',accma, 'test', accma)
```

```
train 3.302403434070066 test 3.302403434070066
```

In [ ]:

Cross Validation score

In [400]:
```
num_folds = 5

kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)

cross_val_results = cross_val_score(gbr,scaled, y, cv=kf)
```

In [401]:
```
cross_val_results
```

Out[401]:
```
array([0.99117423, 0.9875801 , 0.98312377, 0.9946526 , 0.97993346])
```

In [402]:
```
cross_val_results.mean()
```

Out[402]:
```
0.9872928312998044
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: