PREDITING HOUSE PRICES USING ARTIFICIAL INTELLIGENCE

BY M.KANISH PRETHIVE

PHASE IV: DEVOLOUPMENT PART 2

INTRODUCTION

Predicting house prices using artificial intelligence (AI) is a complex task, but it is one that has become increasingly feasible in recent years. AI models can be used to analyze a wide range of data factors, such as historical sales data, property characteristics, and neighborhood demographics, to generate accurate predictions of future house prices.

To predict house prices using AI, you will need to collect a dataset of historical sales data, property characteristics, and neighborhood demographics. You will then need to train an AI model on this dataset. Once the model is trained, you can use it to predict the prices of new properties.

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

Necessory steps to follow:

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 score.

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
```

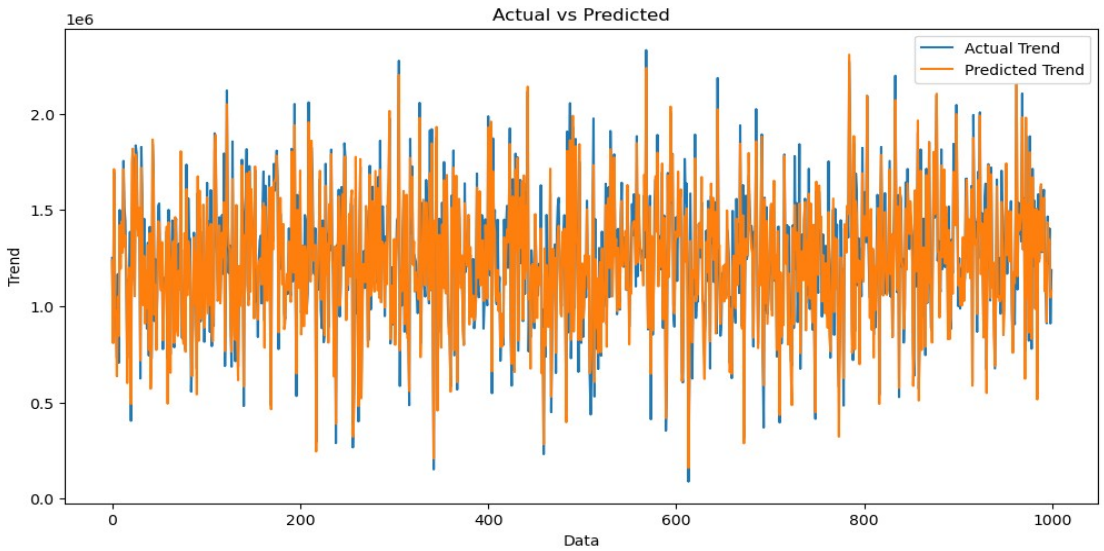
```
LinearRegression()
```

Predicting Prices

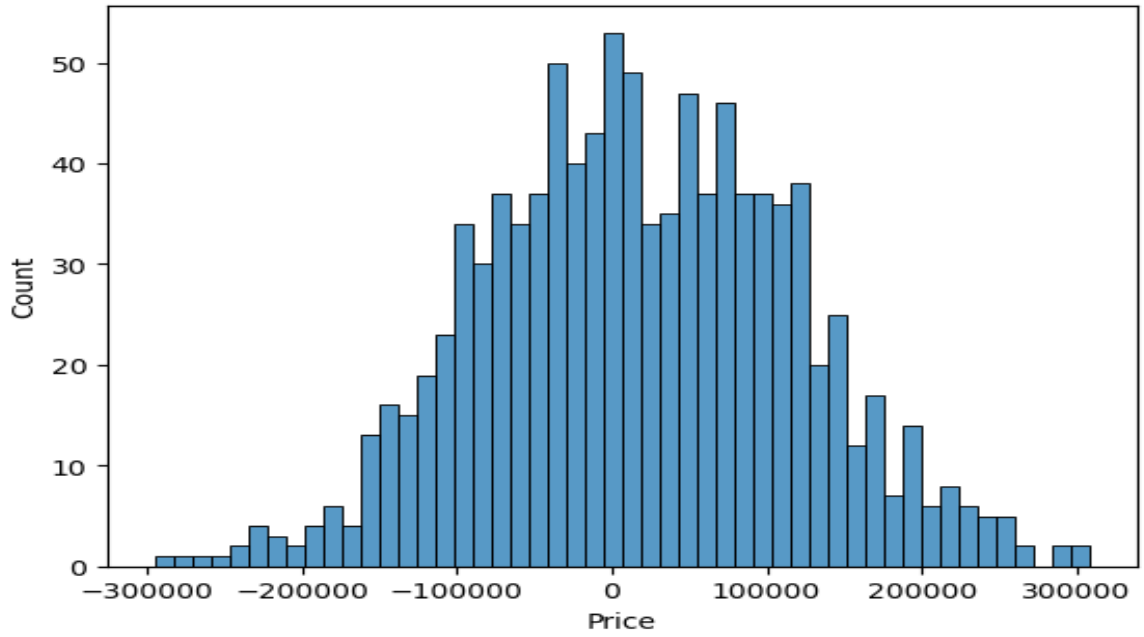
```
Prediction1 = model_lr.predict(X_test_scal)
```

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)
```

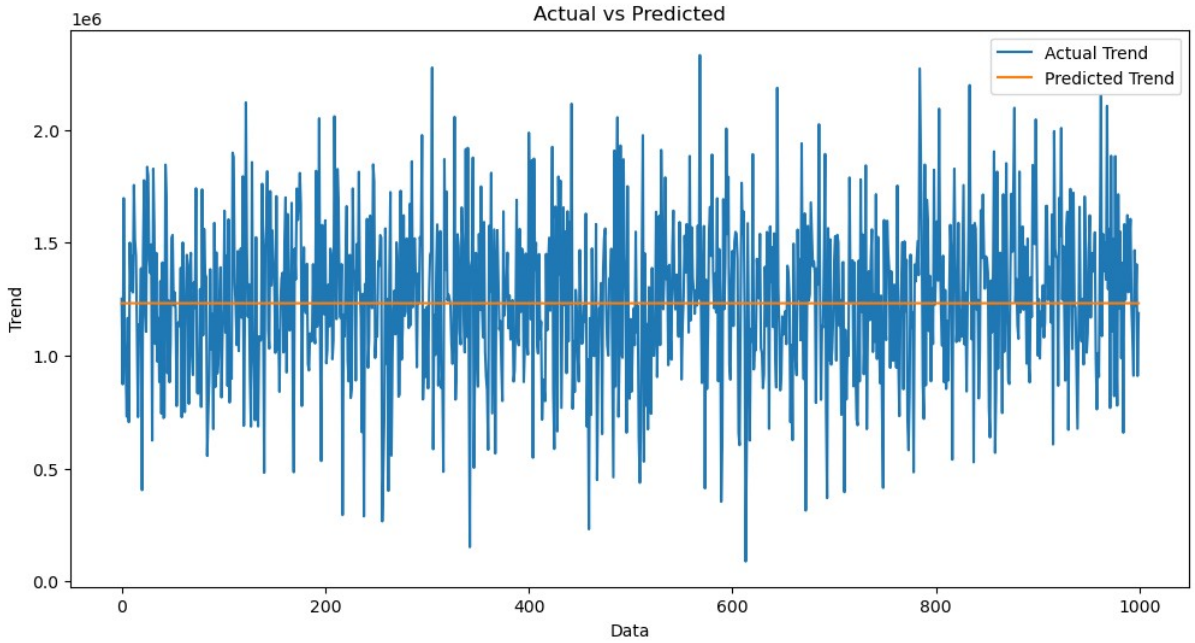
```
SVR
SVR()
```

Predicting Prices

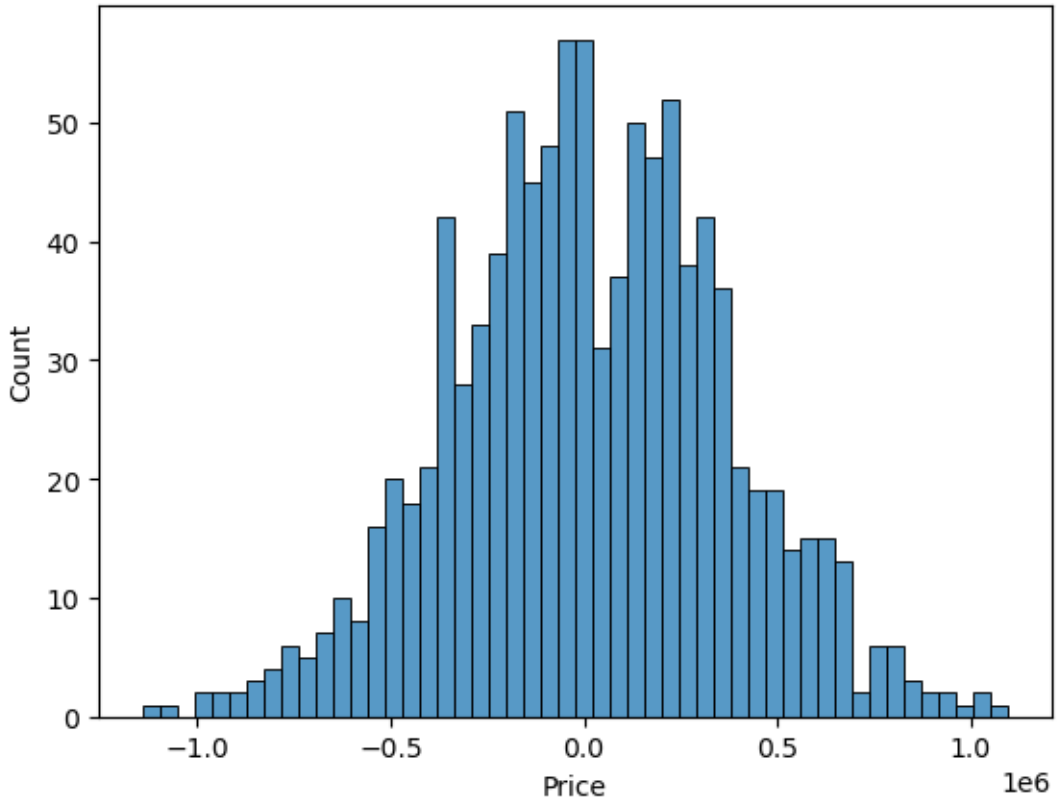
```
Prediction2 = model_svr.predict(X_test_scal)
```

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)
```

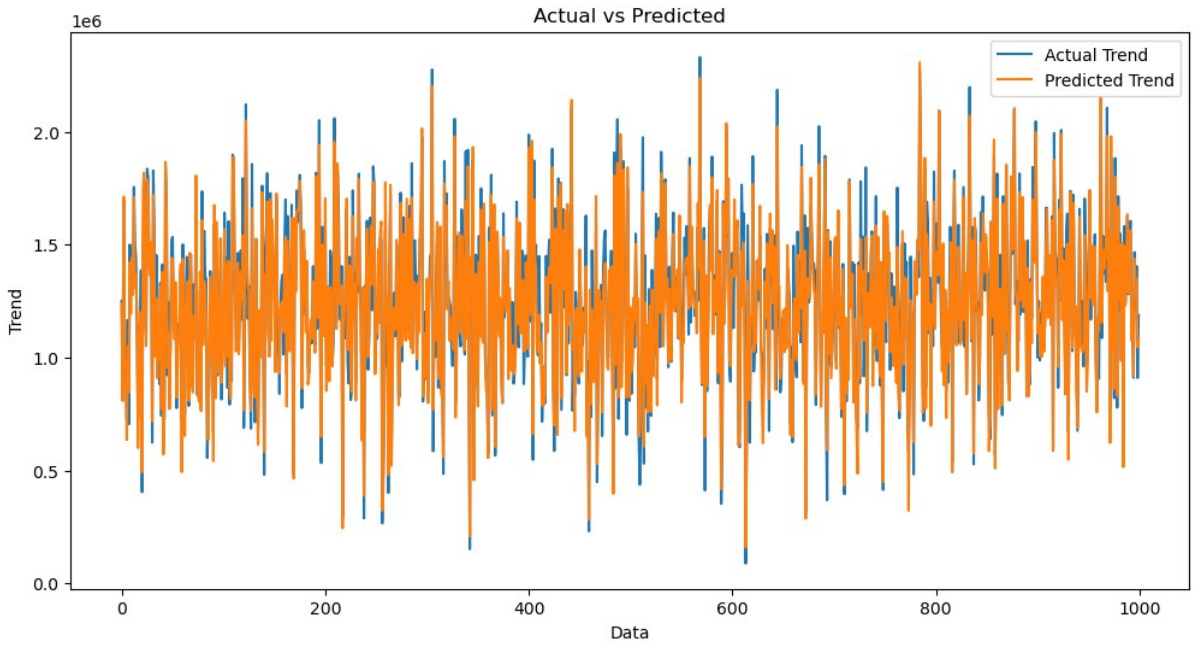
```
Lasso
Lasso(alpha=1)
```

Predicting Prices

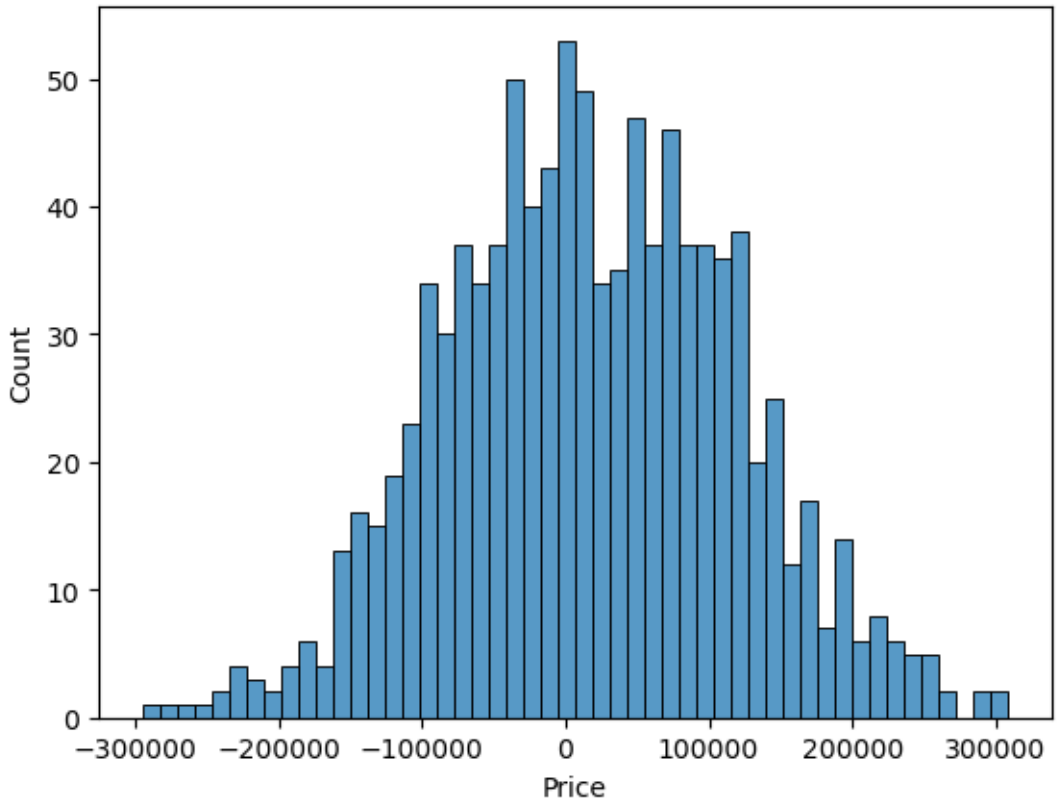
```
Prediction3 = model_lar.predict(X_test_scal)
```

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)
```

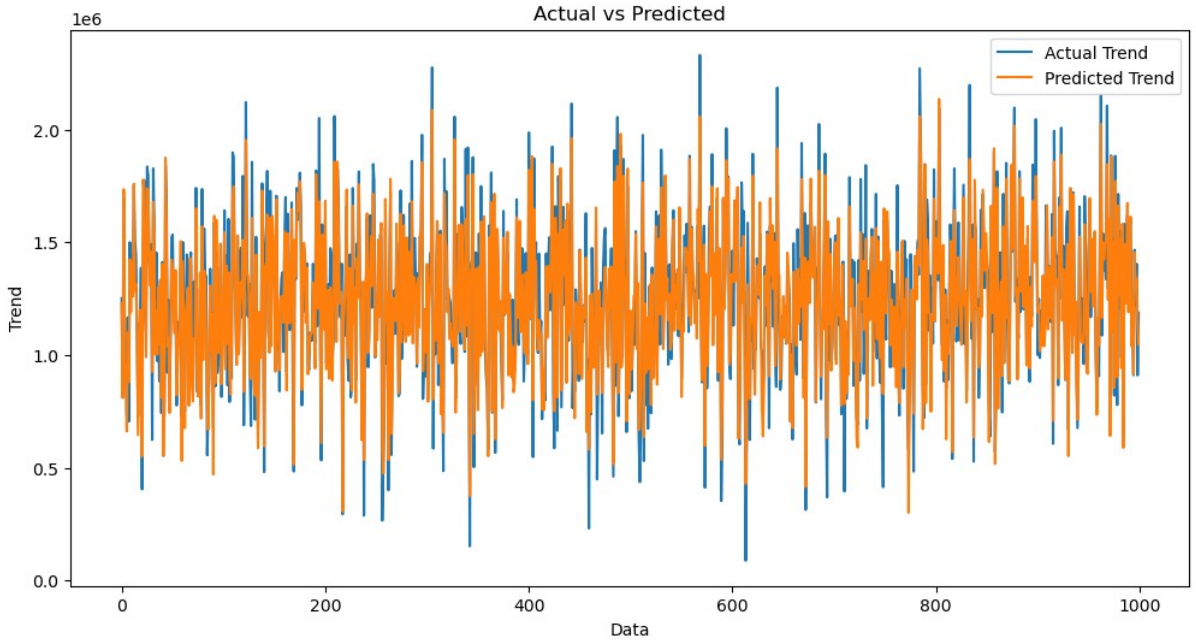
Predicting Prices

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

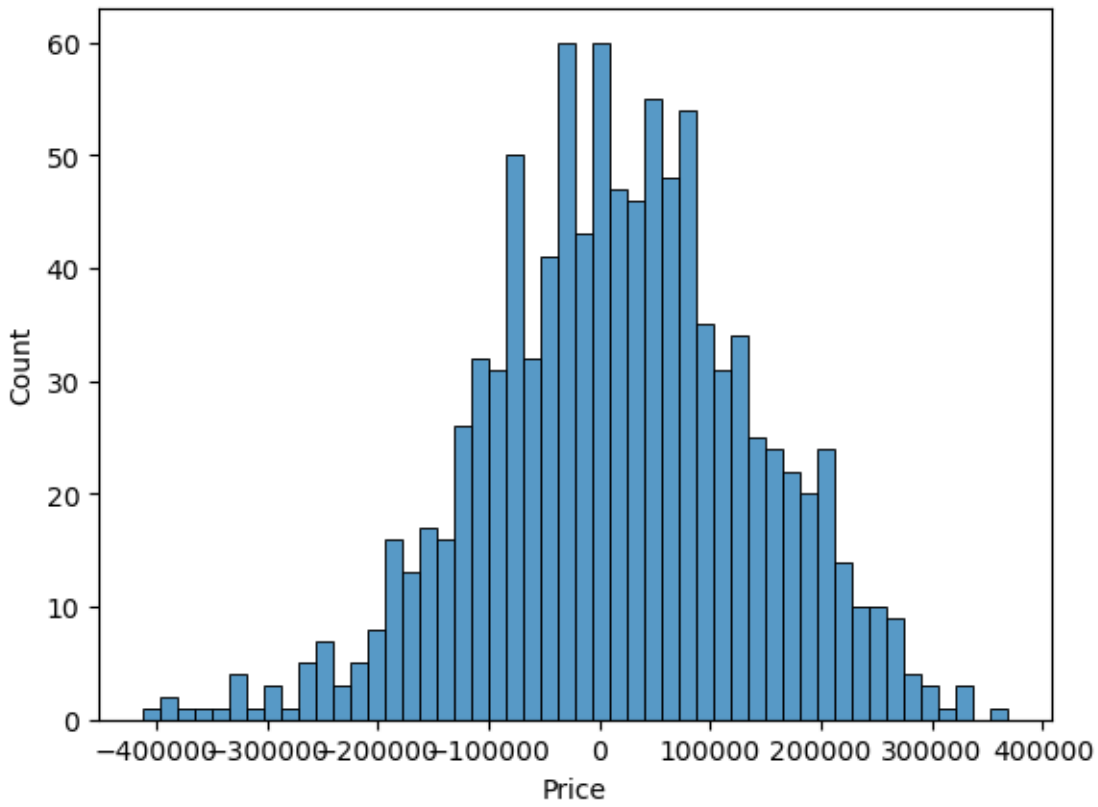
Evaluation of Predicted Data

```
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)
```

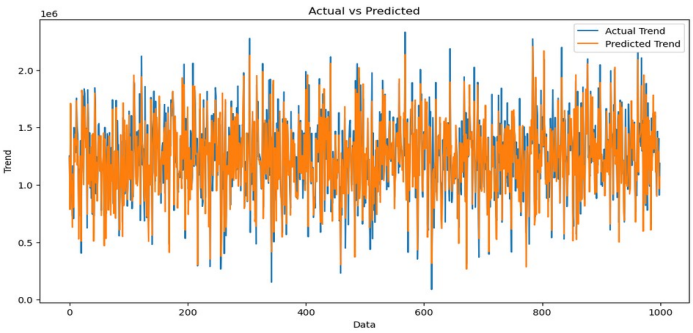
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
```

Predicting Prices

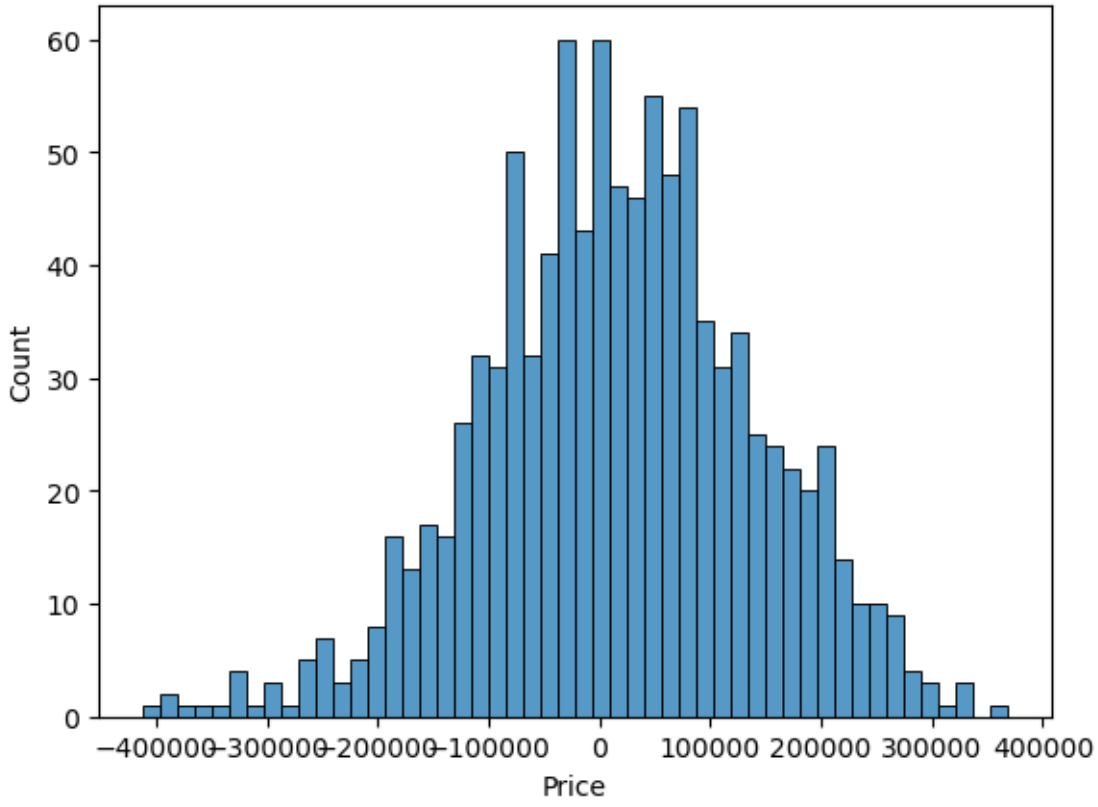
```
Prediction5 = model_xg.predict(X_test_scal)
```

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))
```

THANKING YOU