```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv("C:\\Users\\Mohit Lahoti\\OneDrive\\Desktop\\
HomeLLC_csv.csv")
```

all data Fatch from following websites

https://www.macrotrends.net/2604/30-year-fixed-mortgage-rate-chart Interest Rate https://data.worldbank.org/indicator/SP.POP.TOTL?locations=US Population https://data.worldbank.org/indicator/NY.GDP.MKTP.KD.ZG?locations=US GDP https://www.macrotrends.net/countries/USA/united-states/unemployment-rate UNEMPLOYMENT https://data.worldbank.org/indicator/NY.GDP.PCAP.CD?locations=US GDP Per Capita Income https://www.macrotrends.net/countries/USA/united-states/urban-population Increment in Urban Population https://fred.stlouisfed.org/series/CSUSHPISA. = U.S. National Home Price Index

```python
df
```

```
    YEARS  U.S. National Home Price Index  Interest Rates in %  
Population  \
0    2002                      122.279250                 6.50  
287625193
1    2003                      133.731333                 5.82  
290107933
2    2004                      150.440250                 5.84  
292805298
3    2005                      171.737000                 5.87  
295516599
4    2006                      183.447500                 6.41  
298379912
5    2007                      179.918917                 6.34  
301231207
6    2008                      164.057417                 6.07  
304093966
7    2009                      148.545083                 5.04  
306771529
8    2010                      144.674500                 4.69  
309327143
9    2011                      139.259500                 4.44  
311583481
10   2012                      140.993833                 3.65  
313877662
11   2013                      154.520750                 4.02  
316059947
```

```
12   2014                             164.698167                 4.16
318386329
13   2015                             172.181750                 3.84
320738994
14   2016                             180.925500                 3.67
323071755
15   2017                             191.397667                 3.98
325122128
16   2018                             202.476417                 4.56
326838199
17   2019                             209.463333                 3.91
328329953
18   2020                             222.143417                 3.08
331511512
19   2021                             260.045667                 2.99
332031554
20   2022                             298.486750                 5.47
333287557

    GDP Growth in %  Unemployment in %  GDP Per Capita Income  \
0              1.70               5.78               37997.76
1              2.80               5.99               39490.27
2              3.85               5.53               41724.63
3              3.48               5.08               44123.41
4              2.78               4.62               46302.00
5              2.01               4.62               48050.22
6              0.12               5.78               48570.05
7             -2.60               9.25               47194.94
8              2.71               9.63               48650.64
9              1.55               8.95               50065.97
10             2.28               8.07               51784.42
11             1.84               7.37               53291.13
12             2.29               6.17               55123.85
13             2.71               5.28               56762.73
14             1.67               4.87               57866.74
15             2.24               4.36               59907.75
16             2.95               3.90               62823.31
17             2.29               3.67               65120.39
18            -2.77               8.05               63528.63
19             5.95               5.35               70219.47
20             2.06               3.61               76398.59

   Increment in Urban Population
0                    228,400,290
1                    230,876,596
2                    233,532,722
3                    236,200,507
4                    238,999,326
5                    241,795,278
```

```
6                     244,607,104
7                     247,276,259
8                     249,849,720
9                     252,208,133
10                    254,614,421
11                    256,953,576
12                    259,430,732
13                    261,950,744
14                    264,473,000
15                    266,788,716
16                    268,844,029
17                    270,737,596
18                    274,040,676
19                    275,164,510
20                    276,908,634
```

```
df.columns
```

```
Index(['YEARS', 'U.S. National Home Price Index', 'Interest Rates in
%',
       'Population', 'GDP Growth in %', 'Unemployment in %',
       'GDP Per Capita Income', 'Increment in Urban Population'],
      dtype='object')
```

```
df.head()
```

```
   YEARS  U.S. National Home Price Index  Interest Rates in %
Population  \
0   2002                      122.279250                 6.50
287625193
1   2003                      133.731333                 5.82
290107933
2   2004                      150.440250                 5.84
292805298
3   2005                      171.737000                 5.87
295516599
4   2006                      183.447500                 6.41
298379912

   GDP Growth in %  Unemployment in %  GDP Per Capita Income  \
0             1.70               5.78               37997.76
1             2.80               5.99               39490.27
2             3.85               5.53               41724.63
3             3.48               5.08               44123.41
4             2.78               4.62               46302.00

  Increment in Urban Population
0                   228,400,290
1                   230,876,596
2                   233,532,722
```

```
3                    236,200,507
4                    238,999,326
```

```python
df.tail()
```

```
    YEARS  U.S. National Home Price Index  Interest Rates in %  
Population  \
16   2018                      202.476417                 4.56  
326838199   
17   2019                      209.463333                 3.91  
328329953   
18   2020                      222.143417                 3.08  
331511512   
19   2021                      260.045667                 2.99  
332031554   
20   2022                      298.486750                 5.47  
333287557   

    GDP Growth in %  Unemployment in %  GDP Per Capita Income  \
16             2.95               3.90               62823.31   
17             2.29               3.67               65120.39   
18            -2.77               8.05               63528.63   
19             5.95               5.35               70219.47   
20             2.06               3.61               76398.59   

   Increment in Urban Population  
16                   268,844,029  
17                   270,737,596  
18                   274,040,676  
19                   275,164,510  
20                   276,908,634  
```

```python
df.shape
```

```
(21, 8)
```

```python
df.dtypes
```

```
YEARS                               int64
U.S. National Home Price Index    float64
Interest Rates in %               float64
Population                          int64
GDP Growth in %                   float64
Unemployment in %                 float64
GDP Per Capita Income             float64
Increment in Urban Population      object
dtype: object
```

```python
# covert "increment in urban population" data type object type to int
df['Increment in Urban Population'] = df['Increment in Urban 
Population'].str.replace(',', '').astype(int)
```

```
df.dtypes
```

```
YEARS                                  int64
U.S. National Home Price Index       float64
Interest Rates in %                  float64
Population                             int64
GDP Growth in %                      float64
Unemployment in %                    float64
GDP Per Capita Income                float64
Increment in Urban Population          int32
dtype: object
```

```
#checking the null values
df.isnull().sum()
```

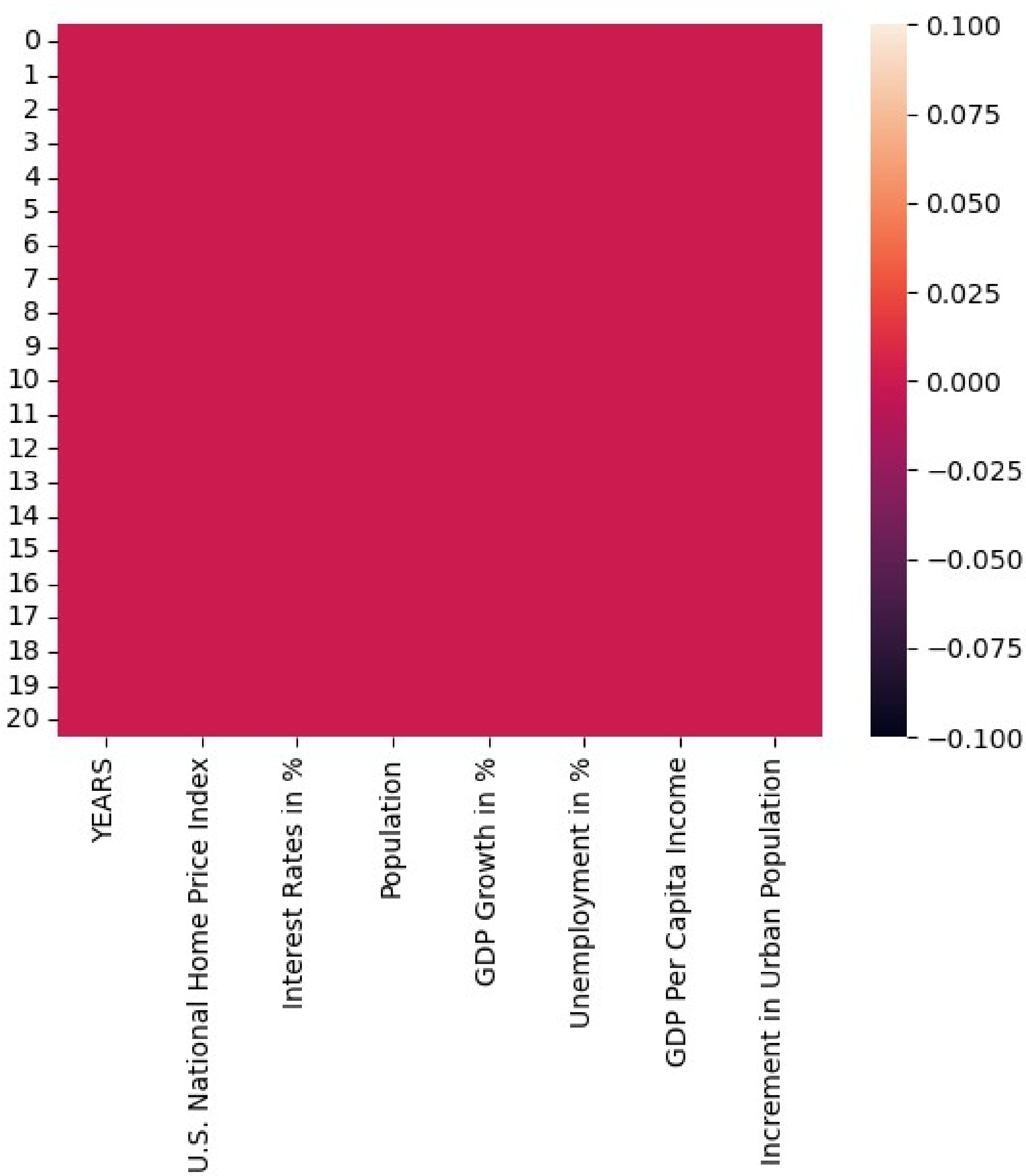
```
YEARS                             0
U.S. National Home Price Index    0
Interest Rates in %               0
Population                        0
GDP Growth in %                   0
Unemployment in %                 0
GDP Per Capita Income             0
Increment in Urban Population     0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21 entries, 0 to 20
Data columns (total 8 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   YEARS                           21 non-null     int64
 1   U.S. National Home Price Index  21 non-null     float64
 2   Interest Rates in %             21 non-null     float64
 3   Population                      21 non-null     int64
 4   GDP Growth in %                 21 non-null     float64
 5   Unemployment in %               21 non-null     float64
 6   GDP Per Capita Income           21 non-null     float64
 7   Increment in Urban Population   21 non-null     int32
dtypes: float64(5), int32(1), int64(2)
memory usage: 1.4 KB
```

```
#check null value by using heatmap
sns.heatmap(df.isnull())
```

```
<Axes: >
```




```
#cheking no.of unique values in each column
df.nunique()
```

```
YEARS                             21
U.S. National Home Price Index    21
Interest Rates in %               21
Population                        21
GDP Growth in %                   19
Unemployment in %                 19
GDP Per Capita Income             21
Increment in Urban Population     21
dtype: int64
```

```
# checking the value counts of each column
for col in df.columns:
    print(df[col].value_counts())
    print('\n')
```

```
2002    1
2013    1
2021    1
2020    1
2019    1
2018    1
2017    1
2016    1
2015    1
2014    1
2012    1
2003    1
2011    1
2010    1
2009    1
2008    1
2007    1
2006    1
2005    1
2004    1
2022    1
Name: YEARS, dtype: int64


122.279250    1
154.520750    1
260.045667    1
222.143417    1
209.463333    1
202.476417    1
191.397667    1
180.925500    1
172.181750    1
164.698167    1
140.993833    1
133.731333    1
139.259500    1
144.674500    1
148.545083    1
164.057417    1
179.918917    1
183.447500    1
171.737000    1
150.440250    1
298.486750    1
```

```
Name: U.S. National Home Price Index, dtype: int64


6.50    1
4.02    1
2.99    1
3.08    1
3.91    1
4.56    1
3.98    1
3.67    1
3.84    1
4.16    1
3.65    1
5.82    1
4.44    1
4.69    1
5.04    1
6.07    1
6.34    1
6.41    1
5.87    1
5.84    1
5.47    1
Name: Interest Rates in %, dtype: int64


287625193    1
316059947    1
332031554    1
331511512    1
328329953    1
326838199    1
325122128    1
323071755    1
320738994    1
318386329    1
313877662    1
290107933    1
311583481    1
309327143    1
306771529    1
304093966    1
301231207    1
298379912    1
295516599    1
292805298    1
333287557    1
Name: Population, dtype: int64
```

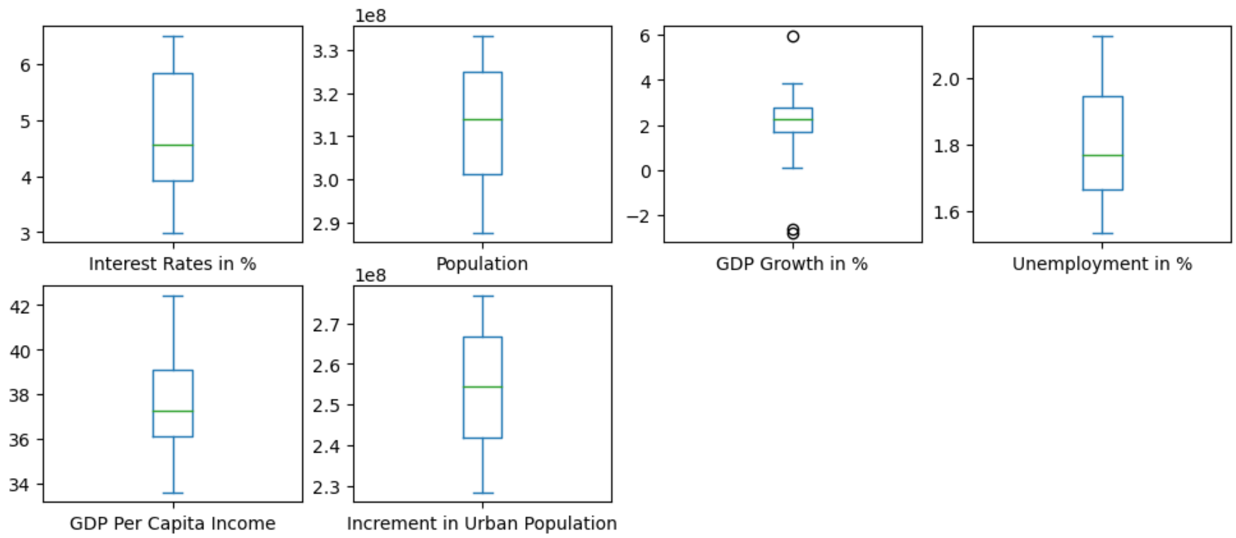
```
 2.29    2
 2.71    2
 1.70    1
 2.28    1
 5.95    1
-2.77    1
 2.95    1
 2.24    1
 1.67    1
 1.84    1
 1.55    1
 2.80    1
-2.60    1
 0.12    1
 2.01    1
 2.78    1
 3.48    1
 3.85    1
 2.06    1
Name: GDP Growth in %, dtype: int64


5.78    2
4.62    2
6.17    1
5.35    1
8.05    1
3.67    1
3.90    1
4.36    1
4.87    1
5.28    1
7.37    1
5.99    1
8.07    1
8.95    1
9.63    1
9.25    1
5.08    1
5.53    1
3.61    1
Name: Unemployment in %, dtype: int64


37997.76    1
53291.13    1
70219.47    1
63528.63    1
65120.39    1
```

```
62823.31    1
59907.75    1
57866.74    1
56762.73    1
55123.85    1
51784.42    1
39490.27    1
50065.97    1
48650.64    1
47194.94    1
48570.05    1
48050.22    1
46302.00    1
44123.41    1
41724.63    1
76398.59    1
Name: GDP Per Capita Income, dtype: int64


228400290    1
256953576    1
275164510    1
274040676    1
270737596    1
268844029    1
266788716    1
264473000    1
261950744    1
259430732    1
254614421    1
230876596    1
252208133    1
249849720    1
247276259    1
244607104    1
241795278    1
238999326    1
236200507    1
233532722    1
276908634    1
Name: Increment in Urban Population, dtype: int64
```

```python
# Checking for duplicate values
print("Number of duplicate rows:", df.duplicated().sum())
```

```
Number of duplicate rows: 0
```

```
Num_cols = ['Interest Rates in %','Population', 'GDP Growth in %',
'Unemployment in %',
       'GDP Per Capita Income', 'Increment in Urban Population']

# Create a new DataFrame containing only the Required columns
num_df = df[Num_cols]

# Getting statistical summary of numerical columns
description = num_df.describe()
description

       Interest Rates in %    Population  GDP Growth in %
Unemployment in %  \
count            21.000000  2.100000e+01        21.000000
21.000000
mean              4.778571  3.126999e+08         1.995714
1.799356
std               1.132865  1.458729e+07         1.904683
0.181174
min               2.990000  2.876252e+08        -2.770000
1.534037
25%               3.910000  3.012312e+08         1.700000
1.665510
50%               4.560000  3.138777e+08         2.280000
1.768378
75%               5.840000  3.251221e+08         2.780000
1.946058
max               6.500000  3.332876e+08         5.950000
2.127529

       GDP Per Capita Income  Increment in Urban Population
count              21.000000                   2.100000e+01
mean               37.558122                   2.539835e+08
std                 2.341360                   1.533544e+07
min                33.619093                   2.284003e+08
25%                36.138086                   2.417953e+08
50%                37.273460                   2.546144e+08
75%                39.128602                   2.667887e+08
max                42.432158                   2.769086e+08

#check for outliers
df_outliers = df[Num_cols]
df_outliers.plot(kind='box', subplots=True, layout=(7, 4),
fontsize=10, figsize=(12, 18))
plt.show()
```




```
#remove outliers with z-score method
from scipy.stats import zscore
out_df = df[Num_cols]
z = np.abs(zscore(out_df))
z
```

```
    Interest Rates in %  Population  GDP Growth in %  Unemployment in 
%  \
0              1.557061    1.761392         0.159091            
0.026705
1              0.941989    1.586990         0.432696            
0.094762
2              0.960080    1.397511         0.997582            
0.175207
3              0.987215    1.207054         0.798527            
0.454211
4              1.475654    1.005918         0.421936            
0.757010
5              1.412338    0.805627         0.007686            
0.757010
6              1.168119    0.604530         1.009111            
0.026705
7              0.236466    0.416443         2.472436            
1.695694
8              0.080114    0.236922         0.384277            
1.856096
9              0.306243    0.078424         0.239789            
1.565928
10             1.020812    0.082733         0.152942            
1.167710
11             0.686140    0.236029         0.083772            
0.829727
12             0.559508    0.399448         0.158322
```

```
0.196636
13              0.848953    0.564712         0.384277
0.328256
14              1.002721    0.728579         0.175230
0.590075
15              0.722321    0.872609         0.131423
0.937140
16              0.197701    0.993156         0.513394
1.274234
17              0.785637    1.097945         0.158322
1.452801
18              1.536386    1.321436         2.563894
1.158330
19              1.617793    1.357967         2.127356
0.284924
20              0.625409    1.446196         0.034585
1.500605

    GDP Per Capita Income  Increment in Urban Population
0                1.723914                       1.709435
1                1.533740                       1.543972
2                1.257800                       1.366493
3                0.972310                       1.188234
4                0.721860                       1.001221
5                0.526510                       0.814399
6                0.469339                       0.626516
7                0.621478                       0.448167
8                0.460512                       0.276211
9                0.307060                       0.118625
10               0.124582                       0.042160
11               0.032118                       0.198460
12               0.218793                       0.363980
13               0.382250                       0.532364
14               0.490595                       0.700898
15               0.687320                       0.855632
16               0.960735                       0.992965
17               1.170250                       1.119491
18               1.025603                       1.340198
19               1.618322                       1.415292
20               2.133119                       1.531832

#threshold value = 3
z_score_threshold = 3

outliers = np.any(z > z_score_threshold, axis=1)

df_out = df[~outliers]

df_out.shape
```

```
(21, 8)
```

```
#shape of old and new data frame
print("old df_",df.shape[0])
print("New df_",df_out.shape[0])
```

```
old df_ 21
New df_ 21
```

```
#view data loss by persentage
print("data loss percentage_",((df.shape[0]-
df_out.shape[0])/df.shape[0])*100)
```

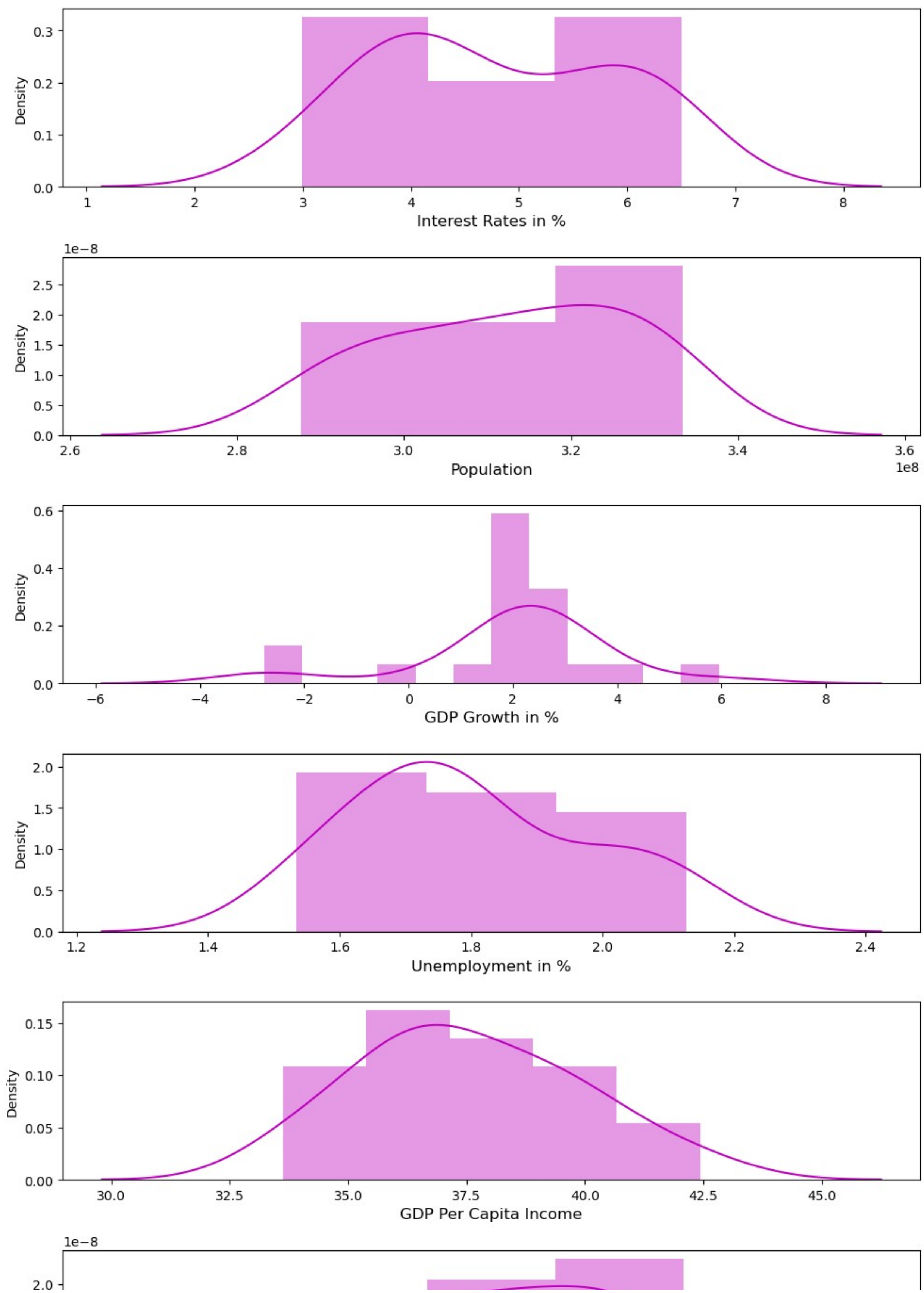
```
data loss percentage_ 0.0
```

```
df=df_out
```

```
# check skewness
df.skew()
```

```
YEARS                                  0.000000
U.S. National Home Price Index         1.000723
Interest Rates in %                    0.123205
Population                            -0.223382
GDP Growth in %                       -1.102806
Unemployment in %                      0.424672
GDP Per Capita Income                  0.276714
Increment in Urban Population         -0.123855
dtype: float64
```

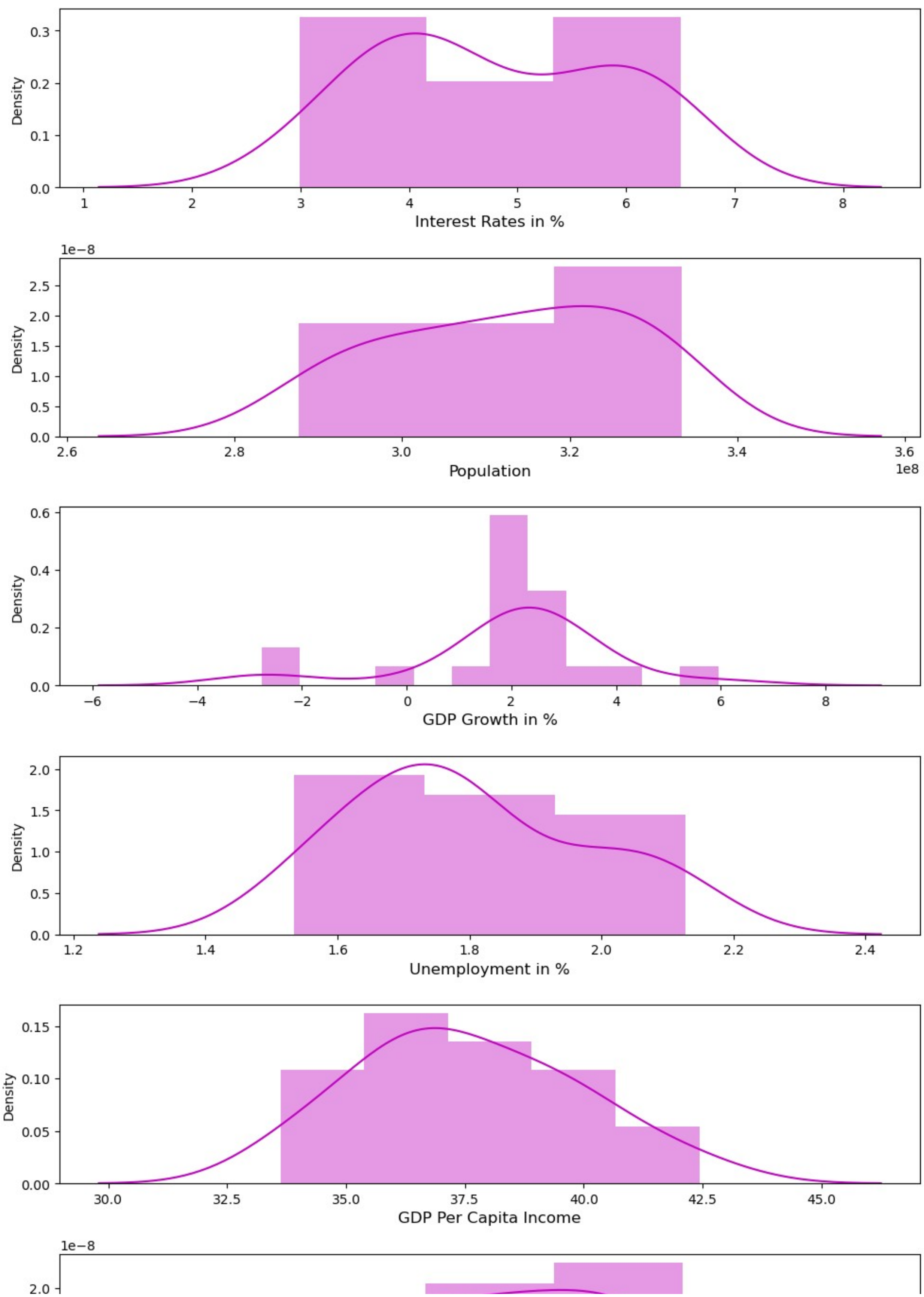
```
#lets check how the data hase been distributed in Required columns
plt.figure(figsize=(10, 18))
plotnumber = 1
for col in Num_cols:
    if plotnumber <=7:
        plt.subplot(7, 1, plotnumber)
        sns.distplot(df[col], color="m")
        plt.xlabel(col, fontsize=12)
        plt.xticks(rotation=0, fontsize=10)
        plt.yticks(rotation=0, fontsize=10)
        plotnumber += 1

plt.tight_layout()
plt.show()
```


Density
0.0
0.1
0.2
0.3
1
2
3
4
5
6
7
8
Interest Rates in %
1e−8
Density
0.0
0.5
1.0
1.5
2.0
2.5
2.6
2.8
3.0
3.2
3.4
3.6
1e8
Population
Density
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6
8
GDP Growth in %
Density
0.0
0.5
1.0
1.5
2.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
Unemployment in %
Density
0.00
0.05
0.10
0.15
30.0
32.5
35.0
37.5
40.0
42.5
45.0
GDP Per Capita Income
1e−8
2.0

```
# apply cube root transformation to reduce skewness
for col in Num_cols:
    if df[col].skew() > 0.5:
        df[col] = np.cbrt(df[col])

#checking skewness again
df.skew()
```

```
YEARS                                 0.000000
U.S. National Home Price Index        1.000723
Interest Rates in %                   0.123205
Population                           -0.223382
GDP Growth in %                      -1.102806
Unemployment in %                     0.424672
GDP Per Capita Income                 0.276714
Increment in Urban Population        -0.123855
dtype: float64
```

```
#lets check again how the data hase been distributed in Required
columns
plt.figure(figsize=(10, 18))
plotnumber = 1
for col in Num_cols:
    if plotnumber <= 7:
        plt.subplot(7, 1, plotnumber)
        sns.distplot(df[col], color="m")
        plt.xlabel(col, fontsize=12)
        plt.xticks(rotation=0, fontsize=10)
        plt.yticks(rotation=0, fontsize=10)
        plotnumber += 1

plt.tight_layout()
plt.show()
```



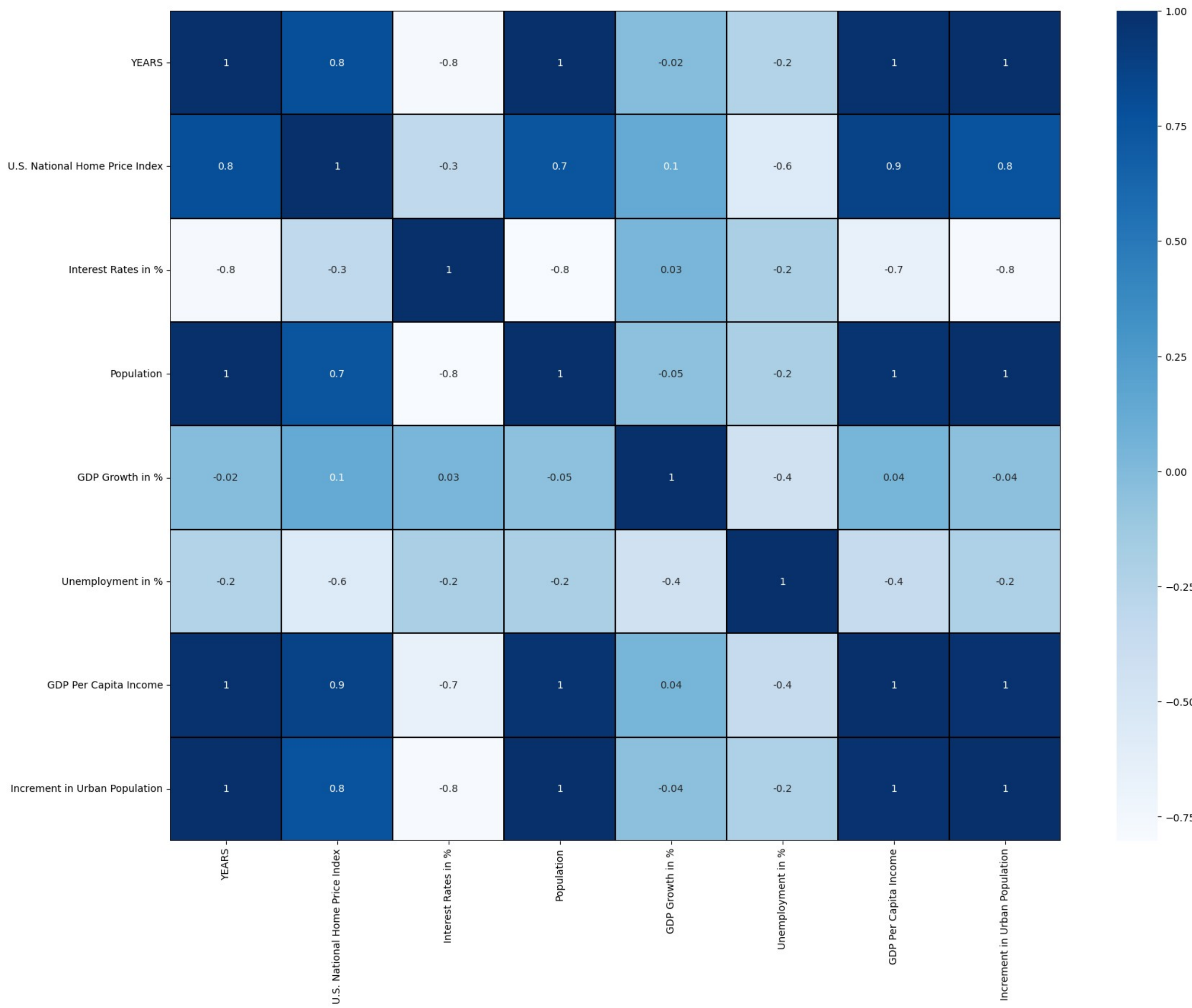
Density
0.0
0.1
0.2
0.3
1
2
3
4
5
6
7
8
Interest Rates in %
1e−8
Density
0.0
0.5
1.0
1.5
2.0
2.5
2.6
2.8
3.0
3.2
3.4
3.6
1e8
Population
Density
0.0
0.2
0.4
0.6
−6
−4
−2
0
2
4
6
8
GDP Growth in %
Density
0.0
0.5
1.0
1.5
2.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
Unemployment in %
Density
0.00
0.05
0.10
0.15
30.0
32.5
35.0
37.5
40.0
42.5
45.0
GDP Per Capita Income
1e−8
2.0

```python
# check correlation
cor = df.corr()
cor
```

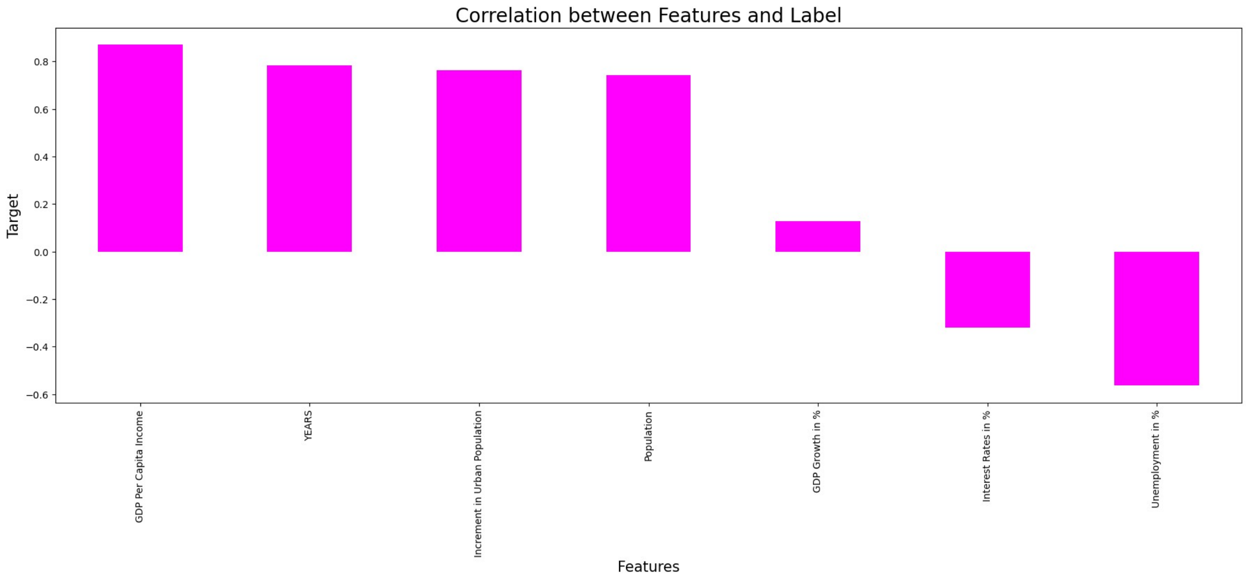
```
                                  YEARS  U.S. National Home Price 
Index  \
YEARS                          1.000000                           
0.783209
U.S. National Home Price Index  0.783209                          
1.000000
Interest Rates in %           -0.776400                          -
0.318690
Population                     0.996496                           
0.743046
GDP Growth in %               -0.023650                           
0.129226
Unemployment in %             -0.242703                          -
0.563685
GDP Per Capita Income          0.982354                           
0.869882
Increment in Urban Population   0.998870                          
0.761370

                                Interest Rates in %  Population  \
YEARS                                     -0.776400    0.996496   
U.S. National Home Price Index            -0.318690    0.743046   
Interest Rates in %                        1.000000   -0.802214   
Population                                -0.802214    1.000000   
GDP Growth in %                            0.032346   -0.053949   
Unemployment in %                         -0.200077   -0.197168   
GDP Per Capita Income                     -0.665183    0.969644   
Increment in Urban Population             -0.791918    0.999322   

                                GDP Growth in %  Unemployment in %  \
YEARS                                 -0.023650          -0.242703   
U.S. National Home Price Index         0.129226          -0.563685   
Interest Rates in %                    0.032346          -0.200077   
Population                            -0.053949          -0.197168   
GDP Growth in %                        1.000000          -0.436333   
Unemployment in %                     -0.436333           1.000000   
GDP Per Capita Income                  0.043205          -0.352498   
Increment in Urban Population         -0.043710          -0.215936   

                                GDP Per Capita Income  \
YEARS                                        0.982354   
U.S. National Home Price Index               0.869882   
Interest Rates in %                         -0.665183   
Population                                   0.969644   
GDP Growth in %                              0.043205   
Unemployment in %                           -0.352498   
```

```
GDP Per Capita Income                                    1.000000
Increment in Urban Population                            0.975693

                                Increment in Urban Population
YEARS                                                0.998870
U.S. National Home Price Index                       0.761370
Interest Rates in %                                 -0.791918
Population                                           0.999322
GDP Growth in %                                     -0.043710
Unemployment in %                                   -0.215936
GDP Per Capita Income                                0.975693
Increment in Urban Population                        1.000000
```

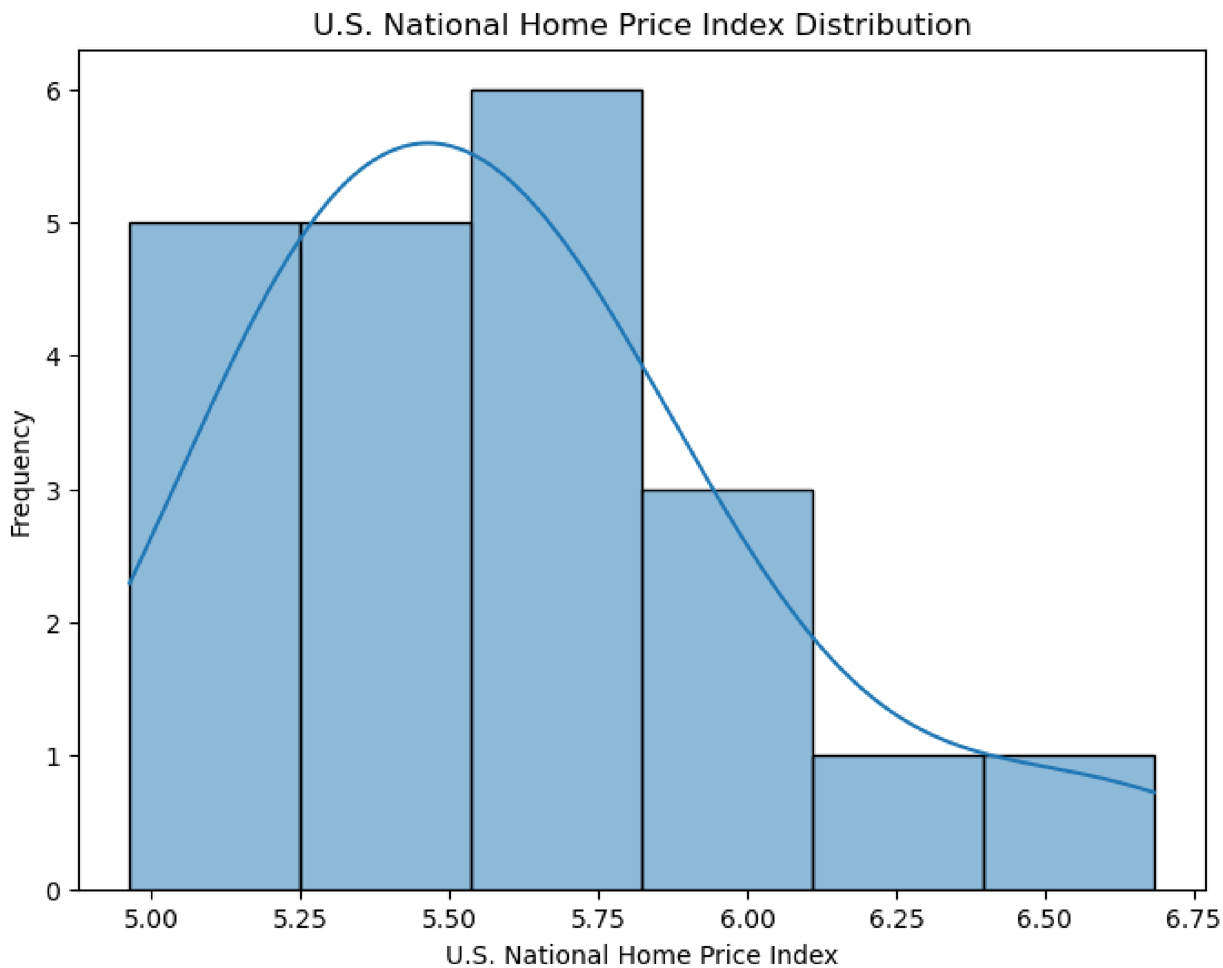
```
# Visualizing the correlation by plotting a heatmap
plt.figure(figsize=(20, 15))
sns.heatmap(df.corr(), linewidths=0.1, fmt=".1g", linecolor="black",
annot=True, cmap="Blues")
plt.yticks(rotation=0)
plt.show()
```



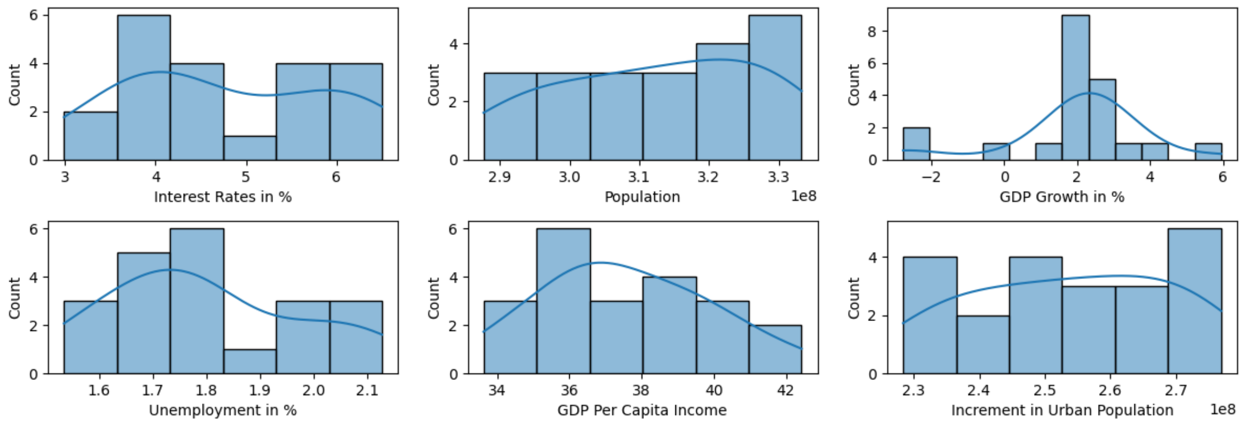

```
#visulaizing the correlation  between label and features using bar
plot
plt.figure(figsize=(22, 7))
df.corr()['U.S. National Home Price
Index'].sort_values(ascending=False).drop(['U.S. National Home Price
Index']).plot(kind='bar', color='magenta')
plt.xlabel('Features', fontsize=15)
plt.ylabel('Target', fontsize=15)
plt.title('Correlation between Features and Label', fontsize=20)
plt.show()
```



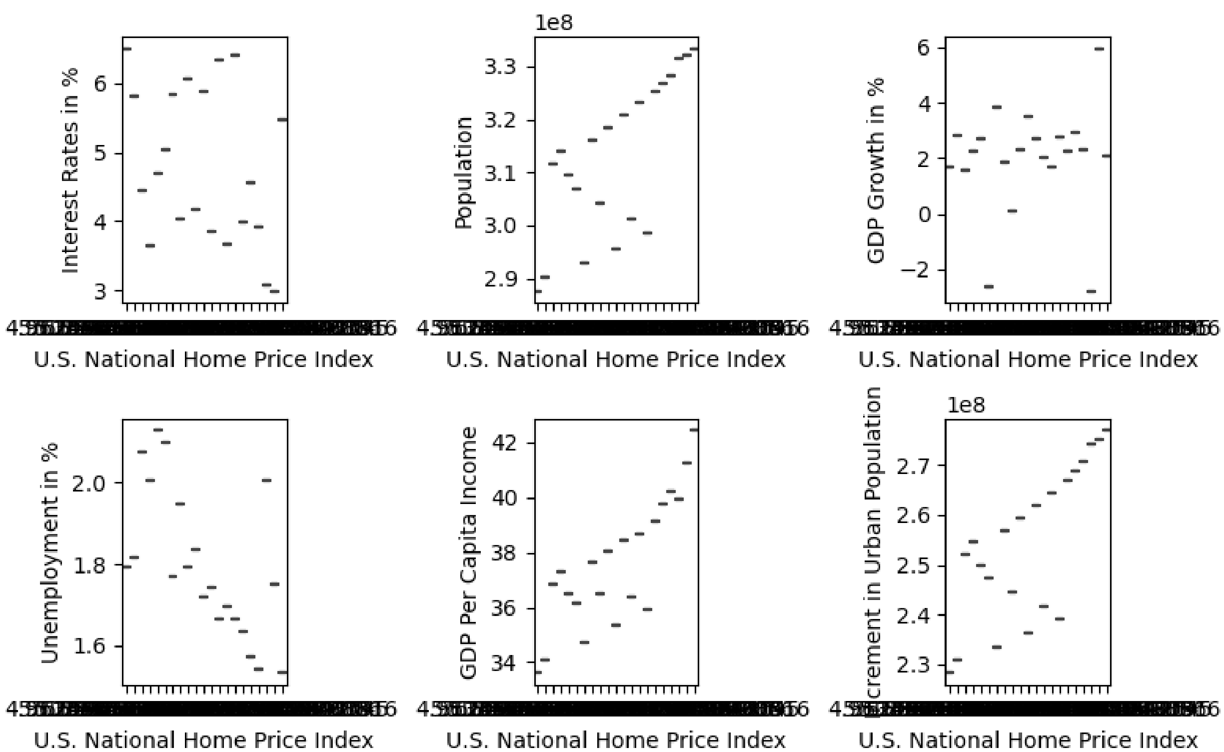

```
# Let's focus on the 'U.S. National Home Price Index' column, which is our target variable
plt.figure(figsize=(8, 6))
sns.histplot(df['U.S. National Home Price Index'], kde=True)
plt.title('U.S. National Home Price Index Distribution')
plt.xlabel('U.S. National Home Price Index')
plt.ylabel('Frequency')
plt.show()
```



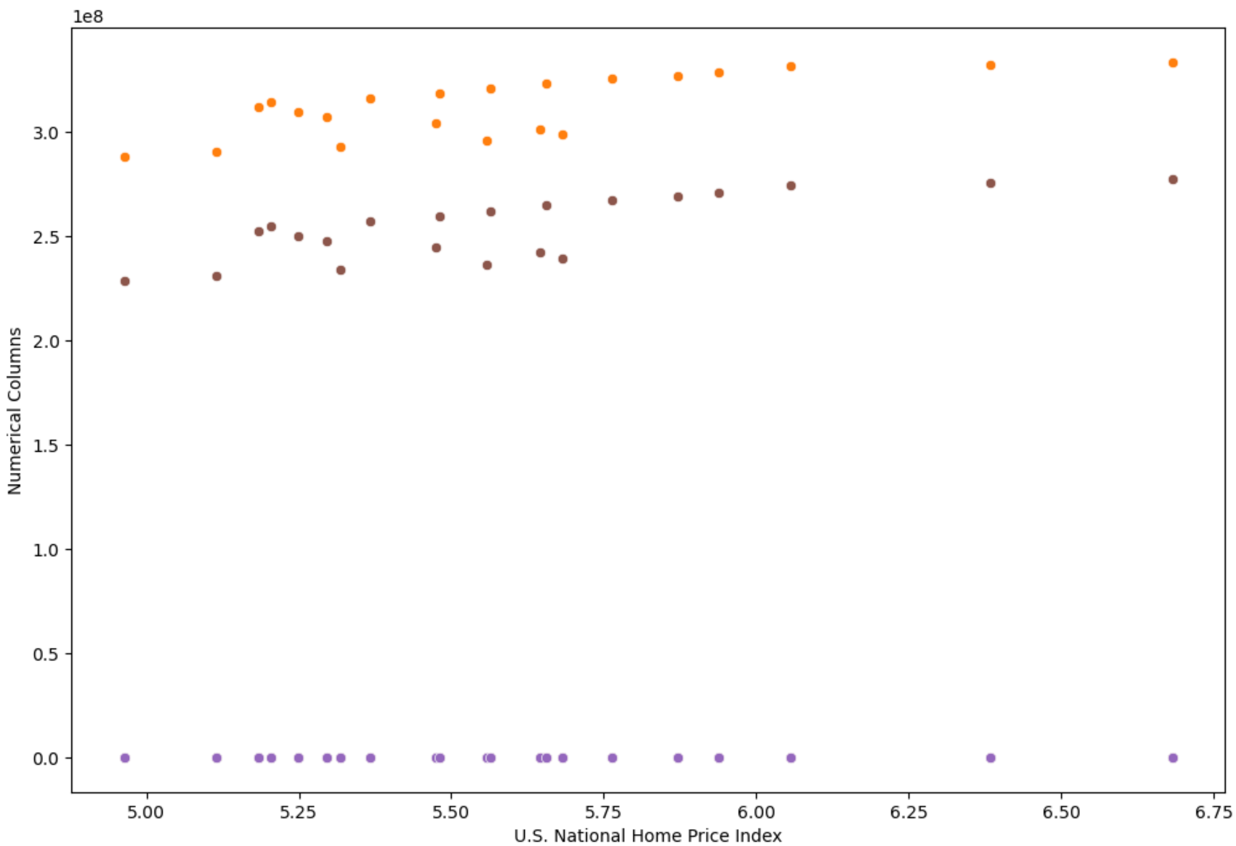

```
# for other  columns
plt.figure(figsize=(12, 18))
for i, col in enumerate(Num_cols, 1):
    plt.subplot(9, 3, i)
    sns.histplot(df[col], kde=True)
    plt.xlabel(col)
    plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



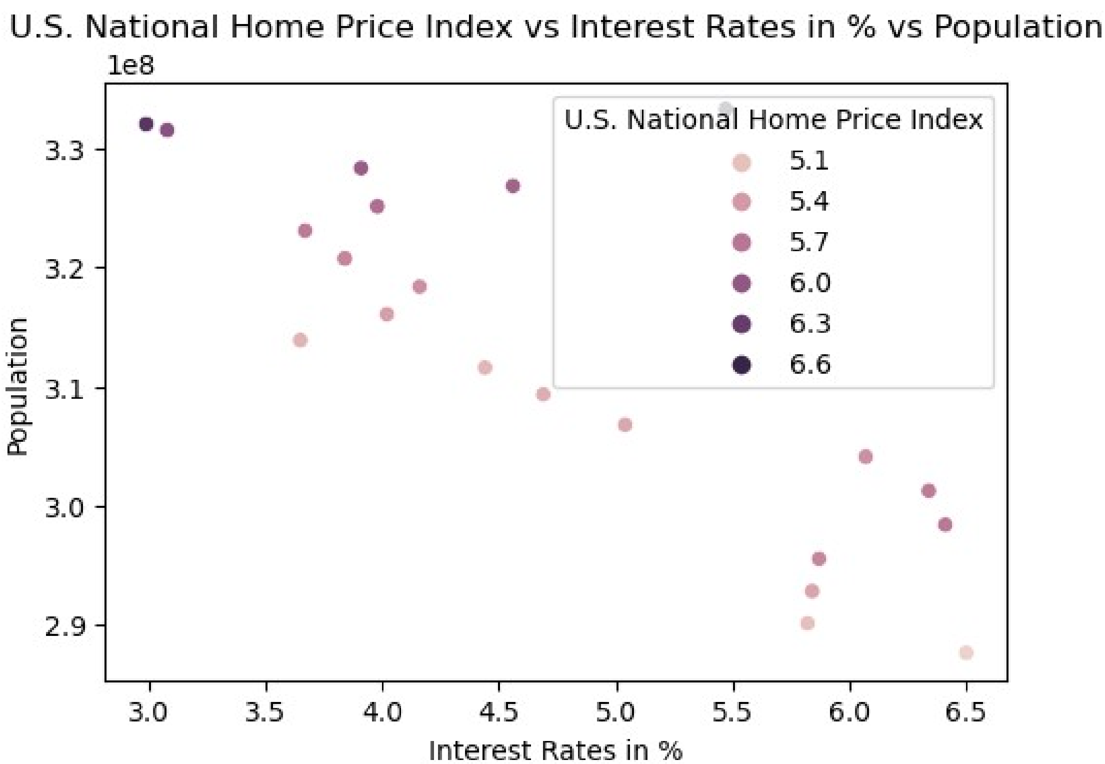

```python
# Relationship between U.S. National Home Price Index and  columns
plt.figure(figsize=(8, 5))
for i, col in enumerate(Num_cols, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x='U.S. National Home Price Index', y=col, data=df)
    plt.xlabel('U.S. National Home Price Index')
    plt.ylabel(col)

plt.tight_layout()
plt.show()
```



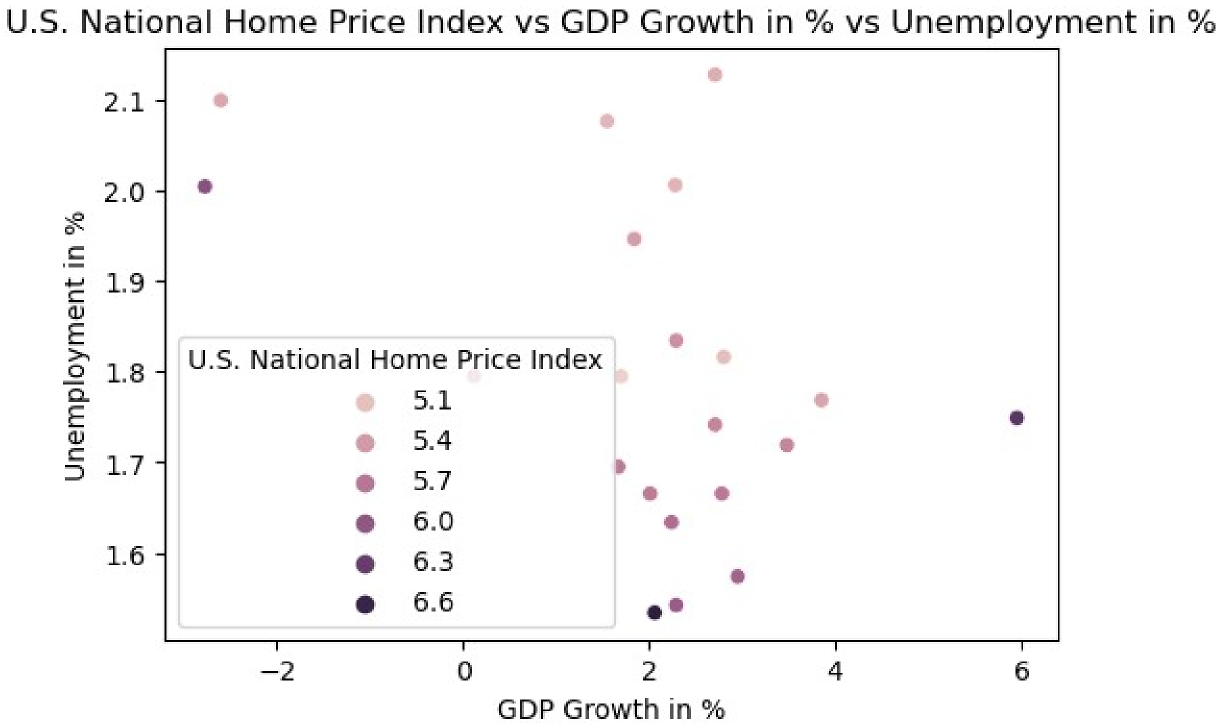
```python
plt.figure(figsize=(12, 8))
for col in Num_cols:
    sns.scatterplot(x='U.S. National Home Price Index', y=col,
data=df)
plt.xlabel('U.S. National Home Price Index')
plt.ylabel('Numerical Columns')
plt.show()
```



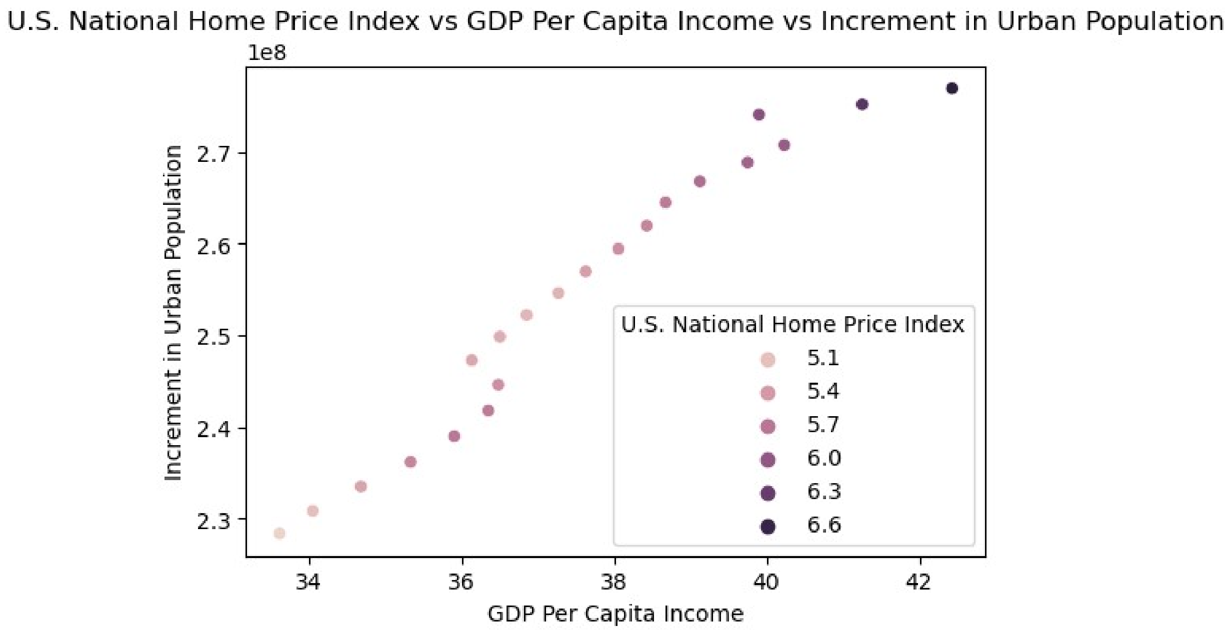

```python
# Relationship between U.S. National Home Price Index, Interest Rates
in %, and Population
plt.figure(figsize=(6, 4))
sns.scatterplot(x='Interest Rates in %', y='Population', hue='U.S.
National Home Price Index', data=df)
plt.xlabel('Interest Rates in %')
plt.ylabel('Population')
plt.title('U.S. National Home Price Index vs Interest Rates in % vs
Population')
plt.show()
```



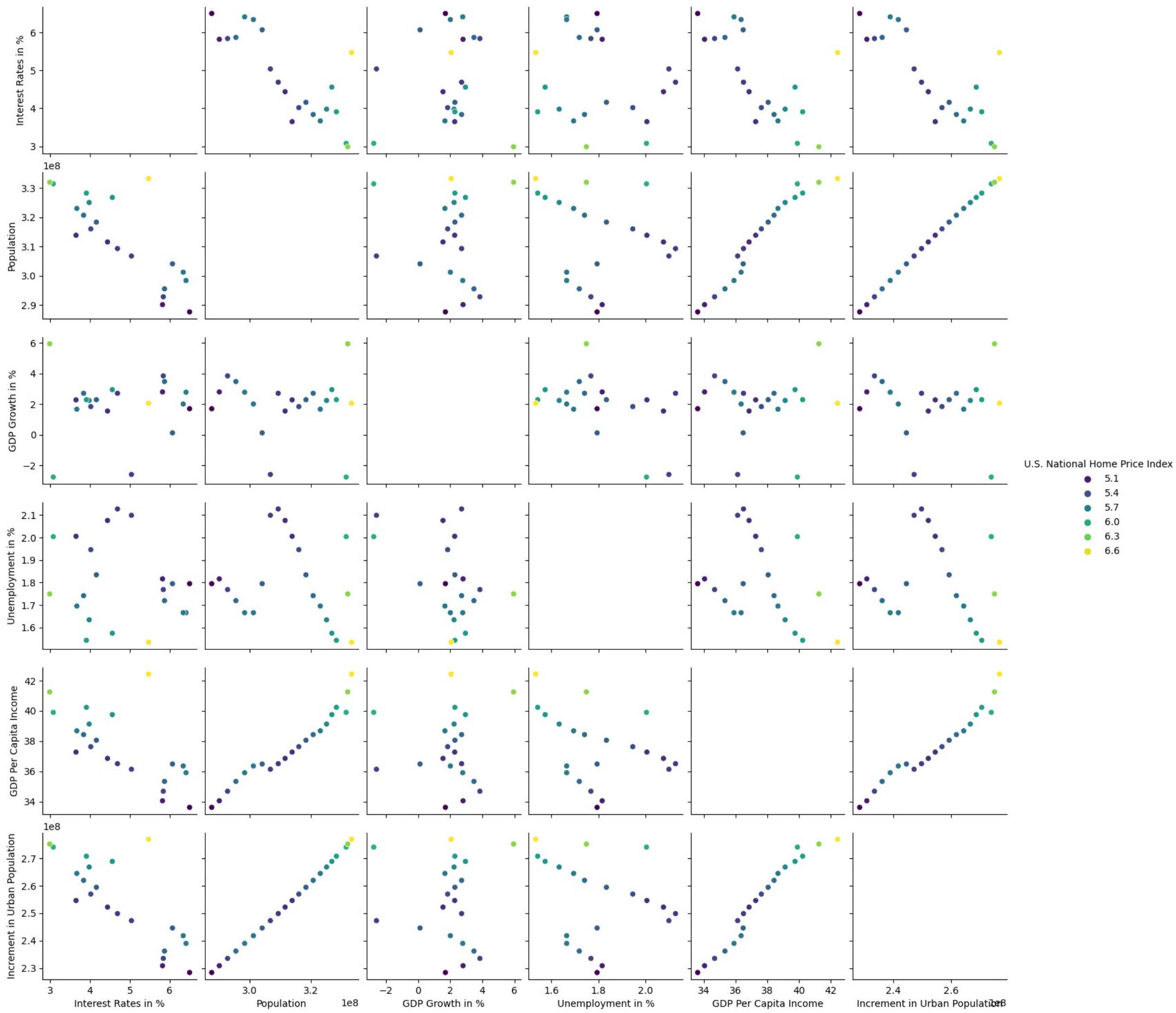

```python
# Relationship between U.S. National Home Price Index, GDP Growth in
%, and Unemployment in %
plt.figure(figsize=(6, 4))
sns.scatterplot(x='GDP Growth in %', y='Unemployment in %', hue='U.S.
National Home Price Index', data=df)
plt.xlabel('GDP Growth in %')
plt.ylabel('Unemployment in %')
plt.title('U.S. National Home Price Index vs GDP Growth in % vs
Unemployment in %')
plt.show()
```




```
# Relationship between U.S. National Home Price Index, GDP Per Capita
Income, and Increment in Urban Population
plt.figure(figsize=(6, 4))
sns.scatterplot(x='GDP Per Capita Income', y='Increment in Urban
Population', hue='U.S. National Home Price Index', data=df)
plt.xlabel('GDP Per Capita Income')
plt.ylabel('Increment in Urban Population')
plt.title('U.S. National Home Price Index vs GDP Per Capita Income vs
Increment in Urban Population')
plt.show()
```




```
#pairplot
df_subset = df[['U.S. National Home Price Index'] + Num_cols]
sns.pairplot(df_subset, hue='U.S. National Home Price Index',
diag_kind='kde',palette='viridis')
plt.show()
```




```
#Separating features and label
X_reg = df.drop('U.S. National Home Price Index',axis=1)
y_reg= df['U.S. National Home Price Index']

#Feature Scalling using standard scalarization
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_reg = pd.DataFrame(scaler.fit_transform(X_reg),
columns=X_reg.columns)
X_reg
```

	YEARS	Interest Rates in %	Population	GDP Growth in % \
0	-1.651446	1.557061	-1.761392	-0.159091
1	-1.486301	0.941989	-1.586990	0.432696
2	-1.321157	0.960080	-1.397511	0.997582
3	-1.156012	0.987215	-1.207054	0.798527
4	-0.990867	1.475654	-1.005918	0.421936

5	-0.825723	1.412338	-0.805627	0.007686
6	-0.660578	1.168119	-0.604530	-1.009111
7	-0.495434	0.236466	-0.416443	-2.472436
8	-0.330289	-0.080114	-0.236922	0.384277
9	-0.165145	-0.306243	-0.078424	-0.239789
10	0.000000	-1.020812	0.082733	0.152942
11	0.165145	-0.686140	0.236029	-0.083772
12	0.330289	-0.559508	0.399448	0.158322
13	0.495434	-0.848953	0.564712	0.384277
14	0.660578	-1.002721	0.728579	-0.175230
15	0.825723	-0.722321	0.872609	0.131423
16	0.990867	-0.197701	0.993156	0.513394
17	1.156012	-0.785637	1.097945	0.158322
18	1.321157	-1.536386	1.321436	-2.563894
19	1.486301	-1.617793	1.357967	2.127356
20	1.651446	0.625409	1.446196	0.034585

	Unemployment in %	GDP Per Capita Income	Increment in Urban Population
0	-0.026705	-1.723914	-1.709435
1	0.094762	-1.533740	-1.543972
2	-0.175207	-1.257800	-1.366493
3	-0.454211	-0.972310	-1.188234
4	-0.757010	-0.721860	-1.001221
5	-0.757010	-0.526510	-0.814399
6	-0.026705	-0.469339	-0.626516
7	1.695694	-0.621478	-0.448167
8	1.856096	-0.460512	-0.276211
9	1.565928	-0.307060	-0.118625
10	1.167710	-0.124582	0.042160
11	0.829727	0.032118	0.198460
12	0.196636	0.218793	0.363980
13	-0.328256	0.382250	0.532364
14	-0.590075	0.490595	0.700898

```
15           -0.937140              0.687320
0.855632
16           -1.274234              0.960735
0.992965
17           -1.452801              1.170250
1.119491
18            1.158330              1.025603
1.340198
19           -0.284924              1.618322
1.415292
20           -1.500605              2.133119
1.531832
```

```
#checking  variance inflation factor (vif)
from statsmodels.stats.outliers_influence import
variance_inflation_factor

Vif = pd.DataFrame()
Vif['VIF Values'] = [variance_inflation_factor(X_reg.values, i) for i
in range(len(X_reg.columns))]
Vif['Features'] = X_reg.columns
Vif
```

	VIF Values	Features
0	16722.155094	YEARS
1	15.563145	Interest Rates in %
2	21332.792341	Population
3	3.058715	GDP Growth in %
4	2.763393	Unemployment in %
5	154.438314	GDP Per Capita Income
6	72763.554418	Increment in Urban Population

```
y_reg.value_counts()
```

```
4.963457    1
5.366143    1
6.382878    1
6.056353    1
5.938854    1
5.872073    1
5.762959    1
5.655877    1
5.563256    1
5.481460    1
5.204752    1
5.113808    1
5.183323    1
5.249654    1
5.296058    1
5.474342    1
```

```
5.645368    1
5.682035    1
5.558462    1
5.318486    1
6.683055    1
Name: U.S. National Home Price Index, dtype: int64
```

Modelling

```
#import liabraries for modelling process
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import LinearRegression

#finding the best random state

MaxAccu = 0
MaxRs = 0

for i in range(1, 200):
    X_reg_train, X_reg_test, y_reg_train, y_reg_test =
train_test_split(X_reg, y_reg, test_size=0.3, random_state=i)
    lr = LinearRegression()
    lr.fit(X_reg_train, y_reg_train)
    pred = lr.predict(X_reg_test)
    acc = r2_score(y_reg_test, pred)
    if acc > MaxAccu:
        MaxAccu = acc
        MaxRs = i

print("Maximum R2 score is", MaxAccu, "on Random State", MaxRs)

Maximum R2 score is 0.9854899827046829 on Random State 163

X_reg_train, X_reg_test, y_reg_train,y_reg_test =
train_test_split(X_reg, y_reg, test_size=0.3, random_state=163)

#modelling
#import liabraries
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
```

```
lr.fit(X_reg_train,y_reg_train)
pred_lr = lr.predict(X_reg_test)
pred_train = lr.predict(X_reg_train)

print("R2_score:",r2_score(y_reg_test,pred_lr))
print("R2_score on training
Data:",r2_score(y_reg_train,pred_train)*100)
print ("MeanAbsoluteError:",mean_absolute_error(y_reg_test,pred_lr))
print ("MeanSquaredError:",mean_squared_error(y_reg_test,pred_lr))
print ("Root Mean Squared
Error:",np.sqrt(mean_squared_error(y_reg_test,pred_lr)))
```

```
R2_score: 0.9854899827046829
R2_score on training Data: 92.89150072260882
MeanAbsoluteError: 0.047339378805776225
MeanSquaredError: 0.004227293218151316
Root Mean Squared Error: 0.0650176377466247
```

```
Rd = Ridge()
Rd.fit(X_reg_train, y_reg_train)
pred_Rd = Rd.predict(X_reg_test)
pred_train = Rd.predict(X_reg_train)

print("R2_score:", r2_score(y_reg_test, pred_Rd))
print("R2_score on training Data:", r2_score(y_reg_train,
pred_train)*100)
print("Mean Absolute Error:", mean_absolute_error(y_reg_test,
pred_Rd))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred_Rd))
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred_Rd)))
```

```
R2_score: 0.7628615659194506
R2_score on training Data: 78.43679696905279
Mean Absolute Error: 0.18786969991787103
Mean Squared Error: 0.06908700890902827
Root Mean Squared Error: 0.26284407718080366
```

```
Dtr = DecisionTreeRegressor()
Dtr.fit(X_reg_train, y_reg_train)
pred_Dtr = Dtr.predict(X_reg_test)
pred_train = Dtr.predict(X_reg_train)

print("R2_score:", r2_score(y_reg_test, pred_Dtr))
print("R2_score on training Data:", r2_score(y_reg_train, pred_train))
print("Mean Absolute Error:", mean_absolute_error(y_reg_test,
pred_Dtr))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred_Dtr))
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred_Dtr)))
```

```
R2_score: 0.7286774618572862
R2_score on training Data: 1.0
Mean Absolute Error: 0.20473718121802875
Mean Squared Error: 0.07904607569230519
Root Mean Squared Error: 0.28115133948161297
```

```
Rfr = RandomForestRegressor()
Rfr.fit(X_reg_train, y_reg_train)
pred_Rfr = Rfr.predict(X_reg_test)
pred_train = Rfr.predict(X_reg_train)

print("R2_score:", r2_score(y_reg_test, pred_Rfr))
print("R2_score on training Data:", r2_score(y_reg_train,
pred_train)*100)
print("Mean Absolute Error:", mean_absolute_error(y_reg_test,
pred_Rfr))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred_Rfr))
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred_Rfr)))
```

```
R2_score: 0.5652916654636178
R2_score on training Data: 93.28366233846761
Mean Absolute Error: 0.2210392062128559
Mean Squared Error: 0.1266462718175098
Root Mean Squared Error: 0.35587395495808594
```

```
Svr = SVR()
Svr.fit(X_reg_train, y_reg_train)
pred_Svr = Svr.predict(X_reg_test)
pred_train = Svr.predict(X_reg_train)

print("R2_score:", r2_score(y_reg_test, pred_Rfr))
print("R2_score on training Data:", r2_score(y_reg_train,
pred_train)*100)
print("Mean Absolute Error:", mean_absolute_error(y_reg_test,
pred_Svr))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred_Svr))
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred_Svr)))
```

```
R2_score: 0.5652916654636178
R2_score on training Data: 93.64073581085529
Mean Absolute Error: 0.2822645105129307
Mean Squared Error: 0.2077352497250742
Root Mean Squared Error: 0.45577982593032157
```

```
Knn = KNeighborsRegressor()
Knn.fit(X_reg_train, y_reg_train)
pred_Knn = Knn.predict(X_reg_test)
pred_train = Knn.predict(X_reg_train)
```

```python
print("R2_score:", r2_score(y_reg_test, pred_Knn))
print("R2_score on training Data:", r2_score(y_reg_train,
pred_train)*100)
print("Mean Absolute Error:", mean_absolute_error(y_reg_test,
pred_Knn))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred_Knn))
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred_Knn)))
```

```
R2_score: 0.21989949581877433
R2_score on training Data: 44.8304537967118
Mean Absolute Error: 0.36130570610502627
Mean Squared Error: 0.22727151206539217
Root Mean Squared Error: 0.47673002010088705
```

Cross validation

```python
from sklearn.model_selection import cross_val_score

scores = cross_val_score(lr, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_lr) - scores.mean()
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [ 0.35465761 -0.15971131  0.14790728
0.92812555  0.62530293]
Mean Cross Validation Score: 0.3792564132201872
Difference between R2 score and Cross Validation score:
60.62335694844957
```

```python
scores = cross_val_score(Rd, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_Rd) - scores.mean()
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [-0.39245448  0.85395445 -2.70590627 -
2.81663289 -1.37626338]
Mean Cross Validation Score: -1.2874605122308764
Difference between R2 score and Cross Validation score:
205.0322078150327
```

```python
scores = cross_val_score(Dtr, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_Dtr) - scores.mean()
```

```
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [-0.1858033   0.69245978 -0.23545687
0.16030715 -4.31103007]
Mean Cross Validation Score: -0.7759046615458961
Difference between R2 score and Cross Validation score:
150.45821234031825
```

```
scores = cross_val_score(Rfr, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_Rfr) - scores.mean()
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [-0.45442632  0.38941232  0.30631176 -
0.63578677 -4.58825473]
Mean Cross Validation Score: -0.9965487479700942
Difference between R2 score and Cross Validation score:
156.18404134337118
```

```
scores = cross_val_score(Svr, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_Svr) - scores.mean()
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [-3.18257761e+00  4.35933367e-02 -
7.96937923e-01 -1.94062859e-03
 -5.80767394e+00]
Mean Cross Validation Score: -1.9491073521120341
Difference between R2 score and Cross Validation score:
223.60642981277064
```

```
scores = cross_val_score(Knn, X_reg, y_reg, cv=5)
print("Cross Validation Scores:", scores)
print("Mean Cross Validation Score:", scores.mean())
diff = r2_score(y_reg_test, pred_Knn) - scores.mean()
print("Difference between R2 score and Cross Validation score:", diff
* 100)
```

```
Cross Validation Scores: [ 0.1005464  -0.73350654 -2.90871237 -
1.20711952 -4.4900334 ]
Mean Cross Validation Score: -1.8477650861447636
Difference between R2 score and Cross Validation score:
206.7664581963538
```

HyperparameterTuning

```
#GridSearchCV
from sklearn.model_selection import GridSearchCV

lr = LinearRegression()

# Define the hyperparameters and their potential values
parameters = {
    "fit_intercept": [True, False],  # Whether to calculate the
intercept
     "copy_X": [True, False],          # Whether to copy data before
fitting
    "positive": [True, False],        # Whether to constrain the
coefficients to be positive
    "n_jobs": [None, -1]              # Number of jobs to run in
parallel (-1 means using all processors)
}


# Create a GridSearchCV object
grid = GridSearchCV(estimator=lr, param_grid=parameters, cv=5,
n_jobs=-1)

# Fit the grid search to your data
grid.fit(X_reg_train, y_reg_train)

# Get the best hyperparameters
best_params = grid.best_params_
print("Best Parameters:", best_params)

# Evaluate the model with the best parameters on the test data
score = grid.score(X_reg_test, y_reg_test)
print("Test Score:", score)
```

```
Best Parameters: {'copy_X': True, 'fit_intercept': True, 'n_jobs':
None, 'positive': True}
Test Score: 0.8913878858430498
```

```
# Create a Linear Regression model with specified hyperparameters
Model = LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
positive=True)

# Fit the model to the training data
Model.fit(X_reg_train, y_reg_train)

# Make predictions on the test data
pred = Model.predict(X_reg_test)

# Calculate regression metrics
print("R2_score:", r2_score(y_reg_test, pred))
print("Mean Absolute Error:", mean_absolute_error(y_reg_test, pred))
print("Mean Squared Error:", mean_squared_error(y_reg_test, pred))
```

```
print("Root Mean Squared Error:",
np.sqrt(mean_squared_error(y_reg_test, pred)))
```

```
R2_score: 0.8913878858430498
Mean Absolute Error: 0.1588283575519478
Mean Squared Error: 0.03164263999415976
Root Mean Squared Error: 0.17788378226853555
```

```
#Save the model for Task(U.S. National Home Price Index)
import pickle

filename = 'U.S. National Home Price Index.pkl'
pickle.dump(Model, open(filename, 'wb'))

LoadedModel = pickle.load(open('U.S. National Home Price Index.pkl',
'rb'))
result = LoadedModel.score(X_reg_test, y_reg_test)
print(result * 100)
```

```
89.13878858430499
```

```
Conclusion = pd.DataFrame([LoadedModel.predict(X_reg_test)
[:],y_reg_test[:]],index = ["predicted","Original"])
Conclusion
```

```
                  0         1         2         3         4         5  \
predicted  6.774923  5.490402  6.039415  5.626184  5.376732  5.404472   

Original   6.683055  5.366143  5.872073  5.563256  5.249654  5.183323   


                  6  
predicted  6.065704  
Original   6.382878  
```