Factors influencing US home prices

Task:

Find publicly available data for key factors that influence US home price nationally.Then build a data science model that explains how these factors impacted home price over the last 20 years.Use the S&P Case-Schiller Home Price Index as a proxy for home prices.

Approach:

First I have to find the key factors that have influence on US home prices. Then I will require to make a dataset, which contains these key factors. Then I will use machine learning models to explain the impact of these factors on home prices over 20 years.

1.Key Factors

I have used following key factors, that can influence US home price:

- Interest Rate: Average mortgage rate
- Inflation: Consumer Price Index(CPI)
- Economic growth: GDP Growth Rate
- Income levels: Median household income
- Population growth: Population data
- Unemployment Rates: National unemployment rate
- Supply of Homes: Housing starts and permits

2.Dataset

As I am not using any pre-existing datasets, I will gather data for each factor individually from public websites.I have fetched data from Federal Reserve Economic Data (FRED). FRED is an online database consisting of hundreds of thousands of economic data time series from scores of national, international, public, and private sources. FRED, created and maintained by the Research Department at the Federal Reserve Bank of St. Louis.
[(https://fredhelp.stlouisfed.org/fred/about/about-fred/what-is-fred/)]
I have fetched this data using API.

Import required libraries:

Pandas library is used for data manipulations and analysis, Requests library for making http requests or basically to receive data and Numpy is for performing operations on arrays.

```
#importing required libraries
import pandas as pd
import numpy as np
import requests
```

A function is defined to fetch data from FRED website. Function named *fetch_fred_data* is created. It has two arguments *series_id* : the id of the key factors and *api_key* : my api key

```
# Define function to fetch data from FRED
def fetch_fred_data(series_id, api_key):
    url = f'https://api.stlouisfed.org/fred/series/observations?series_id={series_id}&api_key={api_key}&file_typ
    response = requests.get(url)
    data = response.json()
    df = pd.DataFrame(data['observations'])
    df['date'] = pd.to_datetime(df['date'])
    df.set_index('date', inplace=True)
    df[series_id] = df['value'].astype(float)
    df = df[[series_id]]
    return df

#my api key
api_key = '********************************'
```

Data for all required key factors is fetched using their corresponding series id.

```
 # 30-Year Fixed Mortgage Rate
interest_rates = fetch_fred_data('MORTGAGE30US', api_key)
# Consumer Price Index for All Urban Consumers
cpi = fetch_fred_data('CPIAUCSL', api_key)
# Median Household Income
income = fetch_fred_data('MEHOINUSA646N', api_key)
# Unemployment Rate
unemployment = fetch_fred_data('UNRATE', api_key)
# Housing Starts
housing_starts = fetch_fred_data('HOUST', api_key)
# Real GDP
gdp = fetch_fred_data('A191RL1Q225SBEA', api_key)
# Population
population = fetch_fred_data('POPTHM', api_key)
```

Now I will load data of our target variable that is Case-Shiller home price index. I have already downloaded the case-shiller home price index.

```
# Load the Case-Shiller data from a local file
local_case_shiller_path = 'CSUSHPINSA.csv'
case_shiller = pd.read_csv(local_case_shiller_path)

# Process the Case-Shiller data
case_shiller['date'] = case_shiller['DATE']
case_shiller.set_index('date', inplace=True)
case_shiller.rename(columns={'CSUSHPINSA': 'Case_Shiller_Index'}, inplace=True)
case_shiller = case_shiller[['Case_Shiller_Index']]
```

While looking at the fetched data from FRED , I found that these dataset have too many columns containing data from many years.
For example, dataset of interest rate contains data from year-1971 :

```
interest_rates
```

	MORTGAGE30US
date	
1971-04-02	7.33
1971-04-09	7.31
1971-04-16	7.31
1971-04-23	7.31
1971-04-30	7.29
...	...
2024-05-09	7.09
2024-05-16	7.02
2024-05-23	6.94
2024-05-30	7.03
2024-06-06	6.99

2776 rows × 1 columns

dataset of cpi contains data from year-1947:

```
1 cpi
```

	CPIAUCSL
date	
1947-01-01	21.480
1947-02-01	21.620
1947-03-01	22.000
1947-04-01	22.000
1947-05-01	21.950
...	...
2023-12-01	308.742
2024-01-01	309.685
2024-02-01	311.054
2024-03-01	312.230
2024-04-01	313.207

928 rows × 1 columns

And similarly all data set contains data more than required.So, I have removed rows from each dataset such that they contains only required data(i.e. Data from year-2000 to....)

```
#reducing rows from all datasets
interest_rates2= interest_rates.iloc[1501:]
```

```
cpi2=cpi.iloc[636:]
```

```
income2 = income.iloc[16:]
```

```
unemployment2 = unemployment.iloc[636:]
```

```
housing_starts2= housing_starts.iloc[492:]
```

```
gdp2 = gdp.iloc[211:]
```

```
population2 = population.iloc[492:]
```

I have deleted the rows using the iloc() function .
After deleting the rows, now the dataset contains data only after the year-2000.
for example:-

```
interest_rates2
```

	MORTGAGE30US
date	
2000-01-07	8.15
2000-01-14	8.18
2000-01-21	8.26
2000-01-28	8.25
2000-02-04	8.25
...	...
2024-05-09	7.09
2024-05-16	7.02
2024-05-23	6.94
2024-05-30	7.03
2024-06-06	6.99

1275 rows × 1 columns

Similarly reduce case shiller index

```
case_shiller2 = case_shiller.iloc[156:]
case_shiller2
```

	Case_Shiller_Index
date	
2000-01-01	100.000
2000-02-01	100.571
2000-03-01	101.466
2000-04-01	102.540
2000-05-01	103.702
...	...
2023-11-01	311.969
2023-12-01	310.774
2024-01-01	310.521
2024-02-01	312.632

After reducing all these datasets, I load them as dataframe

```
# Load all dataset
df1 = pd.DataFrame(cpi2)
df2 = pd.DataFrame(income2)
df3=pd.DataFrame(interest_rates2)
df4=pd.DataFrame(unemployment2)
df5=pd.DataFrame(housing_starts2)
df6=pd.DataFrame(gdp2)
df7=pd.DataFrame(population2)
```

Then I merged all these dataframe into a single dataframe on a common column i.e. date

```
# Merge dataframes on a common column
merged_df = pd.merge(df1, df2, on='date' , how= 'outer')

merged_df = pd.merge(merged_df, df3, on='date' , how= 'outer')
merged_df = pd.merge(merged_df, df4, on='date',how= 'outer')
merged_df = pd.merge(merged_df, df5, on='date',how= 'outer')
merged_df = pd.merge(merged_df, df6, on='date',how= 'outer')
merged_df = pd.merge(merged_df, df7, on='date',how= 'outer')


# checking merged dataframe
print(merged_df.head())
```

Here I got my merged dataframe containing all 7 features:

```
            CPIAUCSL  MEHOINUSA646N  MORTGAGE30US  UNRATE   HOUST  \
date
2000-01-01     169.3        41990.0           NaN     NaN  1636.0
2000-02-01     170.0            NaN           NaN     NaN  1737.0
2000-03-01     171.0            NaN           NaN     NaN  1604.0
2000-04-01     170.9            NaN           NaN     NaN  1626.0
2000-05-01     171.2            NaN           NaN     NaN  1575.0

            A191RL1Q225SBEA    POPTHM
date
2000-01-01              1.5  281083.0
2000-02-01              NaN  281299.0
2000-03-01              NaN  281531.0
2000-04-01              7.5  281763.0
2000-05-01              NaN  281996.0
```

Then to get the final dataset for modelling , I merged my *merged_df* with case_shiller index and got my final data.

My final dataset , looks like:

	date	Case_Shiller_Index	CPIAUCSL	MEHOINUSA646N	MORTGAGE30US	UNRATE	HOUST	A191RL1Q225SBEA	POPTHM
0	2000-01-01	100.000	169.300	41990.0	NaN	NaN	1636.0	1.5	281083.0
1	2000-02-01	100.571	170.000	41990.0	NaN	NaN	1737.0	1.5	281299.0
2	2000-03-01	101.466	171.000	41990.0	NaN	NaN	1604.0	1.5	281531.0
3	2000-04-01	102.540	170.900	41990.0	NaN	NaN	1626.0	7.5	281763.0
4	2000-05-01	103.702	171.200	41990.0	NaN	NaN	1575.0	7.5	281996.0
...	...	...	...	...	...	...	...	...	...
286	2023-11-01	311.969	308.024	74580.0	6.79	3.7	1510.0	3.4	335925.0
287	2023-12-01	310.774	308.742	74580.0	6.79	3.7	1568.0	3.4	336070.0
288	2024-01-01	310.521	309.685	74580.0	6.79	3.7	1376.0	1.3	336194.0
289	2024-02-01	312.632	311.054	74580.0	6.63	3.9	1546.0	1.3	336306.0
290	2024-03-01	316.646	312.230	74580.0	6.63	3.8	1287.0	1.3	336423.0

291 rows × 9 columns

I saved this final data as *us_home_data.csv*

```
final_data.to_csv('us_home_data.csv')
```

So finally, I got my dataset named as *us_home_data.csv* .I will use this dataset for modelling.

Following are the features used to explain their impact on US home prices (I have used case-shiller index as a proxy for home prices):

- **Date**: date of recording of corresponding data
- **Case_shiller_index** : proxy for us home prices, target variable
- **CPIAUCSL** : Consumer Price index
- **MEHOINUSA646N** : Median Household Income
- **MORTGAGE30US** : Fixed Mortgage rate
- **UNRATE** : Unemployment Rate
- **HOUST** : housing starts
- **POPTHM** : Population
- **A191RL1Q225SBEA** : Real GDP

3.Data Science Model

Preparing tools

I have imported following libraries for the whole modelling work :

- Pandas for data analysis
- Matplotlib/seaborn for plotting and data visualization
- Sci-kit learn for machine learning modelling and evaluation
- Tensorflow for neural networks

```
#importing libraries
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import sklearn
import tensorflow
```

Loading and cleaning data

I have imported the final data(us_home_data.csv) as data into my workspace. Also after importing the data, I have dropped the first column containing the index.

```
#import final data
data=pd.read_csv('us_home_data.csv')
data= data.drop(data.columns[0],axis=1)
# Get the basic statistical summary
```

There was nothing much in clearing the data, I have just converted the date column into datetime format and handled the missing values using forward and backward fill

```
# Convert the 'date' column to datetime format
data['date'] = pd.to_datetime(data['date'])

# Handle missing values with forward and backward fill
df_cleaned = data.ffill().bfill()
```

My *cleaned_data* looks like:

```
                                          -------------
         date  Case_Shiller_Index  CPIAUCSL  MEHOINUSA646N  MORTGAGE30US  \
0  2000-01-01             100.000     169.3        41990.0          7.96
1  2000-02-01             100.571     170.0        41990.0          7.96
2  2000-03-01             101.466     171.0        41990.0          7.96
3  2000-04-01             102.540     170.9        41990.0          7.96
4  2000-05-01             103.702     171.2        41990.0          7.96

   UNRATE   HOUST  A191RL1Q225SBEA    POPTHM
0     4.2  1636.0              1.5  281083.0
1     4.2  1737.0              1.5  281299.0
2     4.2  1604.0              1.5  281531.0
3     4.2  1626.0              7.5  281763.0
4     4.2  1575.0              7.5  281996.0
```

Data Exploration(Exploratory Data Analysis or EDA)

Once I am done with importing and preprocessing of data, the next step is to explore the data. There isn't any set way of doing this.But what I would be trying to do is to become more familiar with the dataset, by checking its statistical summary or by comparing different columns and plotting them. The basic statistical summary of my dataset is:

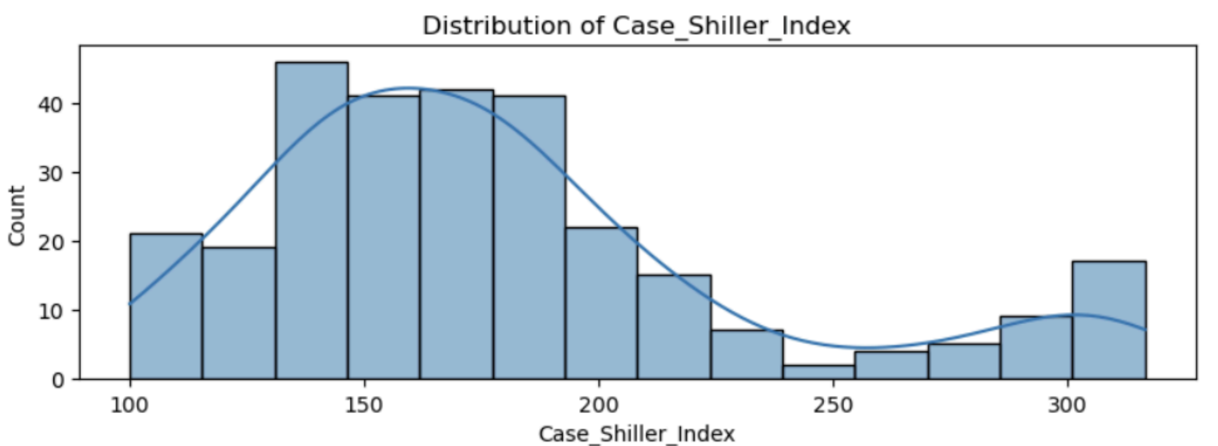
```python
stats_summary = data.describe()
print(stats_summary)
```

```
*************************************************************************************
       Case_Shiller_Index    CPIAUCSL  MEHOINUSA646N  MORTGAGE30US  \
count          291.000000  291.000000     291.000000     283.00000
mean           178.859368  227.299357   54923.505155       5.10788
std             52.944172   35.671198   10378.588906       1.33994
min            100.000000  169.300000   41990.000000       2.88000
25%            143.730500  199.350000   48200.000000       3.94000
50%            167.322000  228.329000   51020.000000       4.83000
75%            197.734500  249.553000   63180.000000       6.21000
max            316.646000  312.230000   74580.000000       7.96000

           UNRATE        HOUST  A191RL1Q225SBEA         POPTHM
count  279.000000   291.000000       291.000000     291.000000
mean     5.819713  1304.230241         2.204124  312215.470790
std      1.977937   438.512673         5.131947   16732.232301
min      3.400000   478.000000       -28.000000  281083.000000
25%      4.400000  1001.500000         1.200000  297630.000000
50%      5.400000  1305.000000         2.400000  313811.000000
75%      6.700000  1625.500000         3.500000  328152.000000
max     14.800000  2273.000000        34.800000  336423.000000
```

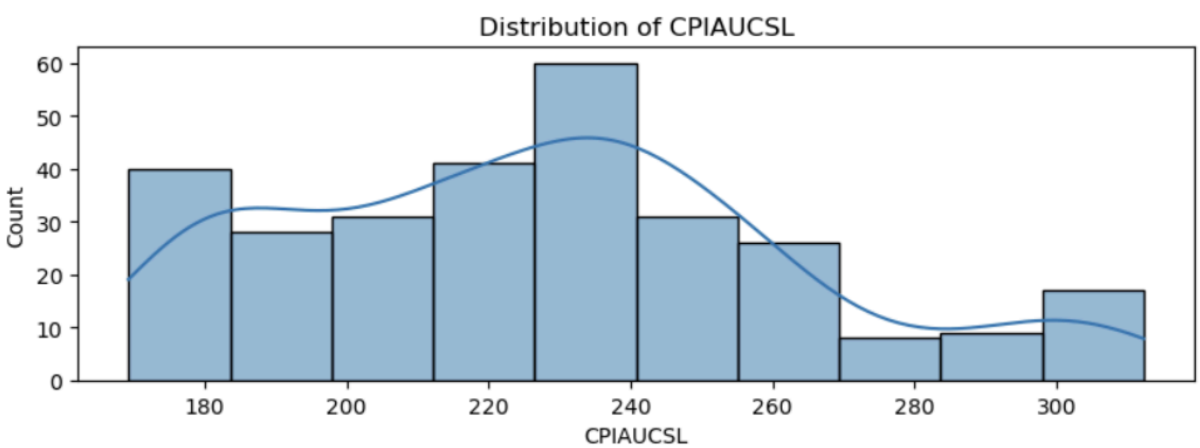
By using matplotlib, I have plotted the histogram of all features of my dataset:

```
# Set up the matplotlib figure for histograms
plt.figure(figsize=(16, 12))

# List of columns to plot
columns = ['Case_Shiller_Index', 'CPIAUCSL', 'MEHOINUSA646N', 'MORTGAGE30US',
           'UNRATE', 'HOUST', 'A191RL1Q225SBEA', 'POPTHM']

# Plot histograms
for i, col in enumerate(columns):
    plt.subplot(4, 2, i + 1)
    sns.histplot(df_cleaned[col], kde=True)
    plt.title(f'Distribution of {col}')

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

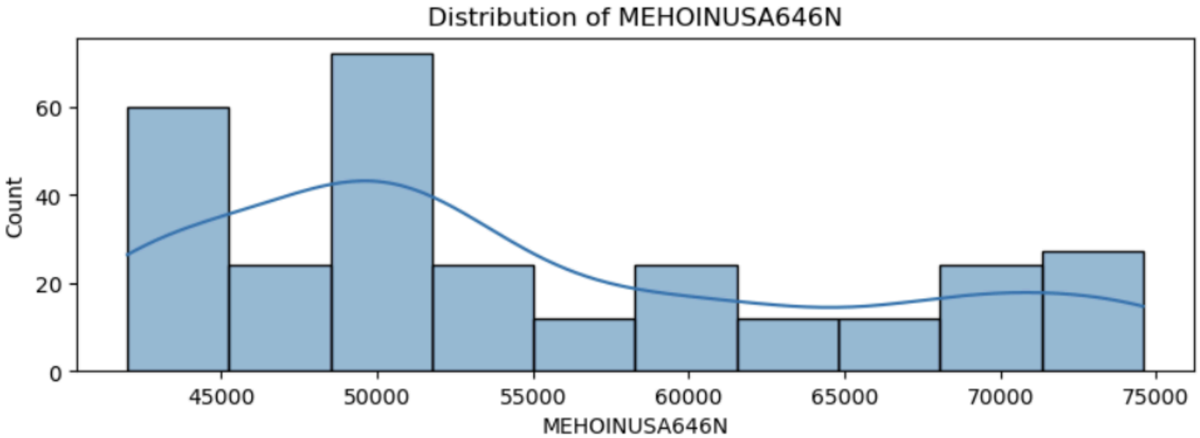
Plot obtained for Case-shiller Index:



Plot obtained for consumer price Index:

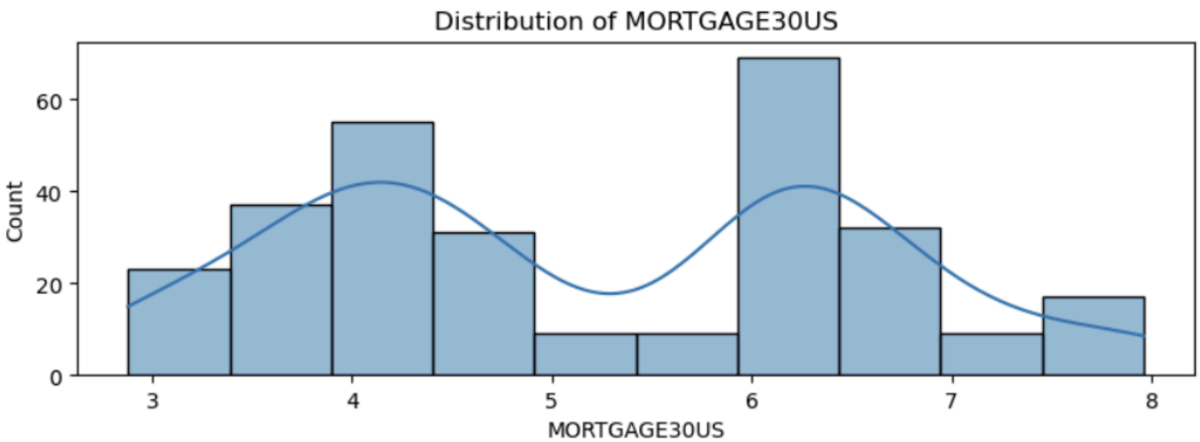


Plot obtained for median household income:

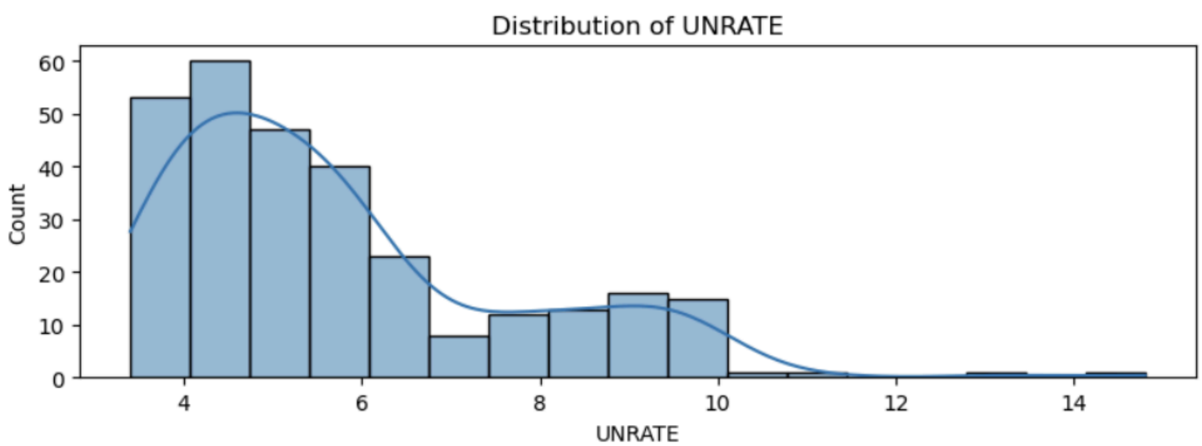


Distribution of UNRATE

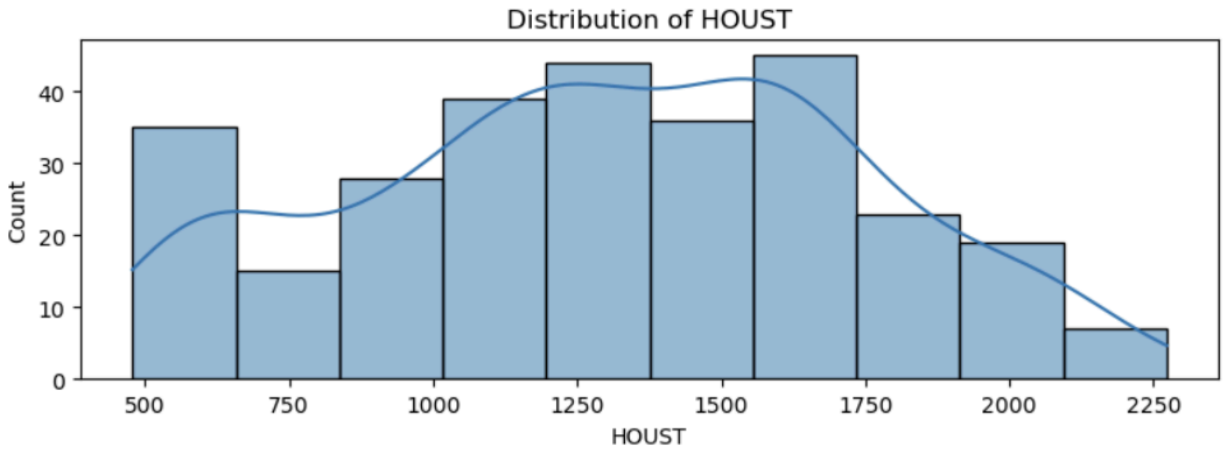
Plot obtained for interest rate:



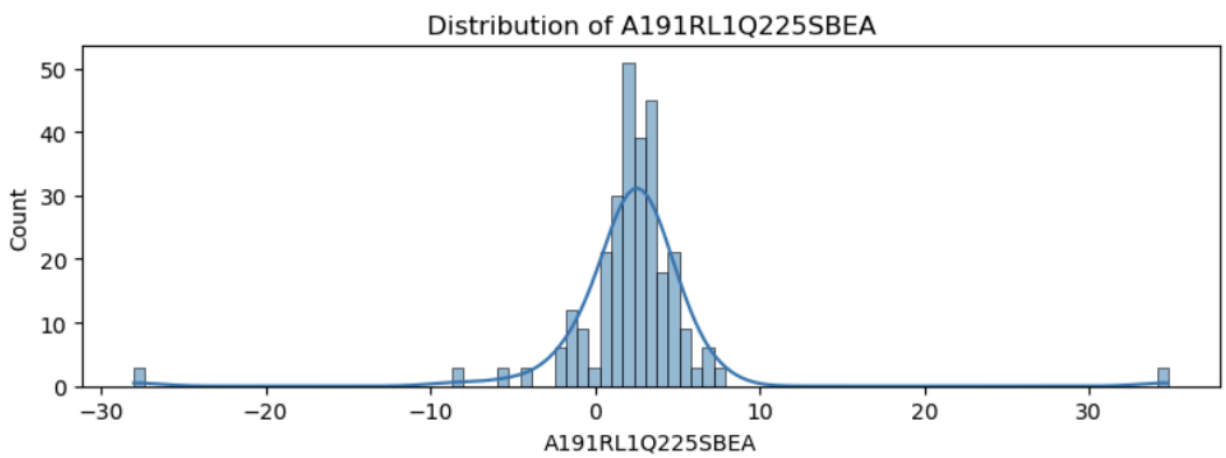
Plot obtained for unemployment rate:



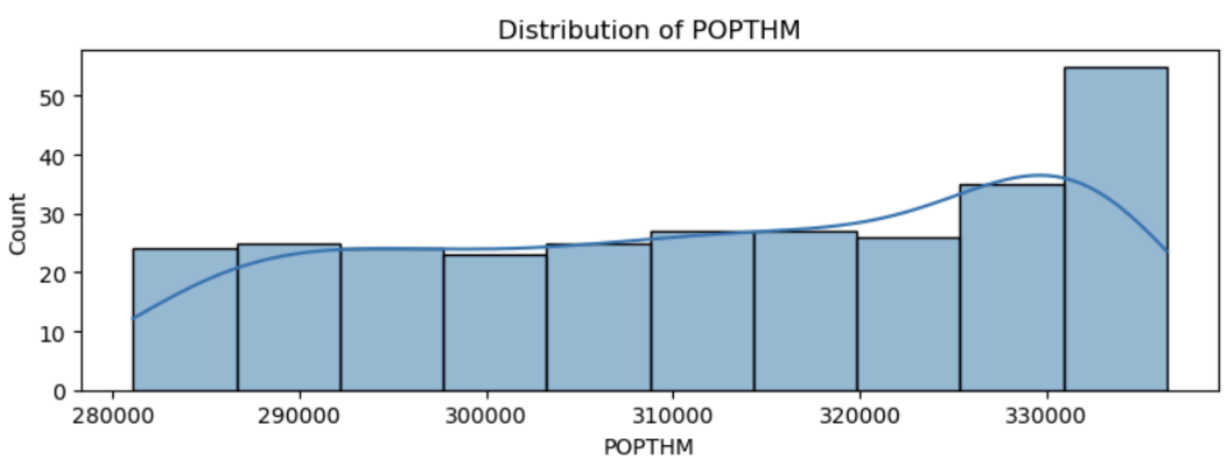
Plot obtained for Housing starts:



Plot obtained for real GDP:

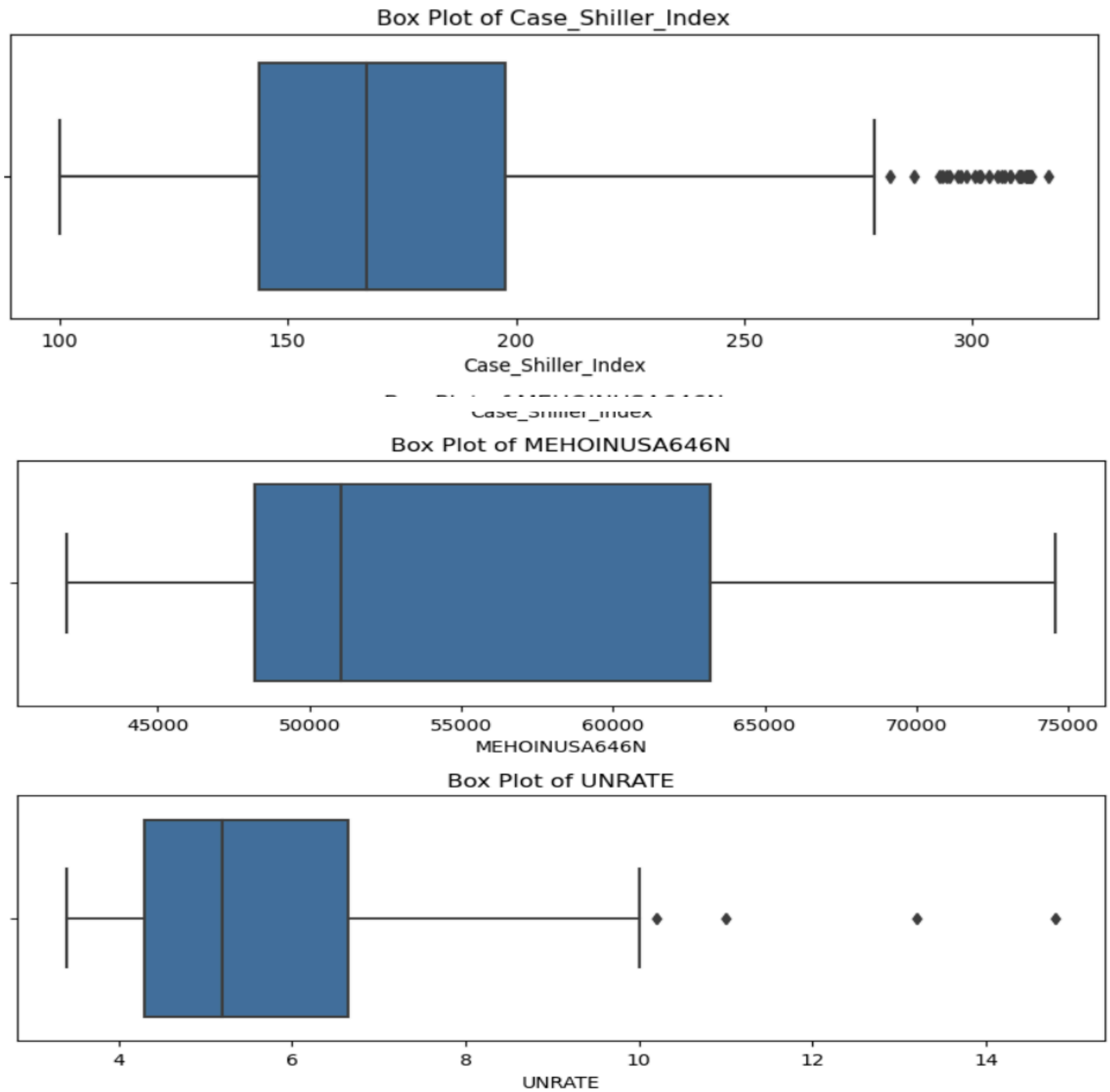


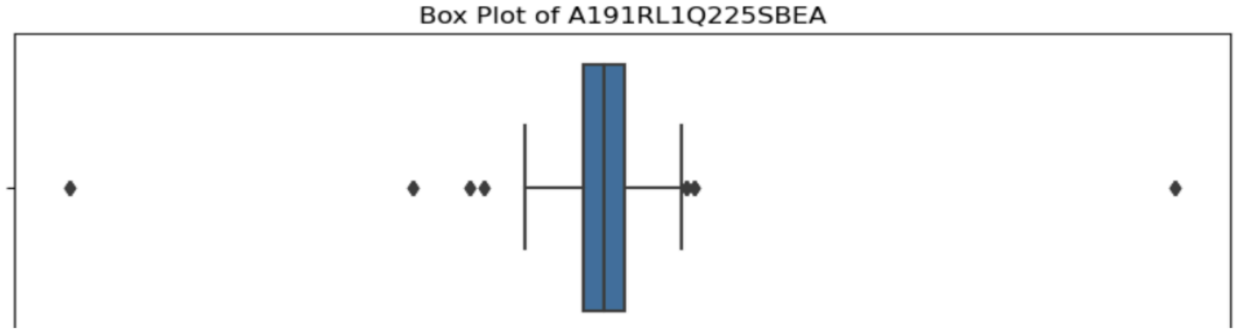
Plot obtained for population :

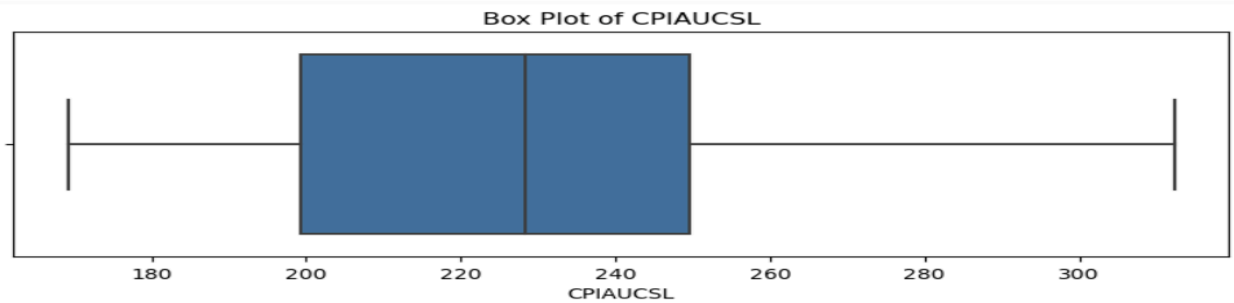


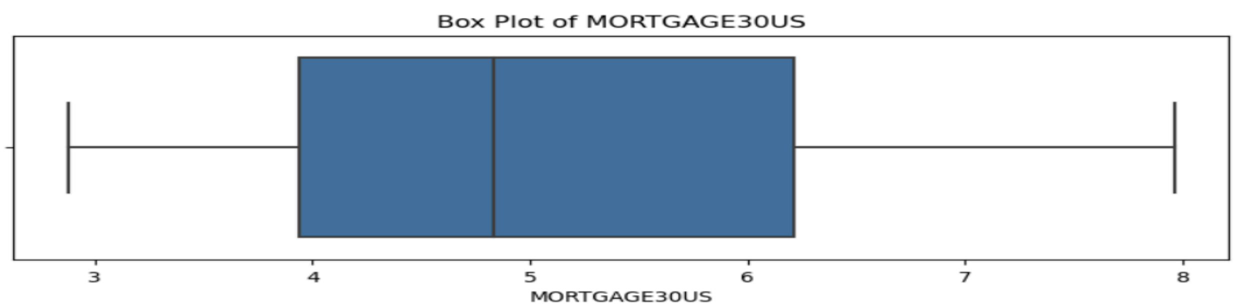
I have also plotted box plot for each features:

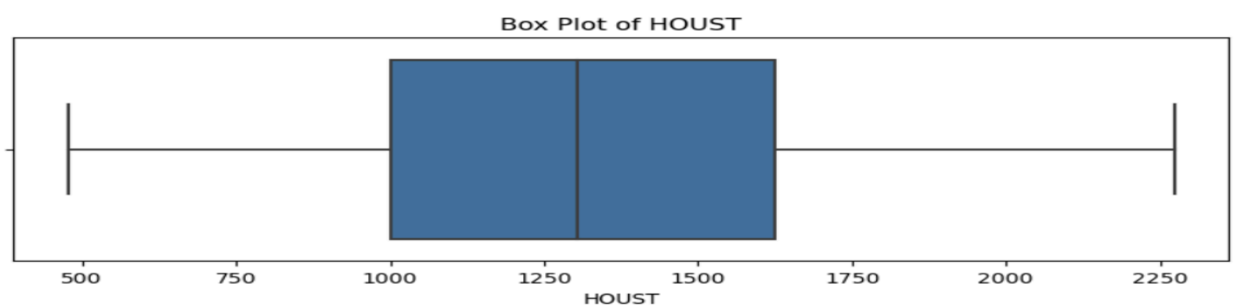
```python
# Set up the matplotlib figure for box plots
plt.figure(figsize=(16, 12))

# Plot box plots
for i, col in enumerate(columns):
    plt.subplot(4, 2, i + 1)
    sns.boxplot(x=df_cleaned[col])
    plt.title(f'Box Plot of {col}')

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

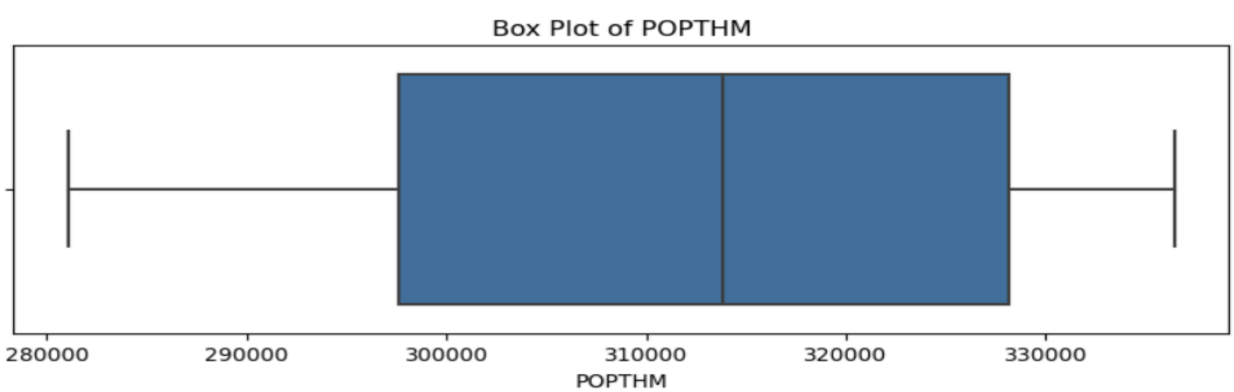



Box Plot of A191RL1Q225SBEA


Box Plot of CPIAUCSL
180
200
220
240
260
280
300
CPIAUCSL


Box Plot of MORTGAGE30US
3
4
5
6
7
8
MORTGAGE30US

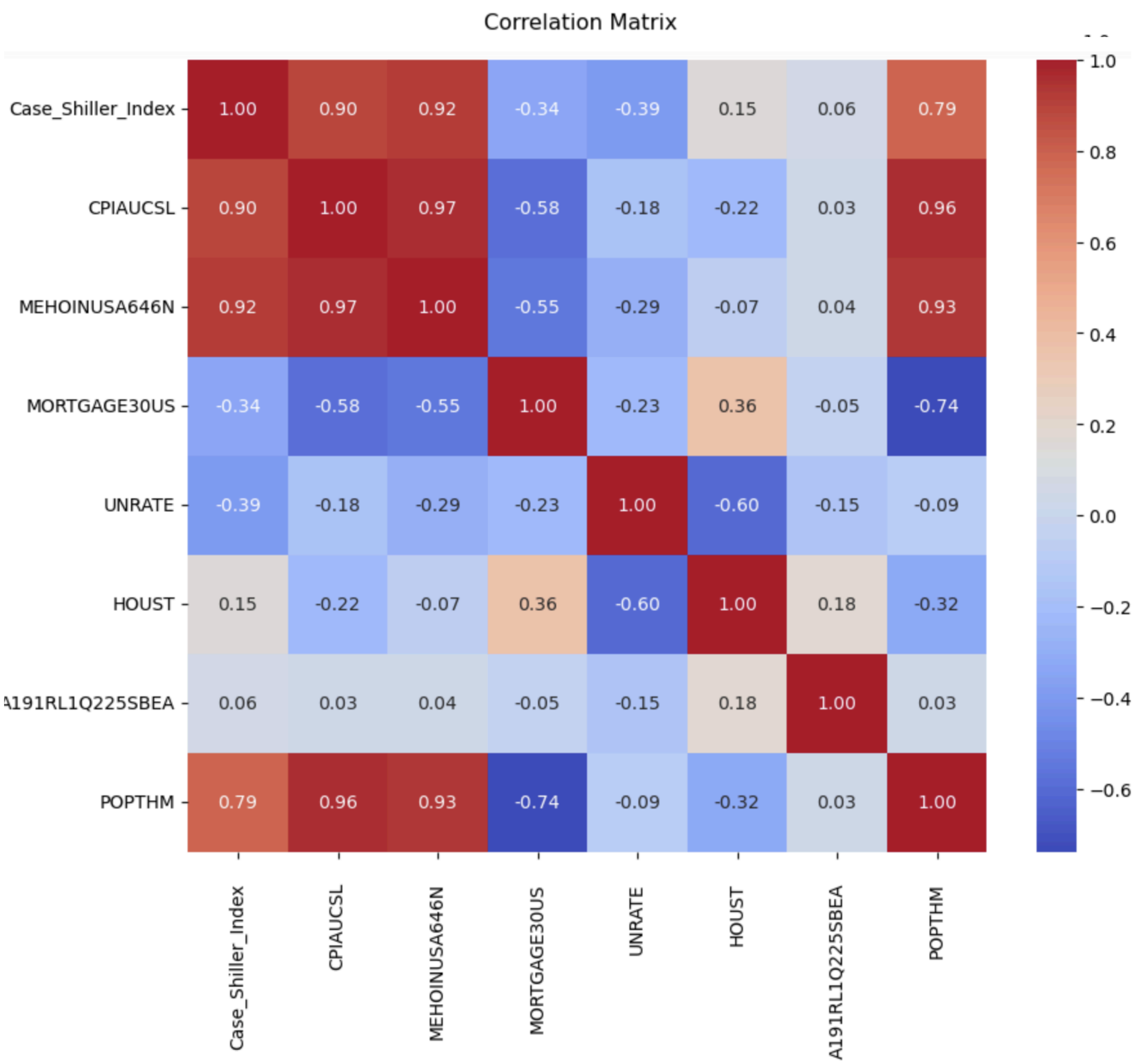

Box Plot of HOUST
500
750
1000
1250
1500
1750
2000
2250
HOUST


Box Plot of POPTHM
280000
290000
300000
310000
320000
330000
POPTHM

Finally, I have compared all the variables in one hit by creating a correlation matrix.
Why?
Because this may give an idea about which independent variables may or may not have an impact on our target variable.

```
# Correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(df_cleaned.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()
```

The correlation matrix, I have obtained:-



Note-A higher positive value means a potential positive correlation(increase) and a higher negative value means a potential negative correlation(decrease).

Machine learning model:

Before building the model, I have to make my dataset ready.Let's again have a look at the dataset.

date	Case_Shiller_Index	CPIAUCSL	MEHOINUSA646N	MORTGAGE30US	UNRATE	HOUST	A191RL1Q225SBEA	POPTHM
2000-01-01	100.000	169.3	41990.0	7.96	4.2	1636.0	1.5	281083.0
2000-02-01	100.571	170.0	41990.0	7.96	4.2	1737.0	1.5	281299.0
2000-03-01	101.466	171.0	41990.0	7.96	4.2	1604.0	1.5	281531.0
2000-04-01	102.540	170.9	41990.0	7.96	4.2	1626.0	7.5	281763.0
2000-05-01	103.702	171.2	41990.0	7.96	4.2	1575.0	7.5	281996.0

I am trying to look at the impact of other variables on our target variable.To do this I split the target variable from the rest.

```
# Define features (X) and target (y)
X = df_cleaned.drop(columns=['date', 'Case_Shiller_Index'])
y = df_cleaned['Case_Shiller_Index']
```

Now I will split my dataset into training and testing data.I have also standardize the features

```
#importing library
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

```
# 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)

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Now, I've got my dataset ready for fitting into models.I will be using the following models and comparing the results.

- Linear Regression
- Random Forest
- Gradient boosting

Importing all required models:

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

Defining the models:

```
# Define the models
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100, random_state=42)
}
```

Training the model and evaluating them by calculating their evaluation metrics:

```
# Train the models and evaluate them
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate evaluation metrics
    rmse = mean_squared_error(y_test, y_pred, squared=False)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f'{name} Performance:')
    print(f'RMSE: {rmse:.2f}')
    print(f'MAE: {mae:.2f}')
    print(f'R²: {r2:.2f}')
    print('-' * 30)
```

Performance of different models:-

```
Linear Regression Performance:
RMSE: 1.52
MAE: 1.12
R²: 1.00
------------------------------
Random Forest Performance:
RMSE: 1.68
MAE: 1.25
R²: 1.00
------------------------------
Gradient Boosting Performance:
RMSE: 1.63
MAE: 1.10
R²: 1.00
------------------------------
```

Evaluation metrics

To choose which model is best fit for my dataset, I have to understand their evaluation metrics. Various parameters of evaluation metrics are:

- MAE
  - ➔ Stands for Mean Absolute Error
  - ➔ Lower values indicates a better fit
- RMSE
  - ➔ Stands for Root Mean Squared Error
  - ➔ Lower values indicate a better fit
  - ➔ RMSE penalizes larger error more due to the squaring of errors
- R^2(R-squared)
  - ➔ Values closer to 1 indicate a better fit

Best Model

By comparing evaluation metrics for all models.

Model	MAE	RMSE	R^2
Linear Regression	0.19	0.59	1
Random Forest	0.23	0.86	1
Gradient Boosting	0.95	0.95	1

I have decided that best fit model for my dataset is linear regression (As it has lowest MAE and RMSE value)

Feature importance

Impact of each feature can explain by their respective coefficients. Linear regression model try to fit a line/curve on the given data to predict the underlying function. These coefficients are associated with the line/curve.

These coefficients can explain the impact of feature on US home prices.For example, if coefficient say “m” of a features is : m=2.1 then it means that for each unit increase in feature , home prices will increase by approximately 2 .1 units, assuming all other features remain constant.

To get the coefficients of features.

```
# For Linear Regression model
lr_model = models['Linear Regression']
coefficients = lr_model.coef_
feature_importance = pd.DataFrame({'Feature': X.columns, 'Coefficient': coefficients})
feature_importance.sort_values(by='Coefficient', ascending=False)
```

Features and their respective coefficients are:

Feature	Coefficient
Case_Shiller_Index_lag1	48.211571
CPIAUCSL	17.294772
MEHOINUSA646N_lag1	2.224163
POPTHM_lag1	1.698071
UNRATE	1.381184
HOUST_lag1	1.268970
MORTGAGE30US_lag1	0.893314
HOUST	0.377887
A191RL1Q225SBEA_lag1	0.064246
A191RL1Q225SBEA	-0.021239
UNRATE_lag1	-0.840127
MEHOINUSA646N	-0.986930
MORTGAGE30US	-1.145231
POPTHM	-2.305245
CPIAUCSL_lag1	-15.513502

As this table contains some extra features, Features/ key factors for my importance are:-

→CPIAUCSL

→ UNRATE

→ HOUST

→ A191RL1Q225SBEA

→MEHOINUSA646N

→ MORTGAGE30US

→POPTHM

<u>Conclusion</u>

After doing all these data gathering, pre-processing, modeling and evaluation I came following conclusion :

- The factor which have most influence or had greatest impact on US home prices in last 20 year is: Consumer Price Index (CPIAUCSL), as it has highest coefficient of 17.294772

- The factor which have lowest impact is Population (POPTHM)

- All the factors arranged in decreasing order of influence upon US home prices:
  - Consumer Price Index (Highest)
  - Unemployment rate
  - Housing starts
  - Real GDP
  - Median Household Income
  - Interest rates
  - Population (Lowest)