PREDITING HOUSE PRICES USING MISHINE LEARNING

BY M.KANISH PRETHIVE

PHASE V:PROJECT SUBMISSION

INTRODUCTION

House price prediction is a challenging task, as house prices are influenced by a variety of factors, including the location, size, and condition of the house, as well as the overall market conditions. Machine learning can be used to develop models that can predict house prices with a high degree of accuracy.

Machine learning models are trained on historical data of house prices and other factors that influence house prices. Once trained, the model can be used to Benefits of using machine learning for house price prediction

Benefits to using machine learning for house price prediction

- Accuracy: Machine learning models can be trained to predict house prices with a high degree of accuracy.
- Scalability: Machine learning models can be scaled to handle large datasets of house prices and other factors.
- Objectivity: Machine learning models are objective and unbiased, unlike human appraisers who may be influenced by their own biases and experiences.

predict the price of a new house based on its features.

Challenges of using machine learning for house price prediction

Despite the benefits, there are also some challenges associated with using machine learning for house price prediction:

- Data quality: The quality of the training data is critical for the performance of the machine learning model. If the training data is noisy or incomplete, the model will not be able to learn the underlying relationships between the features of a house and its price.
- Model interpretation: It can be difficult to interpret the results of machine learning models. This can make it difficult to understand why the model made a particular prediction and to identify potential biases in the model.
- Overfitting: Overfitting is a problem that can occur when the machine learning model learns the training data too well and is unable to generalize to new data. This can lead to inaccurate predictions on new houses.

GIVEN DATA SET

Datasetlink:(https://www.kaggle.com/datasets/vedavyasv/usahousing/data))

Avg. Area	Avg. Area	Avg. Area	Avg. Area	Area Popu	Price	Address
79545.46	5.682861	7.009188	4.09	23086.8	1059034	208
79248.64	6.0029	6.730821	3.09	40173.07	1505891	188
61287.07	5.86589	8.512727	5.13	36882.16	1058988	9127
63345.24	7.188236	5.586729	3.26	34310.24	1260617	USS
59982.2	5.040555	7.839388	4.23	26354.11	630943.5	USNS
80175.75	4.988408	6.104512	4.04	26748.43	1068138	06039
64698.46	6.025336	8.14776	3.41	60828.25	1502056	4759
78394.34	6.98978	6.620478	2.42	36516.36	1573937	972 Joyce
59927.66	5.362126	6.393121	2.3	29387.4	798869.5	USS
81885.93	4.423672	8.167688	6.1	40149.97	1545155	Unit 9446
80527.47	8.093513	5.042747	4.1	47224.36	1707046	6368
50593.7	4.496513	7.467627	4.49	34343.99	663732.4	911
39033.81	7.671755	7.250029	3.1	39220.36	1042814	209
73163.66	6.919535	5.993188	2.27	32326.12	1291332	829
69391.38	5.344776	8.406418	4.37	35521.29	1402818	PSC 5330,
73091.87	5.443156	8.517513	4.01	23929.52	1306675	2278
79706.96	5.06789	8.219771	3.12	39717.81	1556787	064
61929.08	4.78855	5.09701	4.3	24595.9	528485.2	5498
63508.19	5.947165	7.187774	5.12	35719.65	1019426	Unit 7424
62085.28	5.739411	7.091808	5.49	44922.11	1030591	19696
86295	6.627457	8.011898	4.07	47560.78	2146925	030 Larry
60835.09	5.551222	6.517175	2.1	45574.74	929247.6	USNS
64490.65	4.210323	5.478088	4.31	40358.96	718887.2	95198
60697.35	6.170484	7.150537	6.34	28140.97	743999.8	9003 Jay
59748.86	5.33934	7.748682	4.23	27809.99	895737.1	24282
56974.48	8.287562	7.31288	4.33	40694.87	1453975	61938
82173.63	4.018525	6.992699	2.03	38853.92	1125693	3599

1.Model Evaluation and Selection

Model Evaluation and Selection:

- Split the dataset into training and testing sets
- Evaluate models using appropriate metrics (eg, Mean Absolute Error, Mean Squared
- Enor, R-squared) to assess their performance.
- Use cross-validation techniques to tune hyperparameters and ensure model stability.
- Compare the results with traditional linear regression models to highlight
- improvements.
- Select the best-performing model for further analysis.

Model Interpretability:

- Explain how to interpret feature importance from Gradient Boosting and XGBoost models.
- Discuss the insights gained from feature importance analysis and their relevance to house price prediction
- Interpret feature importance from ensemble models like Random Forest and Gradient
- Boosting to understand the factors influencing house prices.

Deployment and Prediction:

- Deploy the chosen regression model to predict house prices.

Develop a user-friendly interface for users to input property features and receive price predictions

GIVEN DATA:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns

Program:

Importing Dependencies

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg

%matplotlib inline


import warnings

warnings.filterwarnings("ignore")
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

Loading Dataset

```
dataset = pd.read_csv('/kaggle/input/usa-housing/USA_Housing.csv')
```

Model 1 - Linear Regression

In [22]:

```
model_lr=LinearRegression()
```

In [23]:

```
model_lr.fit(X_train_scal, Y_train)
```

Out[23]:

```
LinearRegression

LinearRegression()
```

Predicting Prices

In [24]:

```
Prediction1 = model_lr.predict(X_test_scal)
```

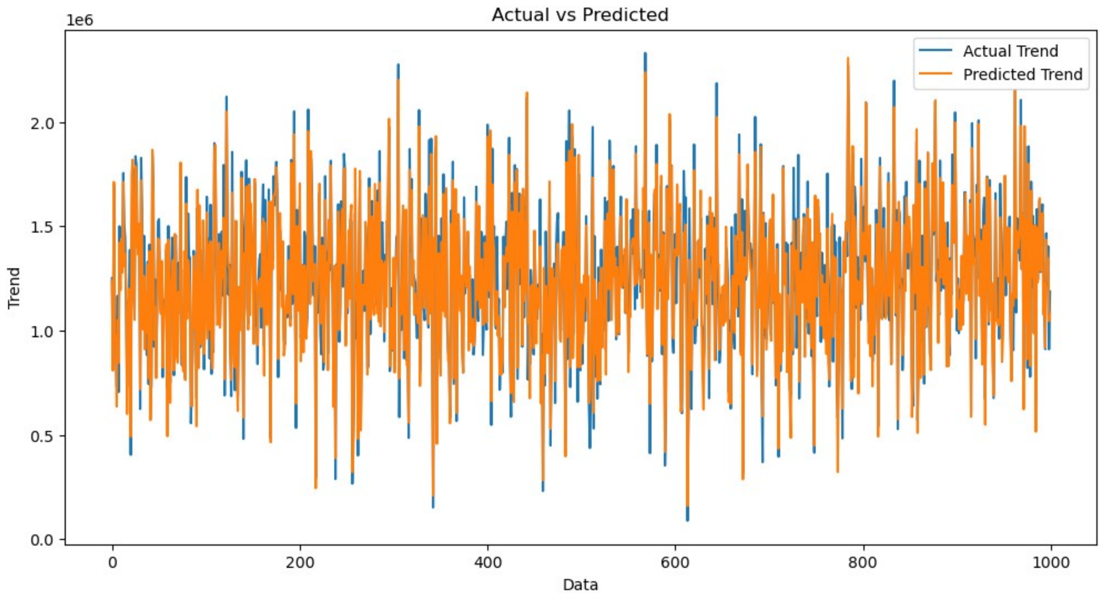
Evaluation of Predicted Data

In [25]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[25]:

```
Text(0.5, 1.0, 'Actual vs Predicted')
```

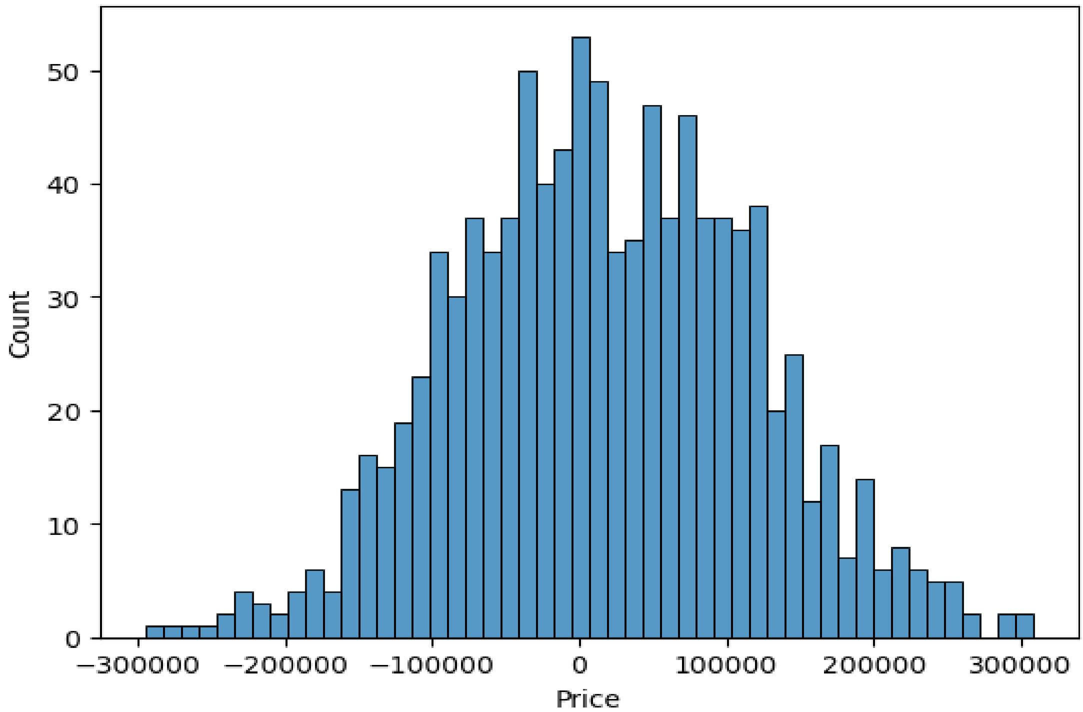


In [26]:

```
sns.histplot((Y_test-Prediction1), bins=50)
```

Out[26]:

```
<Axes: xlabel='Price', ylabel='Count'>
```



In [27]:

```
print(r2_score(Y_test, Prediction1))
 print(mean_absolute_error(Y_test, Prediction1))
```

```
print(mean_squared_error(Y_test, Prediction1))
```

```
0.9182928179392918
82295.49779231755
10469084772.975954
```

Model 2 - Support Vector Regressor

In [28]:

```
model_svr = SVR()
```

In [29]:

```
model_svr.fit(X_train_scal, Y_train)
```

Out[29]:

SVR

```
SVR()
```

Predicting Prices

In [30]:

```
Prediction2 = model_svr.predict(X_test_scal)
```

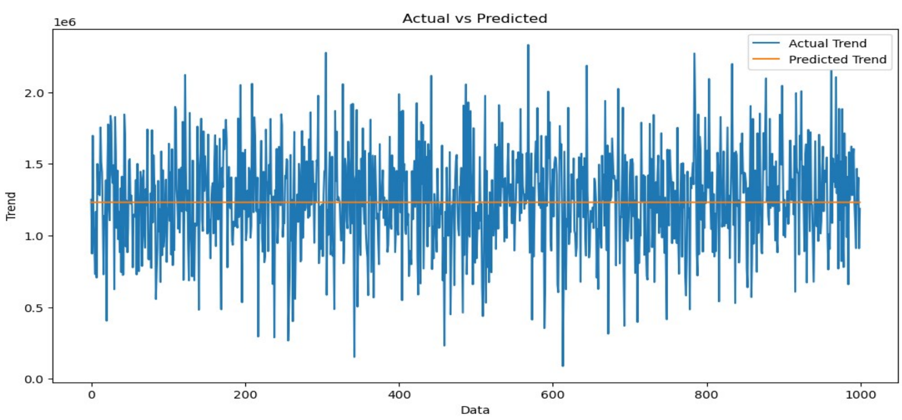
Evaluation of Predicted Data

In [31]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[31]:

```
Text(0.5, 1.0, 'Actual vs Predicted')
```

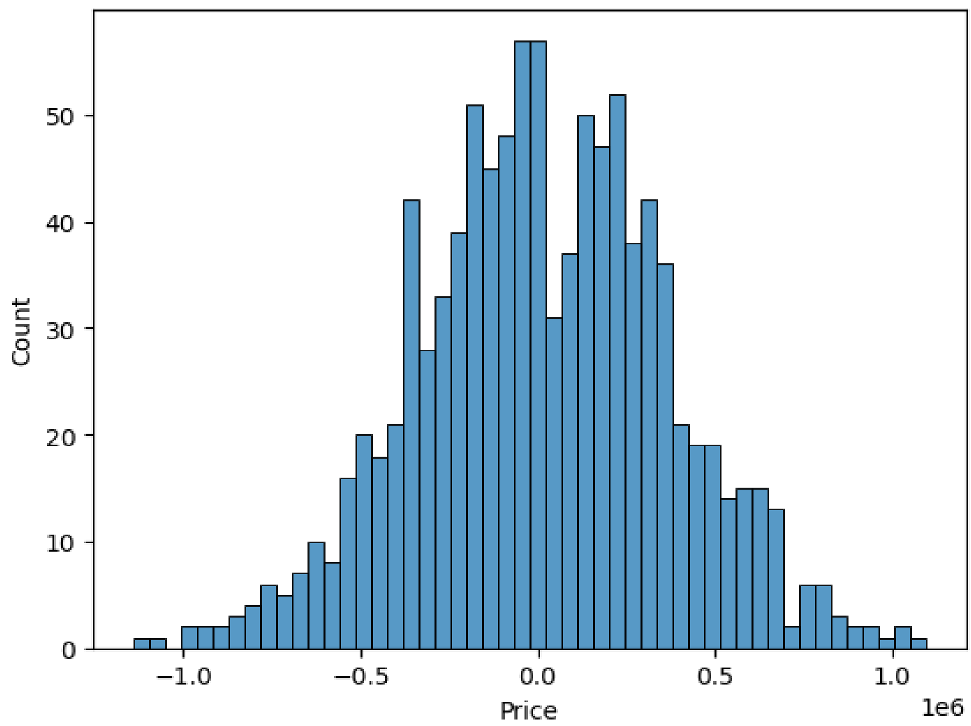


In [32]:

```
sns.histplot((Y_test-Prediction2), bins=50)
```

Out[32]:

```
<Axes: xlabel='Price', ylabel='Count'>
```



In [33]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 3 - Lasso Regression

In [34]:

```
model_lar = Lasso(alpha=1)
```

In [35]:

```
model_lar.fit(X_train_scal,Y_train)
```

Out[35]:

Lasso

```
Lasso(alpha=1)
```

Predicting Prices

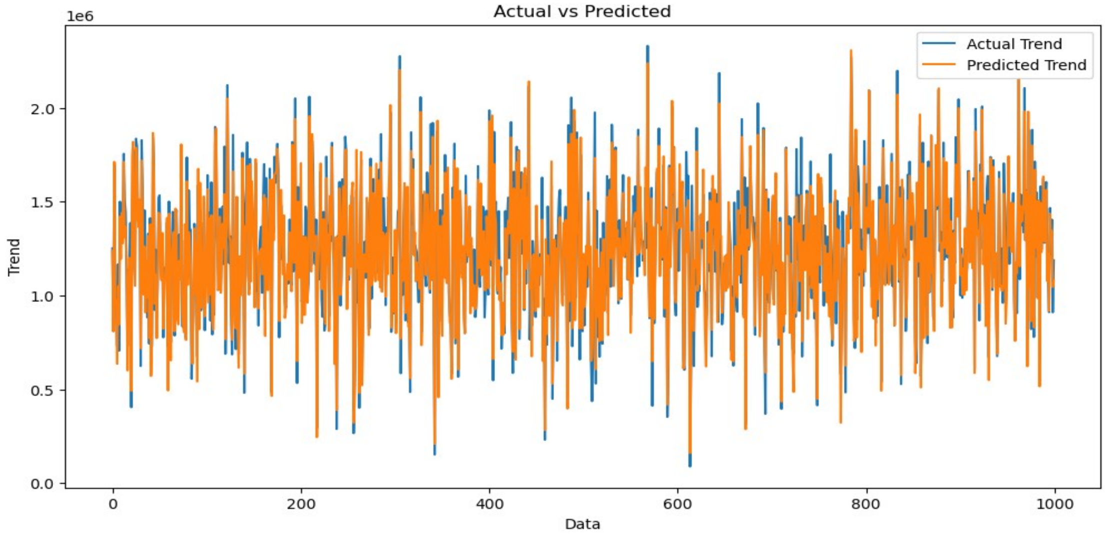
In [36]:

```
Prediction3 = model_lar.predict(X_test_scal)
```

Evaluation of Predicted Data

In [37]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction3, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```



Out[37]:

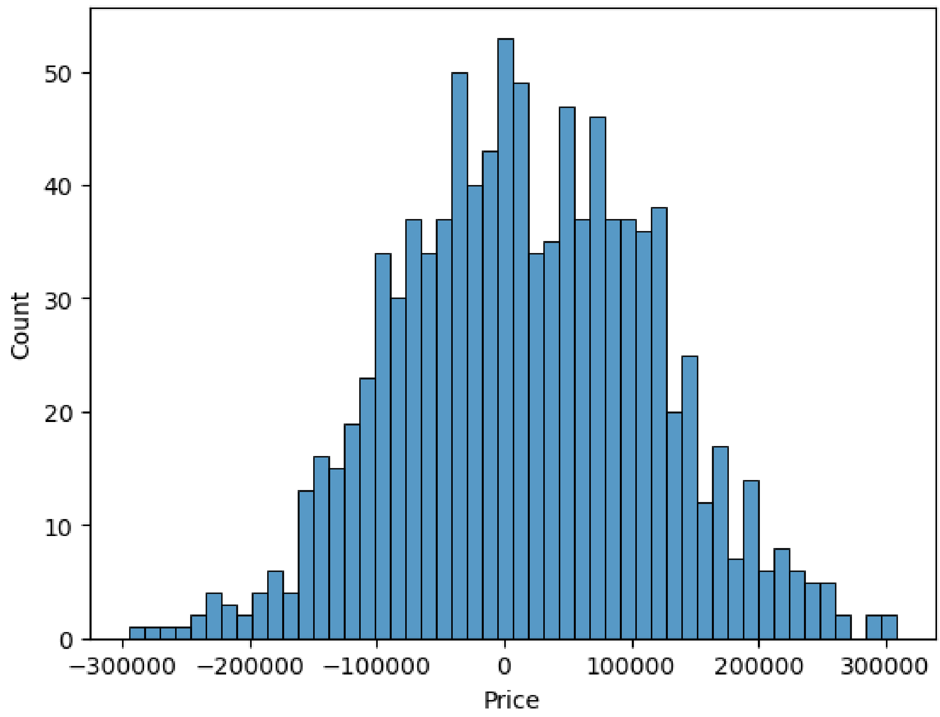
```
Text(0.5, 1.0, 'Actual vs Predicted')
```

In [38]:

```
sns.histplot((Y_test-Prediction3), bins=50)
```

Out[38]:

```
<Axes: xlabel='Price', ylabel='Count'>
```



In [39]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 4 - Random Forest Regressor

In [40]:

```
model_rf = RandomForestRegressor(n_estimators=50)
```

In [41]:

```
model_rf.fit(X_train_scal, Y_train)
```

Out[41]:

```
RandomForestRegressor

RandomForestRegressor(n_estimators=50)
```

Predicting Prices

In [42]:

```
Prediction4 = model_rf.predict(X_test_scal)
```

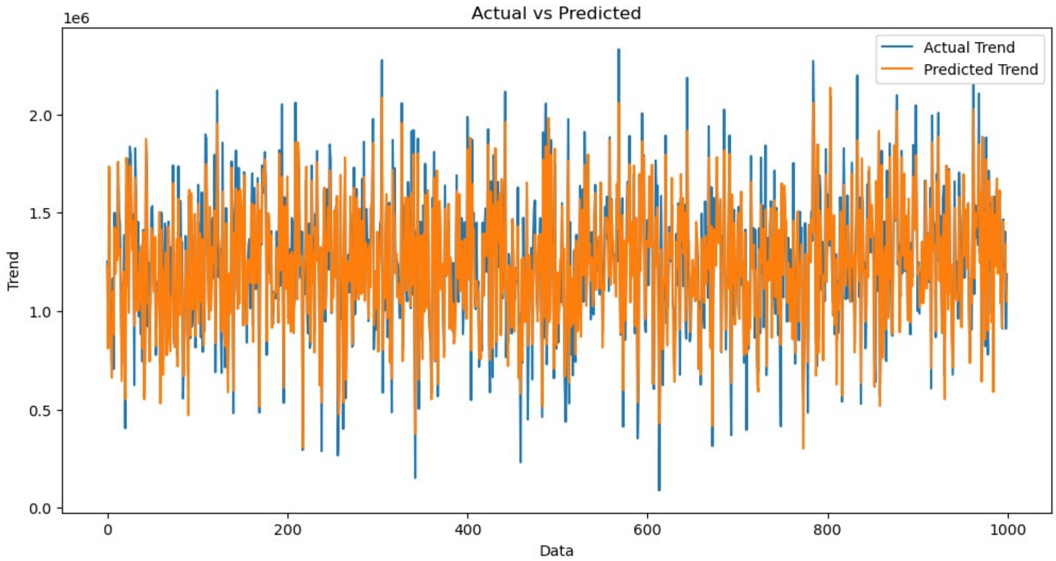
Evaluation of Predicted Data

In [43]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[43]:

```
Text(0.5, 1.0, 'Actual vs Predicted')
```

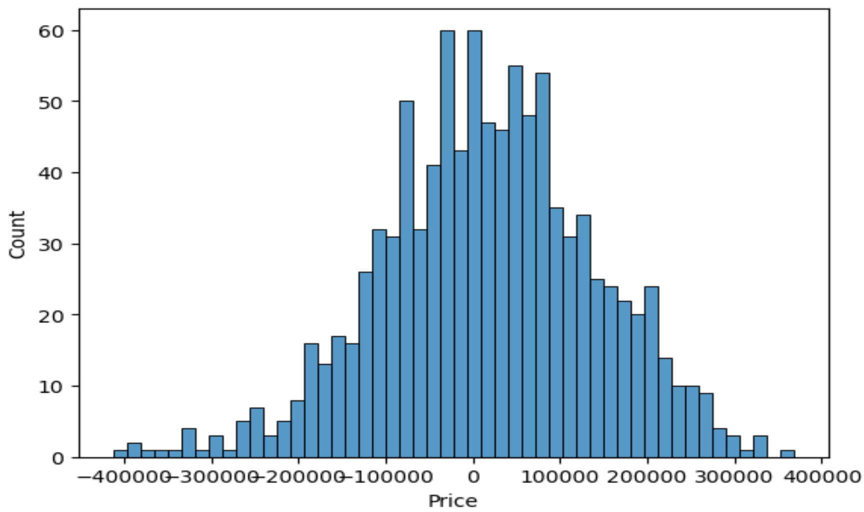


In [44]:

```
sns.histplot((Y_test-Prediction4), bins=50)
```

Out[44]:

```
<Axes: xlabel='Price', ylabel='Count'>
```



In [45]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 5 - XGboost Regressor

In [46]:

```
model_xg = xg.XGBRegressor()
```

In [47]:

```
model_xg.fit(X_train_scal, Y_train)
```

Out[47]:

```
XGBRegressor
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None,
importance_type=None,
             interaction_constraints=None, learning_rate=None,
max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan,
monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

Predicting Prices

In [48]:

```
Prediction5 = model_xg.predict(X_test_scal)
```

Evaluation of Predicted Data

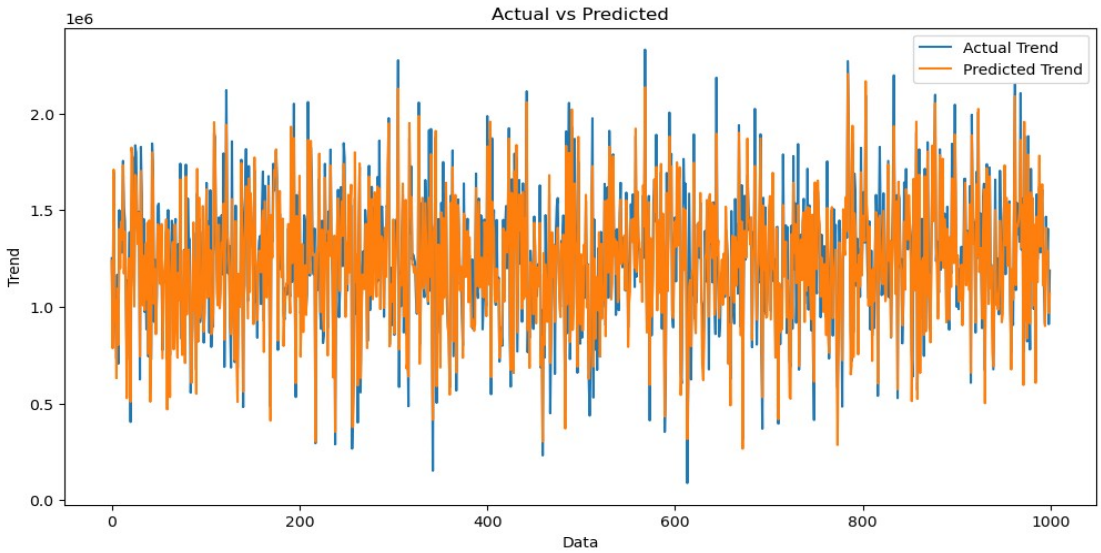
In [49]:

```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
```

```
plt.plot(np.arange(len(Y_test)), Prediction5, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[49]:

```
Text(0.5, 1.0, 'Actual vs Predicted')
```

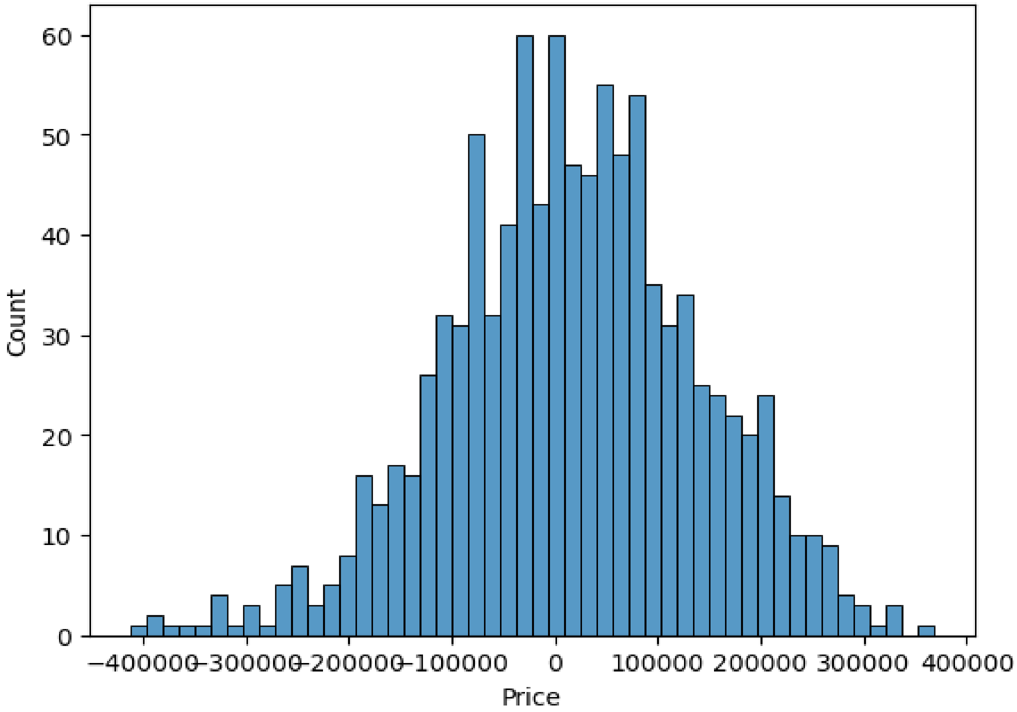


In [50]:

```
sns.histplot((Y_test-Prediction4), bins=50)
```

Out[50]:

```
<Axes: xlabel='Price', ylabel='Count'>
```



In [51]:

```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

2.Loading and processing Data

Loading a dataset:

The first step is to load the dataset into memory. This can be done using a variety of programming languages and libraries

Preprocessing a dataset

Once the dataset is loaded into memory, it is important to preprocess the data before using it for machine learning. Preprocessing involves cleaning and transforming the data so that it is ready for machine learning algorithm

Program:

Importing dependences:

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

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg
```

Loading Dataset:

```
dataset = pd.read_csv('/kaggle/input/usa-housing/USA_Housing.csv')
```

Data Exploration:

```
dataset
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns

```
dataset.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
```

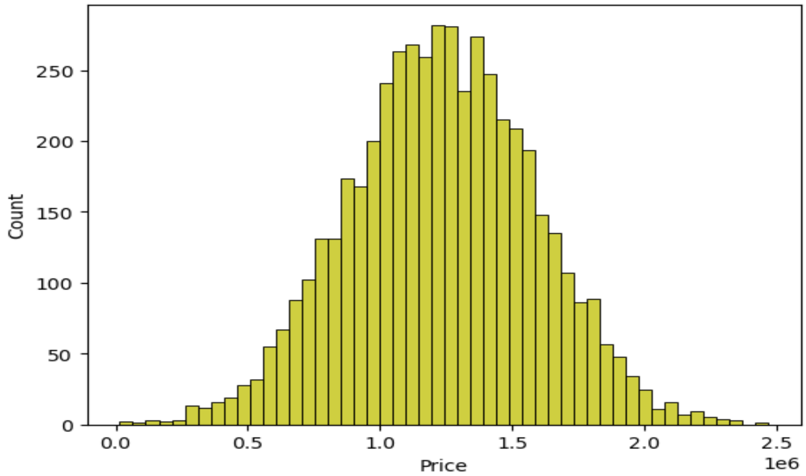
```
memory usage: 273.6+ KB
dataset.describe()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

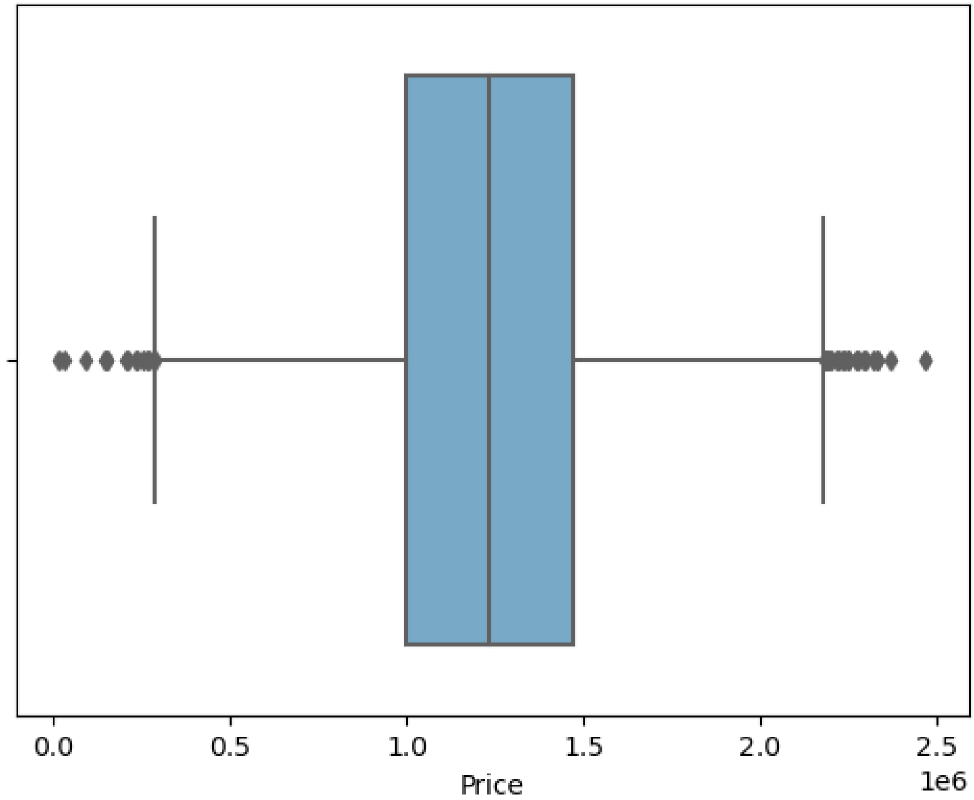
```
dataset.columns
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number
of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price',
'Address'],
      dtype='object')
```

Visualisation and Pre-Processing of Data

```
sns.histplot(dataset, x='Price', bins=50, color='y')
<Axes: xlabel='Price', ylabel='Count'>
```

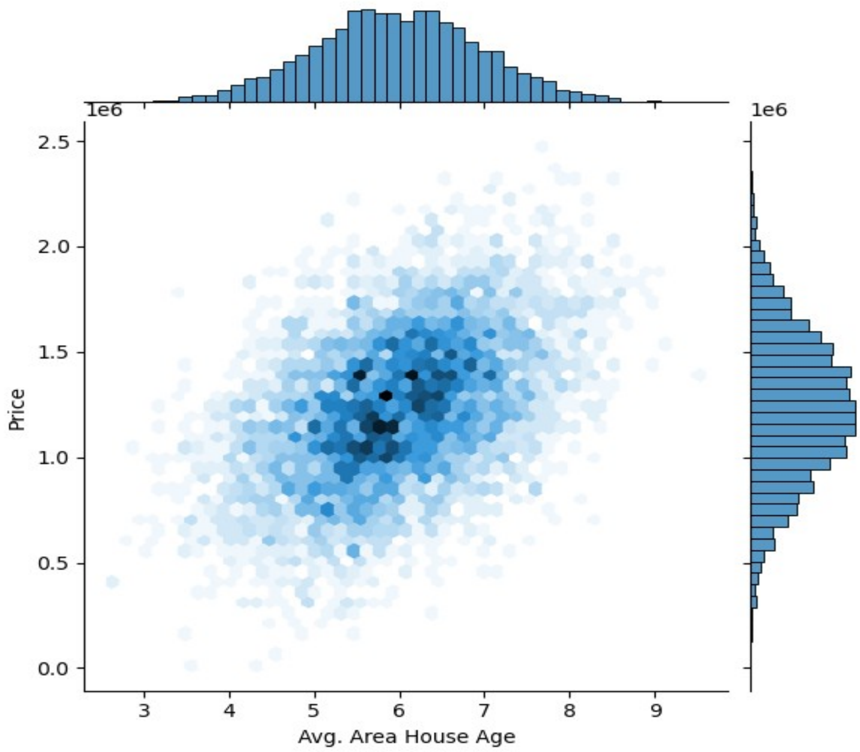



```
sns.boxplot(dataset, x='Price',  palette='Blues')
<Axes: xlabel='Price'>
```



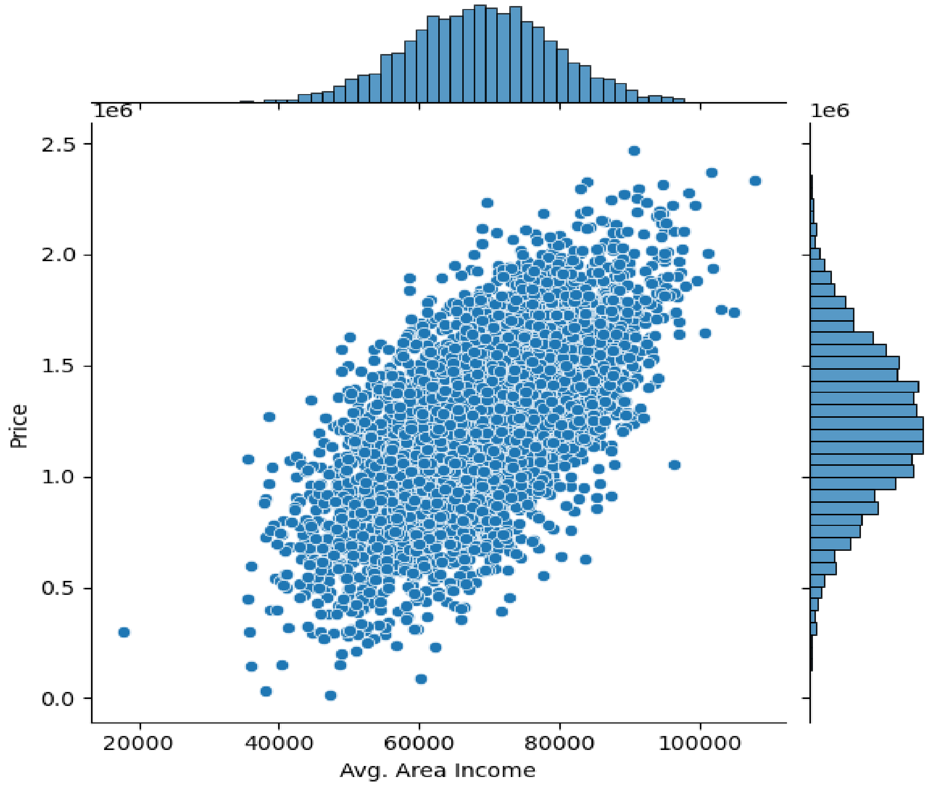

```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price',
kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7caf1d571810>
```



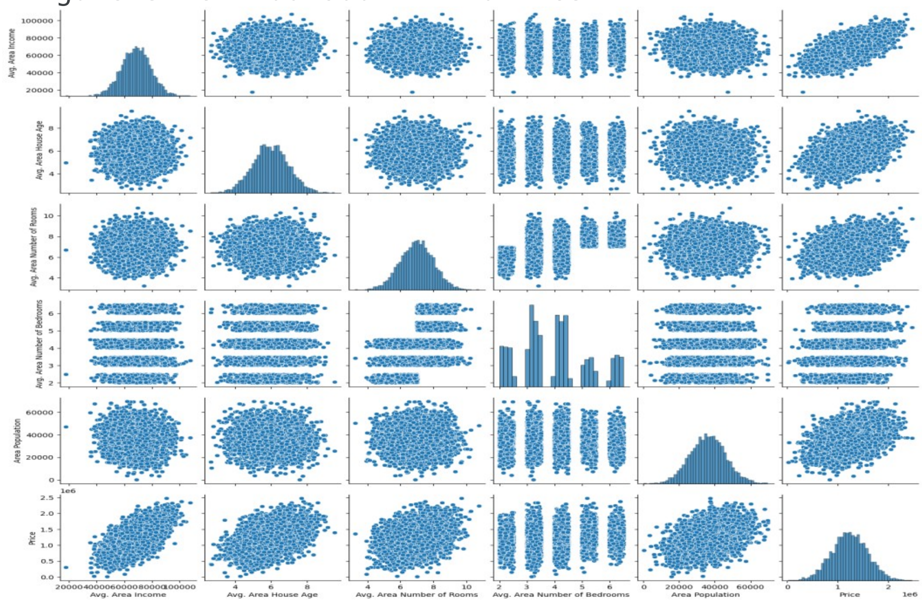
```
sns.jointplot(dataset, x='Avg. Area Income', y='Price')
```

```
<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>
```



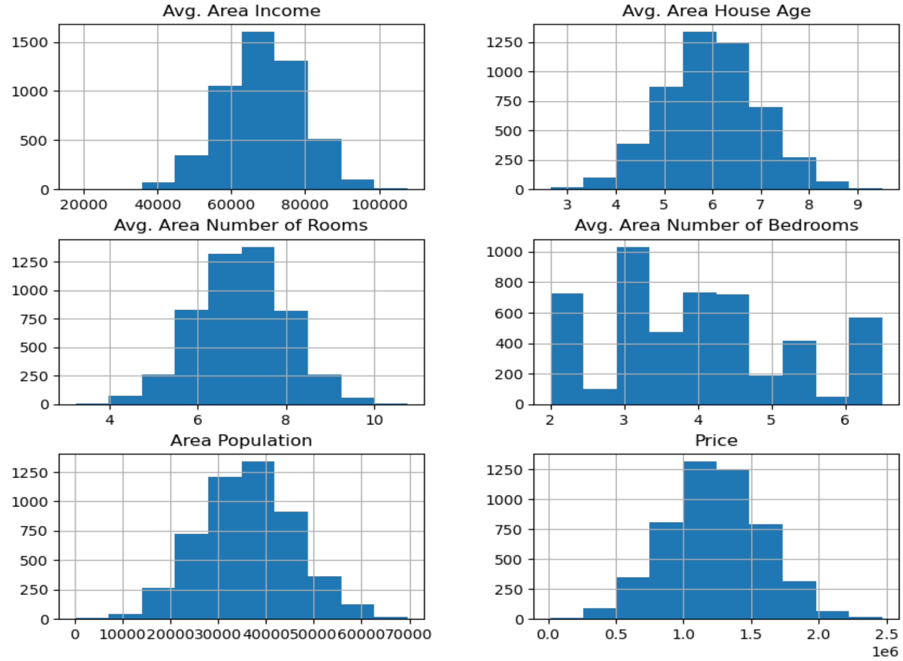

```
plt.figure(figsize=(12,8))
sns.pairplot(dataset)
```

```
<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>
<Figure size 1200x800 with 0 Axes>
```



```
dataset.hist(figsize=(10,8))
```

```
array([[<Axes: title={'center': 'Avg. Area Income'}>,
        <Axes: title={'center': 'Avg. Area House Age'}>],
       [<Axes: title={'center': 'Avg. Area Number of Rooms'}>,
        <Axes: title={'center': 'Avg. Area Number of Bedrooms'}>],
       [<Axes: title={'center': 'Area Population'}>,
        <Axes: title={'center': 'Price'}>]], dtype=object)
```



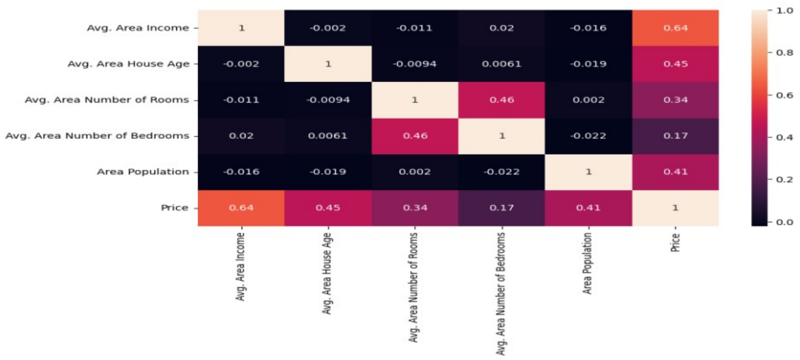
Visualising Correlation

```
dataset.corr(numeric_only=True)
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

```
plt.figure(figsize=(10,5))
sns.heatmap(dataset.corr(numeric_only = True), annot=True)
```

```
<Axes: >
```



3.Building the project by performing different activities like feature engineering, model training, evaluation etc

Feature engineering:

Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models. It involves creating new features, combining existing features, and transforming features into a format that is compatible with the machine learning algorithm being used.

Model training:

Model training is the process of feeding a machine learning model with data so that it can learn from it. This is done by providing the model with a set of input data and output data, and allowing the model to adjust its parameters in order to minimize the error between its predictions and the actual output data.

Model Evaluation

Model evaluation is the process of assessing the performance of a trained machine learning model on unseen data. This is done by providing the model with a set of test data and comparing its predictions to the actual output data. The goal of model evaluation is to ensure that the model is able to generalize well to new data and that it is not overfitting to the training data.

Steps involved in model training and evaluation:

- ❖ Split the data into training and test sets: The data is split into two sets: a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the performance of the model.
- ❖ Train the model: The model is trained on the training set. This involves feeding the model with the input data and output data from the training set and allowing it to adjust its parameters in order to minimize the error between its predictions and the actual output data.

Evaluate the model: The model is evaluated on the test set. This involves feeding the model with the input data from the test set and comparing its predictions to the actual output data. The performance of the model is measured using metrics such as accuracy, precision, recall, and F1

Program:

Importing Dependencies:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score,
mean_absolute_error,mean_squared_error
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg
```

Loading Dataset:

```
dataset = pd.read_csv('/kaggle/input/usa-housing/USA_Housing.csv')
```

Dividing Dataset in to features and target variable:

```
X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area
Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
Y = dataset['Price']
```

Using Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=101)
```

```
Y_train.shape
Y_test.head()
Y_test.shape
```

Standardizing the data

```
sc = StandardScaler()
X_train_scal = sc.fit_transform(X_train)
X_test_scal = sc.fit_transform(X_test)
```

Model Building and Evaluation

```
model_lr=LinearRegression()
```

```
model_lr.fit(X_train_scal, Y_train)
```

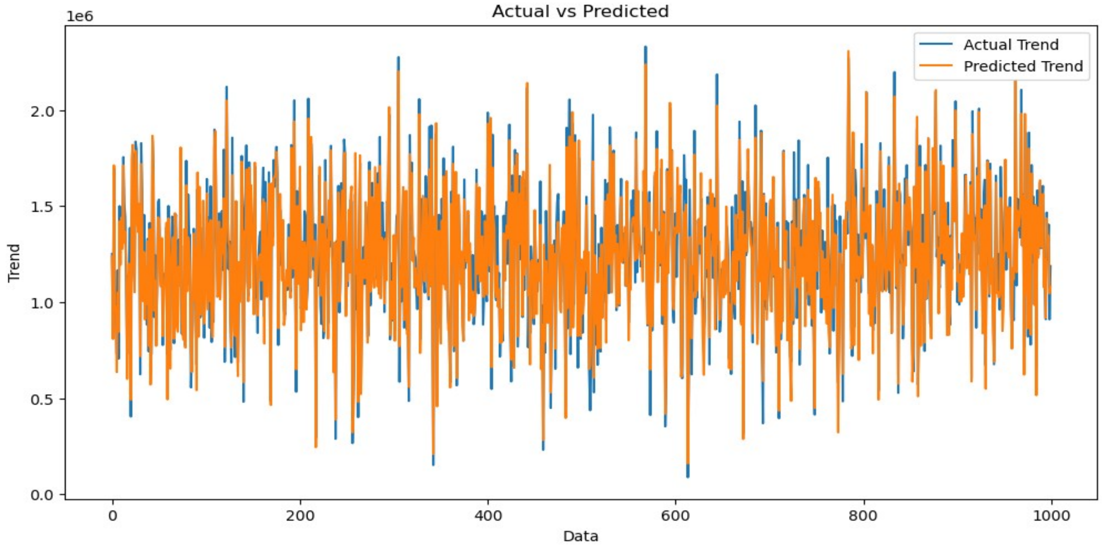
```
LinearRegression
LinearRegression()
```

Predicting Prices

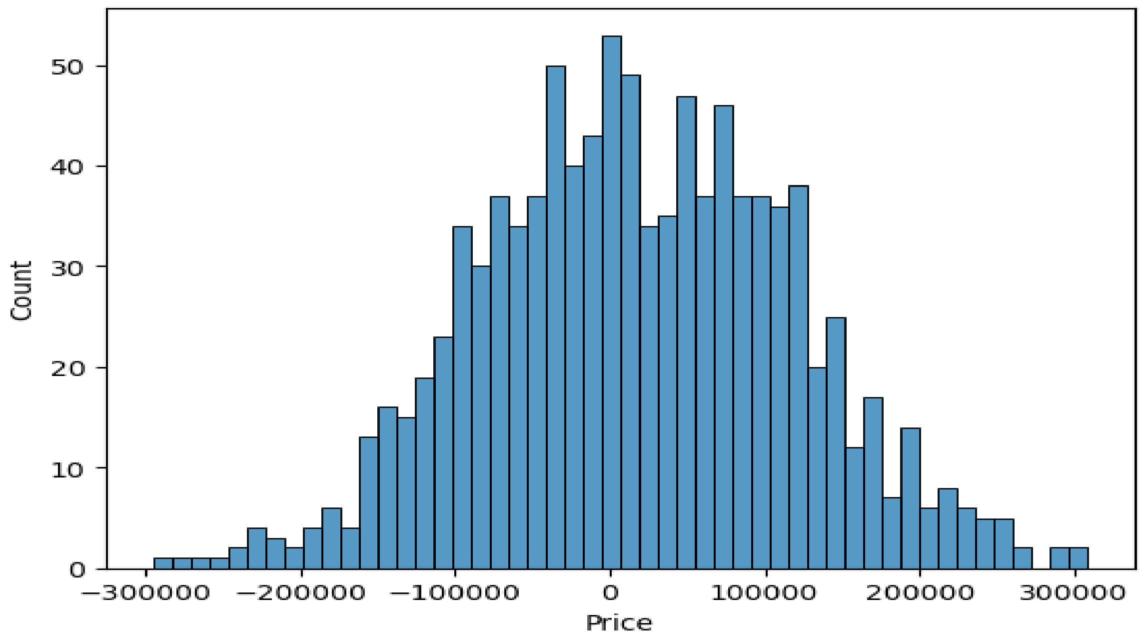
```
Prediction1 = model_lr.predict(X_test_scal)
```

Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
```




```
sns.histplot((Y_test-Prediction1), bins=50)
```




```
print(r2_score(Y_test, Prediction1))
print(mean_absolute_error(Y_test, Prediction1))
print(mean_squared_error(Y_test, Prediction1))
```

Model 2 - Support Vector Regressor

```
model_svr = SVR()
model_svr.fit(X_train_scal, Y_train)
```

SVR

```
SVR()
```

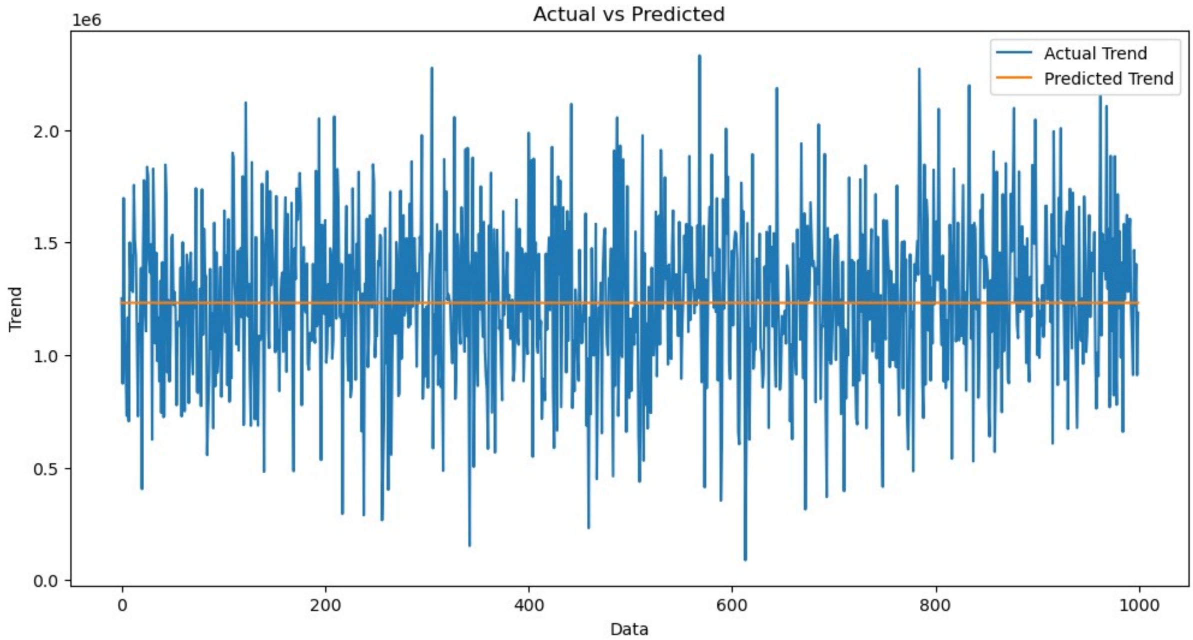
Predicting Prices

```
Prediction2 = model_svr.predict(X_test_scal)
```

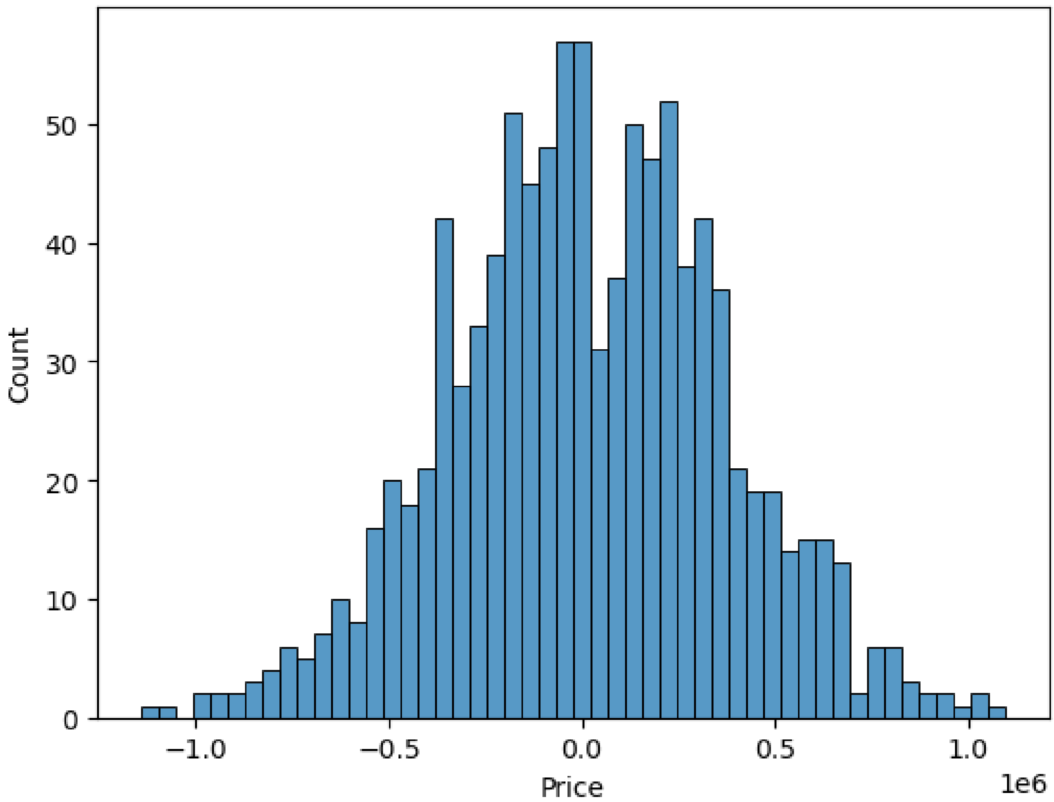
Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
```

```
plt.title('Actual vs Predicted')
```




```
sns.histplot((Y_test-Prediction2), bins=50)
```




```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

Model 3 - Lasso Regression

```
model_lar = Lasso(alpha=1)
model_lar.fit(X_train_scal,Y_train)
```

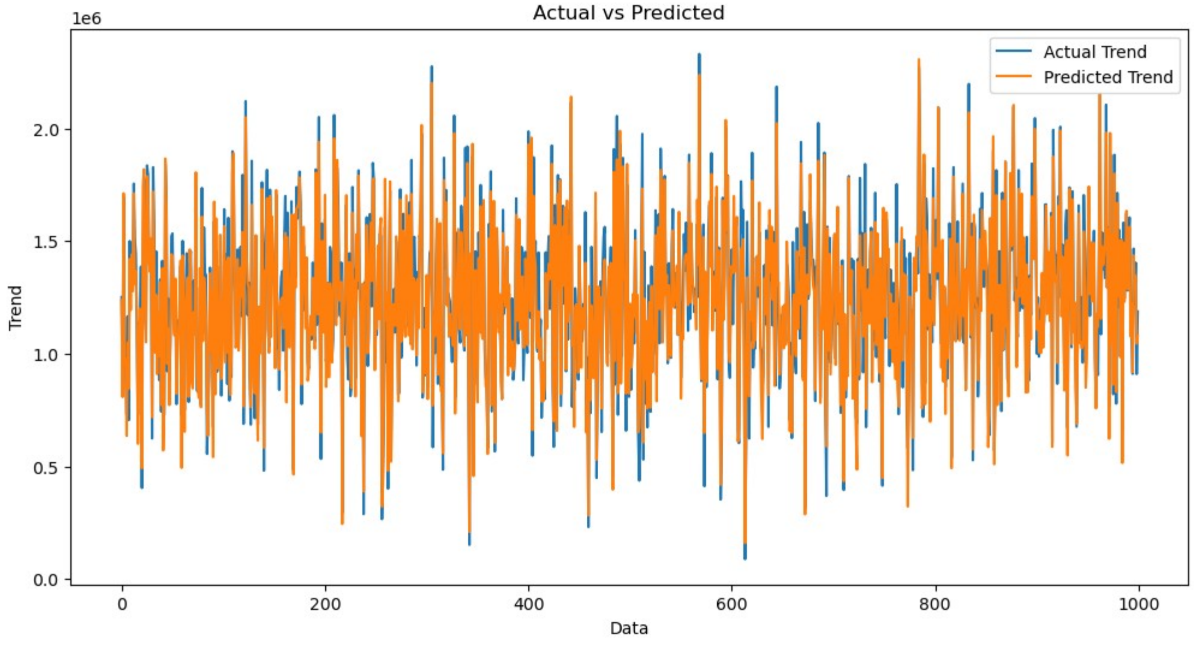
```
Lasso
Lasso(alpha=1)
```

Predicting Prices

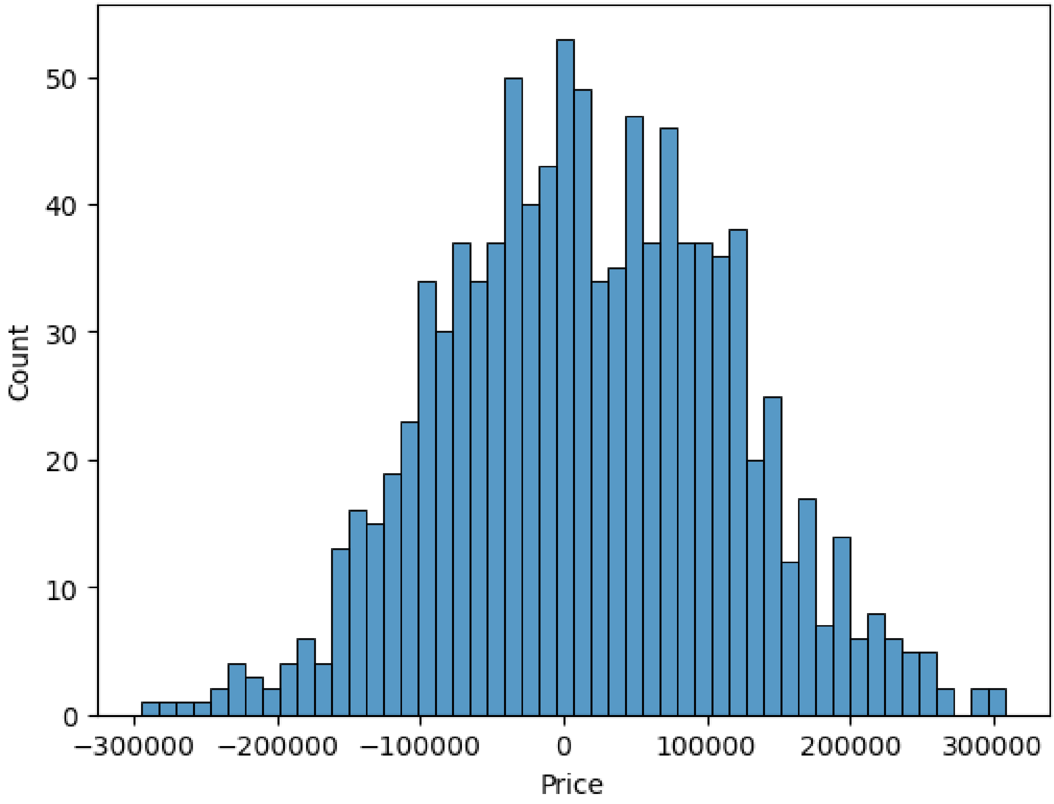
```
Prediction3 = model_lar.predict(X_test_scal)
```

Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction3, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
```




```
sns.histplot((Y_test-Prediction3), bins=50)
```




```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2)
```

Model 4 - Random Forest Regressor

```
model_rf = RandomForestRegressor(n_estimators=50)
model_rf.fit(X_train_scal, Y_train)
```

RandomForestRegressor

```
RandomForestRegressor(n_estimators=50)
```

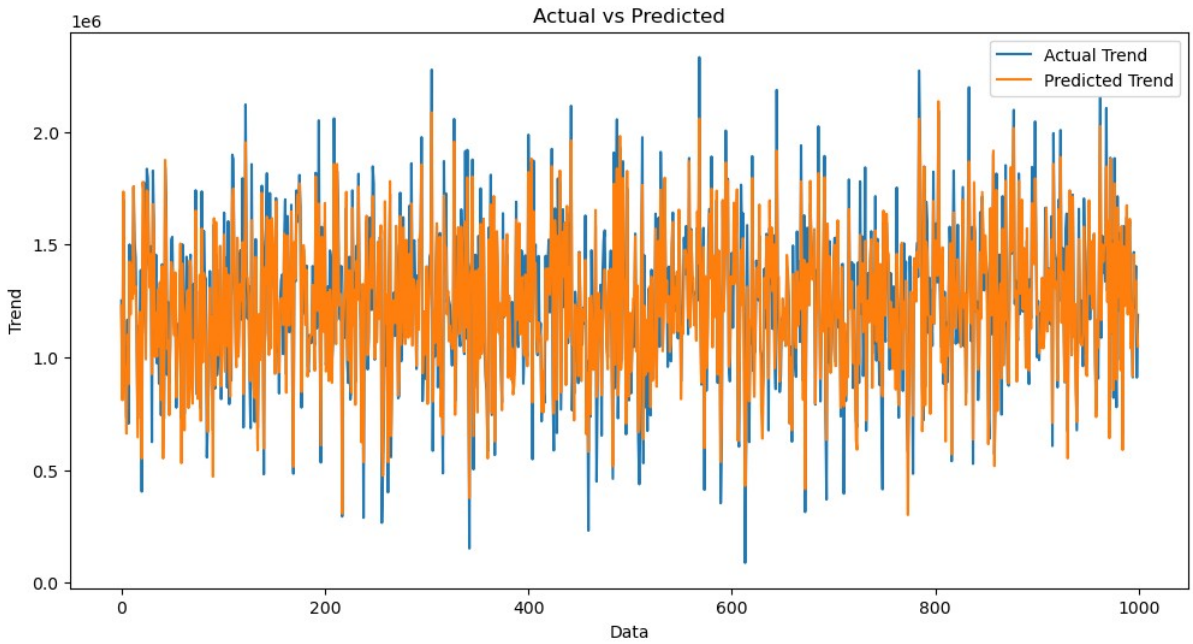
Predicting Prices

```
Prediction4 = model_rf.predict(X_test_scal)
```

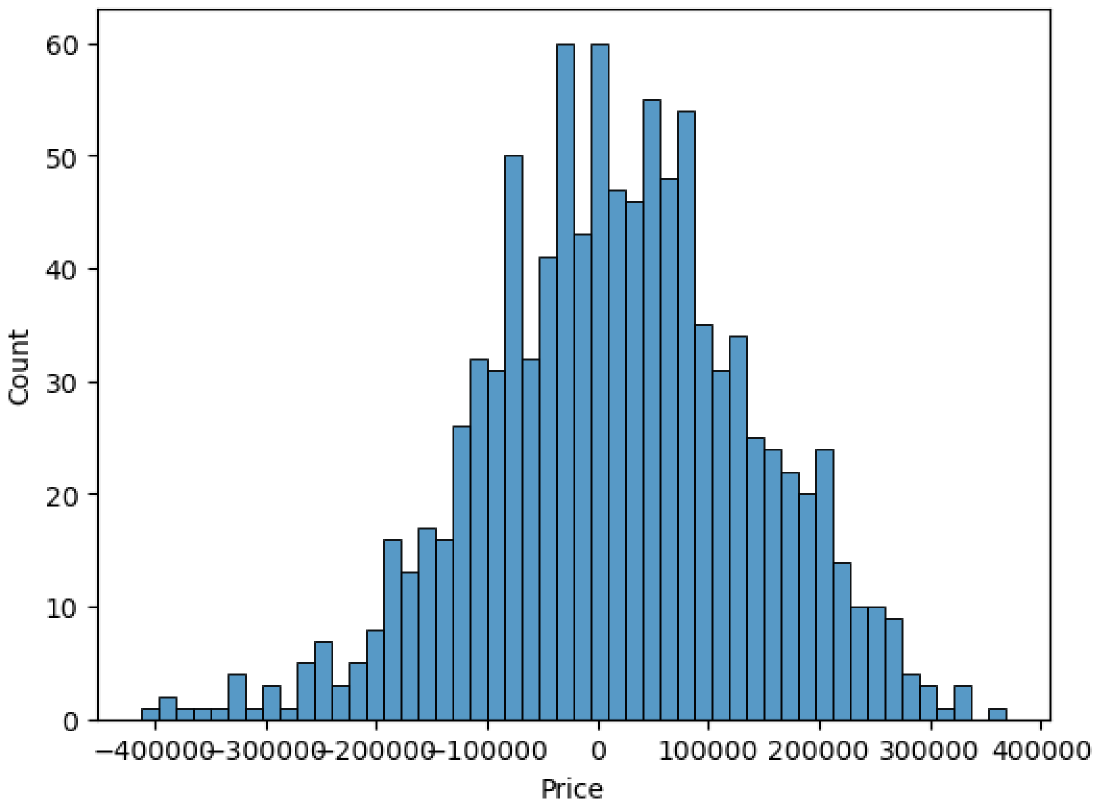
Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted
Trend')
```

```
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
```




```
sns.histplot((Y_test-Prediction4), bins=50)
```



```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

Model 5 - XGboost Regressor

```
model_xg = xg.XGBRegressor()
model_xg.fit(X_train_scal, Y_train)
```

XGBRegressor

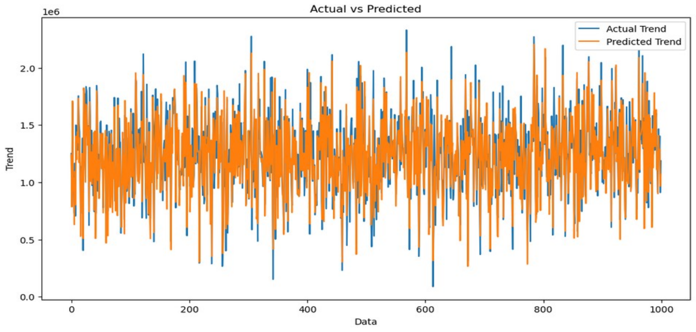
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
```

Predicting Prices

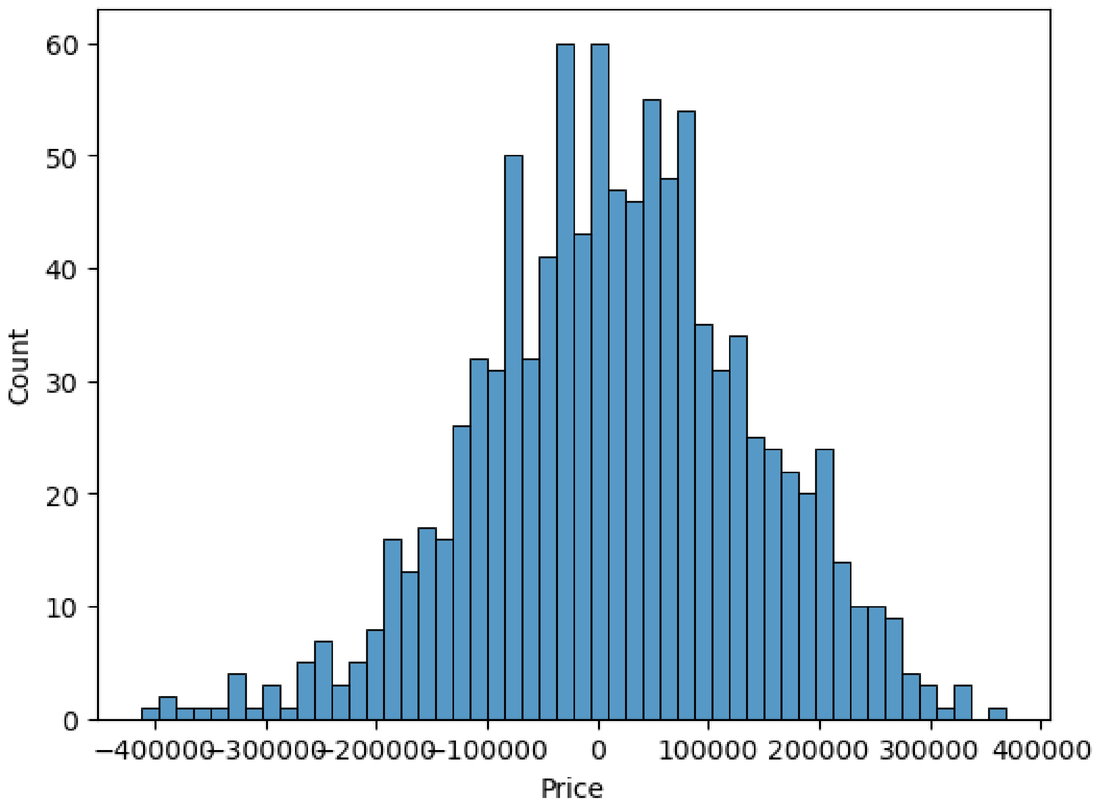
```
Prediction5 = model_xg.predict(X_test_scal)
```

Evaluation of Predicted Data

```
plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction5, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
```




```
sns.histplot((Y_test-Prediction4), bins=50)
```




```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

CONCLUSION

Despite the challenges, machine learning is a powerful tool that can be used to develop accurate and scalable models for house price prediction. Machine learning models can be used by a variety of stakeholders, including homeowners, real estate agents, and mortgage lenders etc.

THANK YOU