

PREDICTING HOUSE PRICE USING MACHINE LEARNING

Phase 2 Submission Document

Project: House Price Prediction



Introduction:

☐ The real estate market is one of the most dynamic and lucrative sectors, with house prices constantly fluctuating based on various factors such as location, size, amenities, and economic conditions. Accurately predicting house prices is crucial for both buyers and sellers, as it can help make informed decisions regarding buying, selling, or investing in properties.

☐ Traditional linear regression models are often employed for house price prediction. However, they may not capture complex relationships between predictors and the target variable, leading to suboptimal predictions. In this project, we will explore advanced regression techniques to enhance the accuracy and robustness of house price prediction models.

☐ Briefly introduce the real estate market and the importance of accurate house price prediction. Highlight the limitations of traditional linear regression models in capturing complex relationships.

② Emphasize the need for advanced regression techniques like Gradient Boosting and XGBoost to enhance prediction accuracy.

Content for Project Phase 2 :

Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for improved Prediction accuracy.

Data Source

A good data source for house price prediction using machine learning should be Accurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: (<https://www.kaggle.com/datasets/vedavyasv/usa-housing>)

| Avg. Area Income | Avg. Area House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area Population | Price | Address |
|------------------|---------------------|---------------------------|------------------------------|-----------------|------------|-----------|
| 79545.45857 | 5.682861322 | 7.009188143 | 4.09 | 23086.8005 | 1059033.56 | 208 |
| 79248.64245 | 6.002899808 | 6.730821019 | 3.09 | 40173.07217 | 1505890.91 | 188 |
| 61287.06718 | 5.86588984 | 8.51272743 | 5.13 | 36882.1594 | 1058987.99 | 9127 |
| 63345.24005 | 7.188236095 | 5.586728665 | 3.26 | 34310.24283 | 1260616.81 | USS |
| 59982.19723 | 5.040554523 | 7.839387785 | 4.23 | 26354.10947 | 630943.489 | USNS |
| 80175.75416 | 4.988407758 | 6.104512439 | 4.04 | 26748.42842 | 1068138.07 | 06039 |
| 64698.46343 | 6.025335907 | 8.147759585 | 3.41 | 60828.24909 | 1502055.82 | 4759 |
| 78394.33928 | 6.989779748 | 6.620477995 | 2.42 | 36516.35897 | 1573936.56 | 972 Joyce |
| 59927.66081 | 5.36212557 | 6.393120981 | 2.3 | 29387.396 | 798869.533 | USS |
| 81885.92718 | 4.42367179 | 8.167688003 | 6.1 | 40149.96575 | 1545154.81 | Unit 9446 |
| 80527.47208 | 8.093512681 | 5.0427468 | 4.1 | 47224.35984 | 1707045.72 | 6368 |
| 50593.6955 | 4.496512793 | 7.467627404 | 4.49 | 34343.99189 | 663732.397 | 911 |
| 39033.80924 | 7.671755373 | 7.250029317 | 3.1 | 39220.36147 | 1042814.1 | 209 |
| 73163.66344 | 6.919534825 | 5.993187901 | 2.27 | 32326.12314 | 1291331.52 | 829 |
| 69391.38018 | 5.344776177 | 8.406417715 | 4.37 | 35521.29403 | 1402818.21 | PSC 5330, |
| 73091.86675 | 5.443156467 | 8.517512711 | 4.01 | 23929.52405 | 1306674.66 | 2278 |
| 79706.96306 | 5.067889591 | 8.219771123 | 3.12 | 39717.81358 | 1556786.6 | 064 |
| 61929.07702 | 4.788550242 | 5.097009554 | 4.3 | 24595.9015 | 528485.247 | 5498 |
| 63508.1943 | 5.94716514 | 7.187773835 | 5.12 | 35719.65305 | 1019425.94 | Unit 7424 |
| 62085.2764 | 5.739410844 | 7.091808104 | 5.49 | 44922.1067 | 1030591.43 | 19696 |
| 86294.99909 | 6.62745694 | 8.011897853 | 4.07 | 47560.77534 | 2146925.34 | 030 Larry |
| 60835.08998 | 5.551221592 | 6.517175038 | 2.1 | 45574.74166 | 929247.6 | USNS |
| 64490.65027 | 4.21032287 | 5.478087731 | 4.31 | 40358.96011 | 718887.232 | 95198 |
| 60697.35154 | 6.170484091 | 7.150536572 | 6.34 | 28140.96709 | 743999.819 | 9003 Jay |
| 59748.85549 | 5.339339881 | 7.748681606 | 4.23 | 27809.98654 | 895737.133 | 24282 |

Data Collection and Preprocessing:

- ❑ Importing the dataset: Obtain a comprehensive dataset containing relevant features such as square footage, number of bedrooms, location, amenities, etc.
- ❑ Data preprocessing: Clean the data by handling missing values, outliers, and categorical variables. Standardize or normalize numerical features.

Exploratory Data Analysis (EDA):

- ❑ Visualize and analyze the dataset to gain insights into the relationships between variables.
- ❑ Identify correlations and patterns that can inform feature selection and engineering.
- ❑ Present various data visualizations to gain insights into the dataset.
- ❑ Explore correlations between features and the target variable (house prices).
- ❑ Discuss any significant findings from the EDA phase that inform feature selection.

Feature Engineering:

- ❑ Create new features or transform existing ones to capture valuable information.
- ❑ Utilize domain knowledge to engineer features that may impact house prices, such as proximity to schools, transportation, or crime rates.
- ❑ Explain the process of creating new features or transforming existing ones.
- ❑ Showcase domain-specific feature engineering, such as proximity scores or composite indicators.
- ❑ Emphasize the impact of engineered features on model performance.

Advanced Regression Techniques:

➤ **Ridge Regression:** Introduce L2 regularization to mitigate overfitting.

➤ **Lasso Regression:** Employ L1 regularization to promote feature selection and simplify the model.

➤ **ElasticNet Regression:** Combine both L1 and L2 regularization to benefit from their respective advantages.

➤ **Random Forest Regression:** Implement an ensemble to handle non-linearity and capture complex relationships in the data.

➤ **Gradient Boosting Regressors (e.g., XGBoost, LightGBM):** Utilize gradient boosting algorithms for improved accuracy.

Model Evaluation and Selection:

- Split the dataset into training and testing sets.
- Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, 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 in Gradient Boosting and XGBoost models.
- Discuss the insights gained from feature importance 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 features and receive price predictions.

Program:

House Price Prediction

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('E:/USA_Housing.csv')
```

Model 1 - Linear Regression

In [1]:

```
model_lr=LinearRegression()
```

In [2]:

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

Out[2]:

```
LinearRegression
LinearRegression()
```

Predicting Prices

In [3]:

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

Evaluation of Predicted Data

In [4]:

```
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')
```

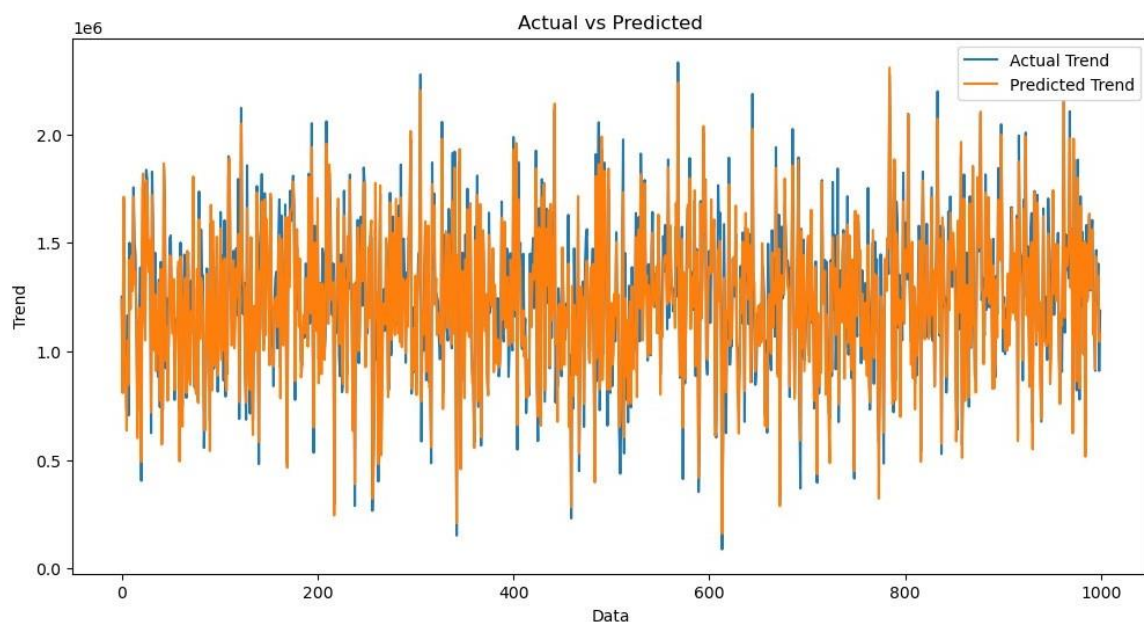
```
plt.ylabel('Trend')
```

```
plt.legend()
```

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

Out[4]:

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

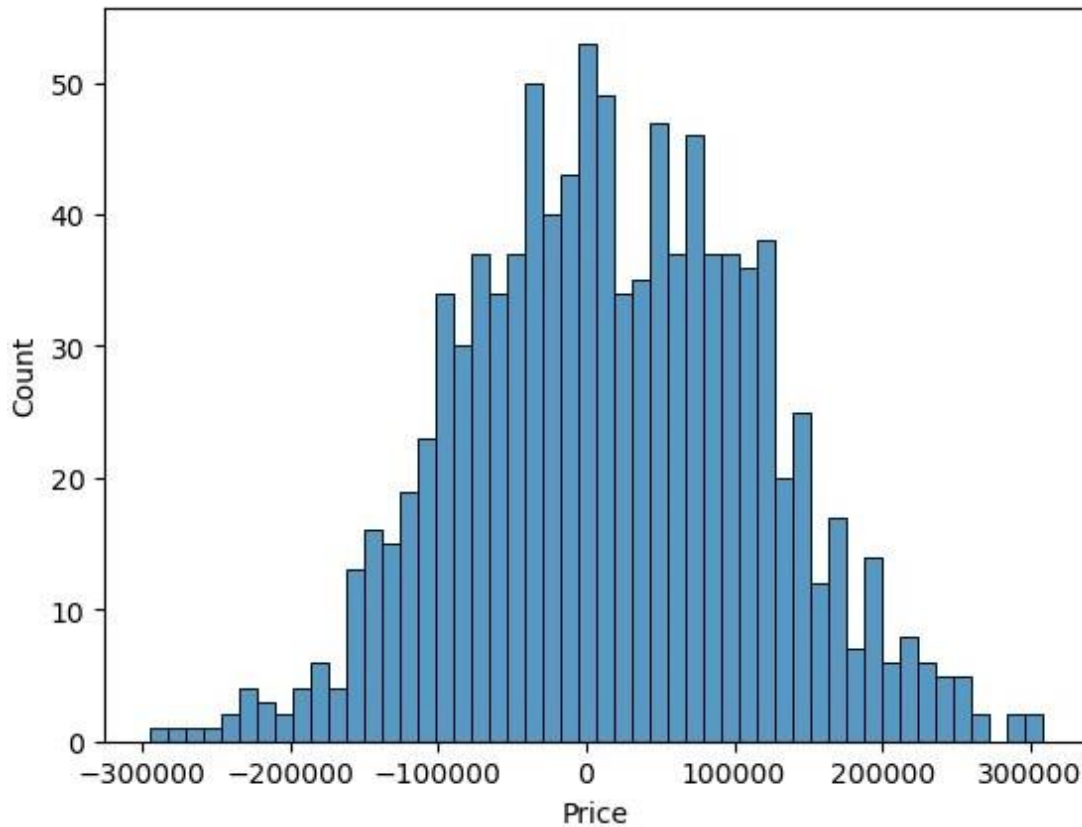


In [5]:

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

Out[5]:

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



In [6]:

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

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

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

Out[6]:

0.9182928179392918

82295.49779231755

10469084772.975954

Model 2 - Support Vector Regressor

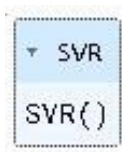
In [7]:

```
model_svr = SVR()
```

In [8]:

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

Out[8]:



Predicting Prices

In [9]:

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

Evaluation of Predicted Data

In [10]:

```
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')
```

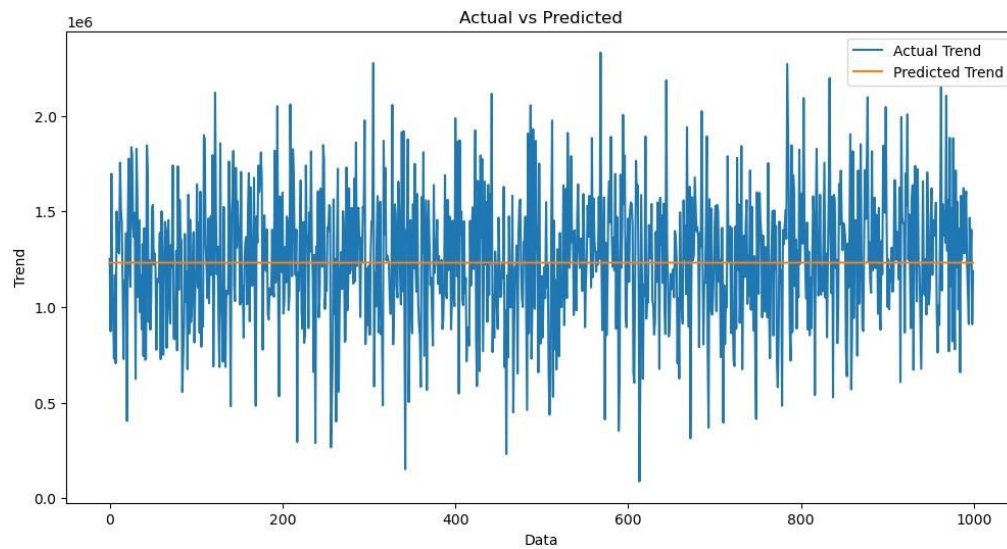
```
plt.ylabel('Trend')
```

```
plt.legend()
```

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


Out[10]:

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

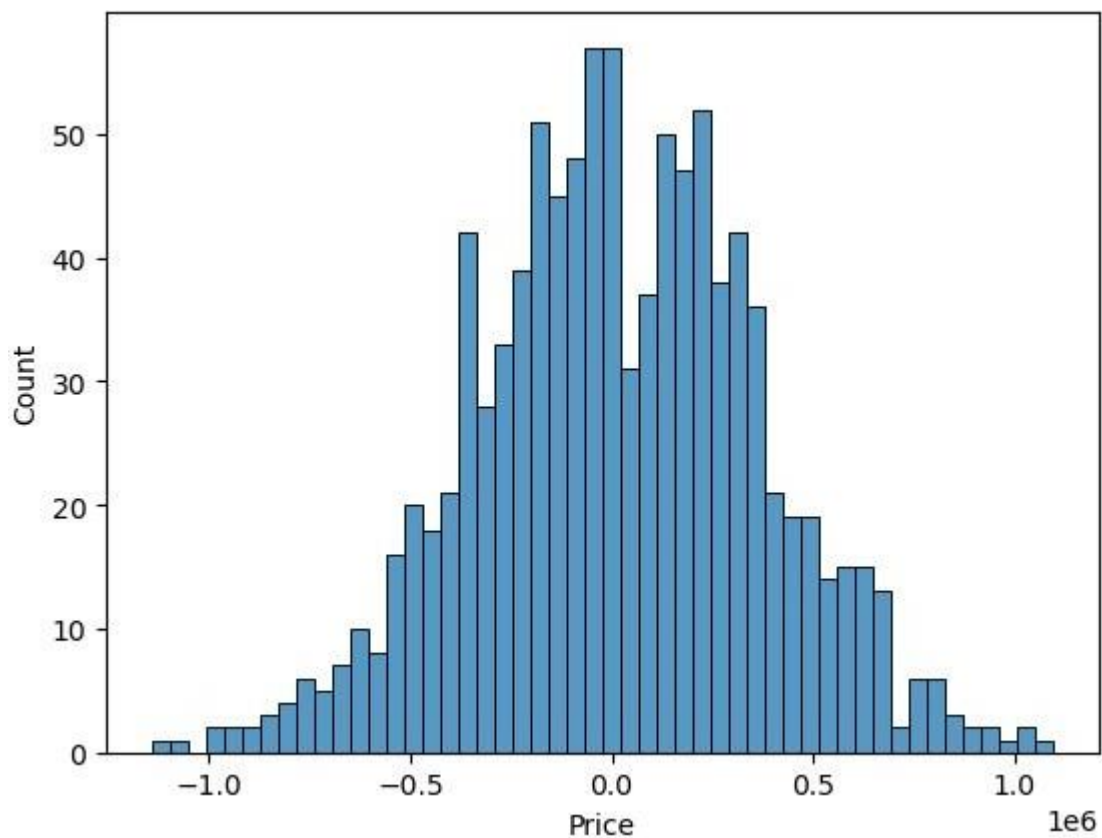


In [11]:

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

Out[12]:

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



In [12]:

```
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

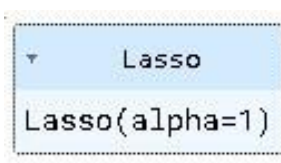
In [13]:

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

In [14]:

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

Out[14]:



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

Predicting Prices

In [15]:

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

Evaluation of Predicted Data

In [16]:

```
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')

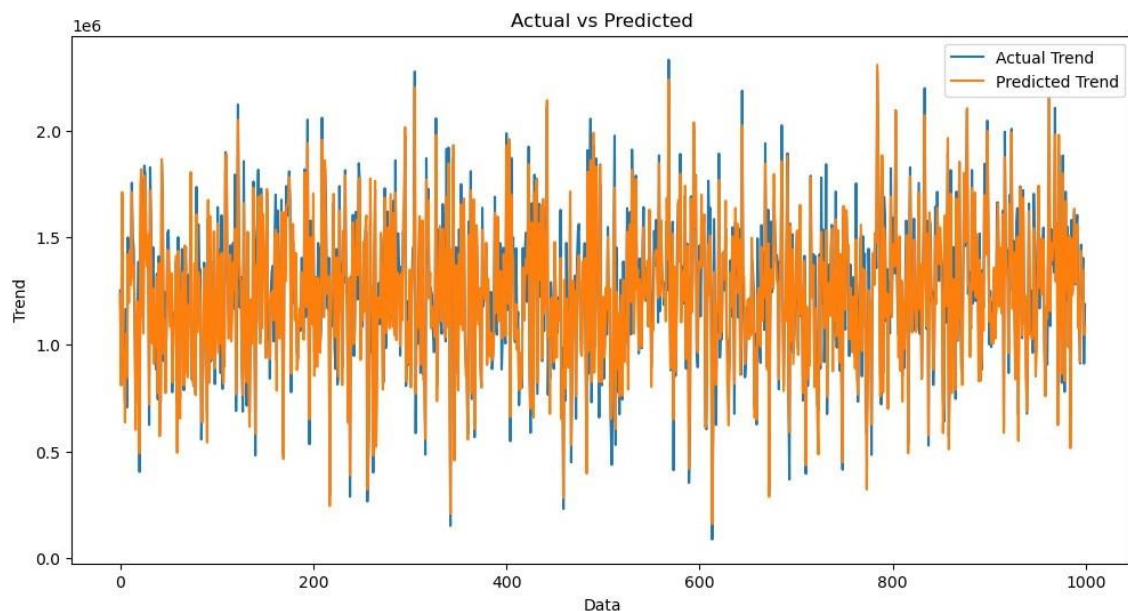
plt.ylabel('Trend')

plt.legend()

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

Out[16]:

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

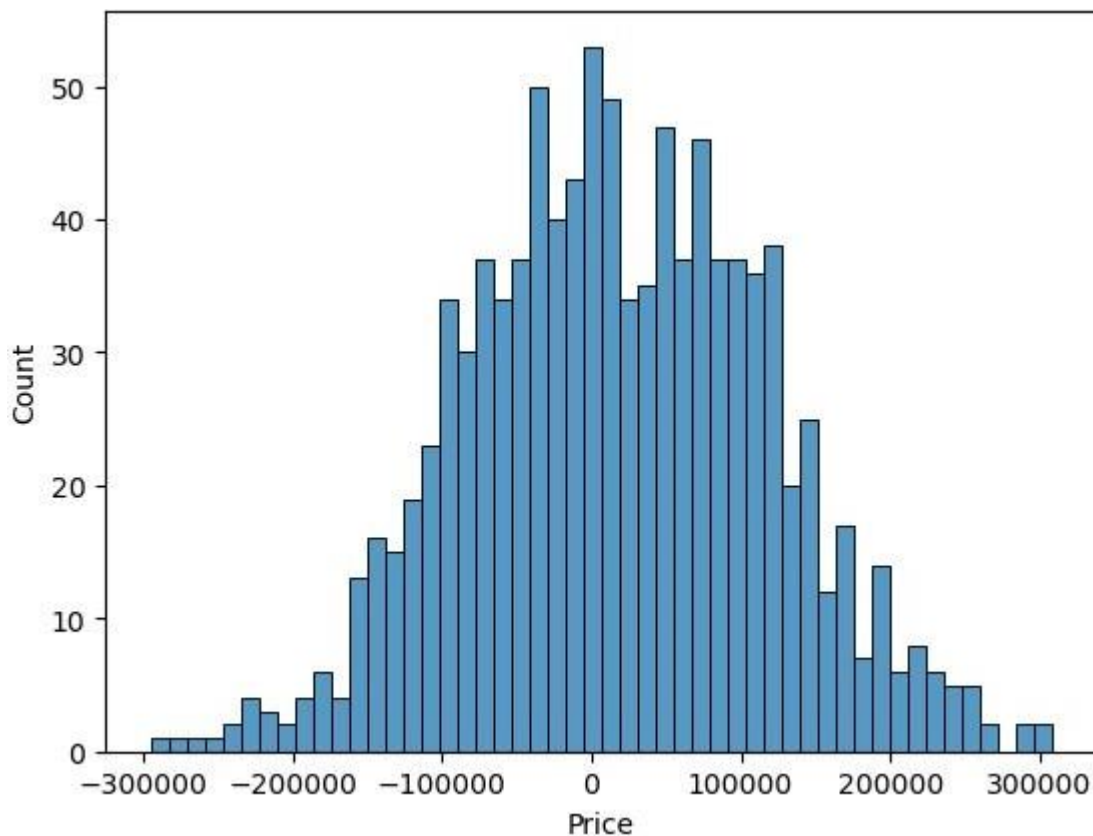


In [17]:

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

Out[17]:

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



In [18]:

```
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 [19]:

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

In [20]:

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

Out[20]:

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

Predicting Prices

In [21]:

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

Evaluation of Predicted Data

In [22]:

```
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')
```

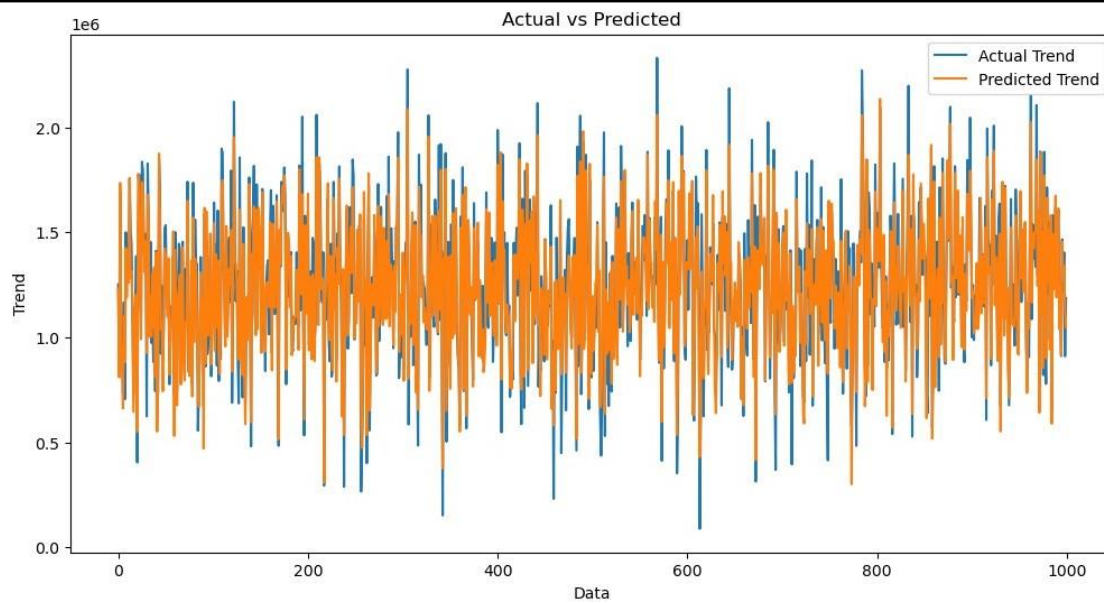
```
plt.ylabel('Trend')
```

```
plt.legend()
```

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

Out[22]:

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

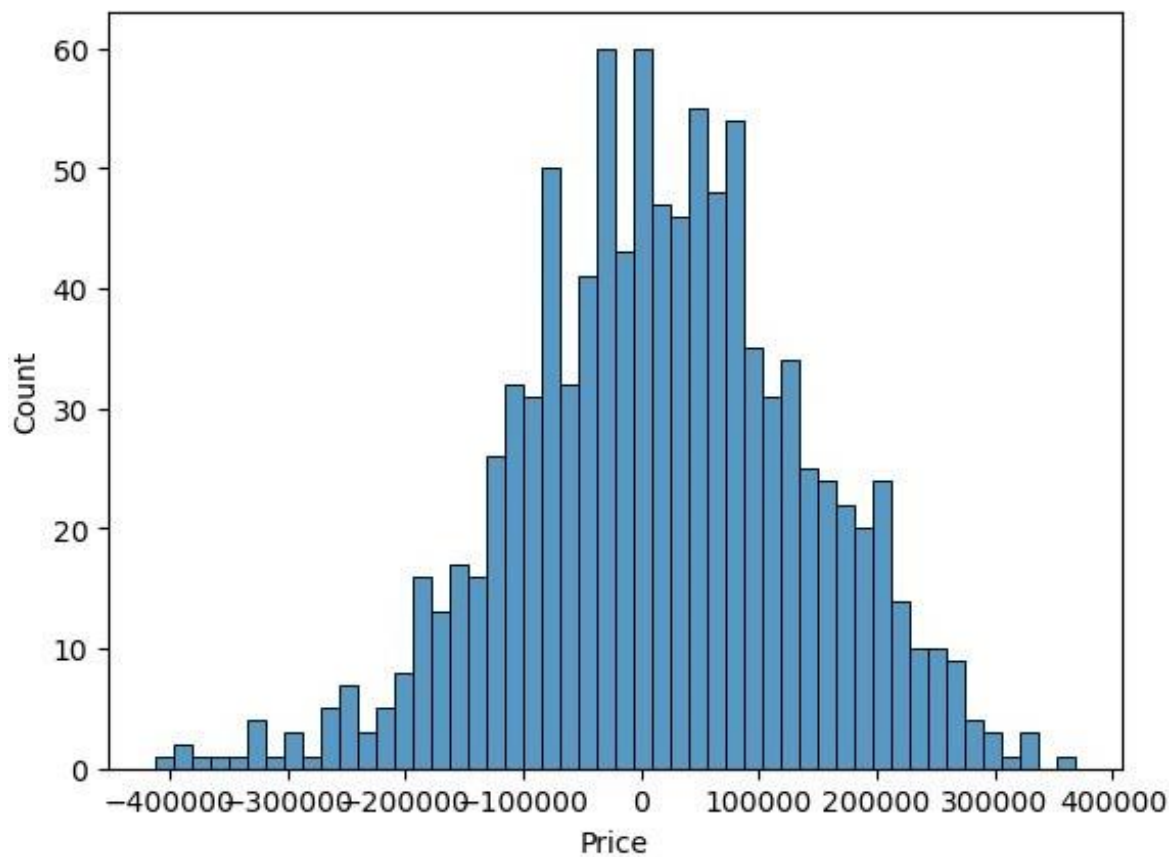


In [23]:

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

Out[23]:

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



In [24]:

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

Out [24] :

```
-0.0006222175925689744  
  
286137.81086908665  
  
128209033251.4034
```

Model 5 - XGboost Regressor

In [25]:

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

In [26]:

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

Out[26]:

```
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 [27]:

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

Evaluation of Predicted Data

In [28]:

```
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')
```

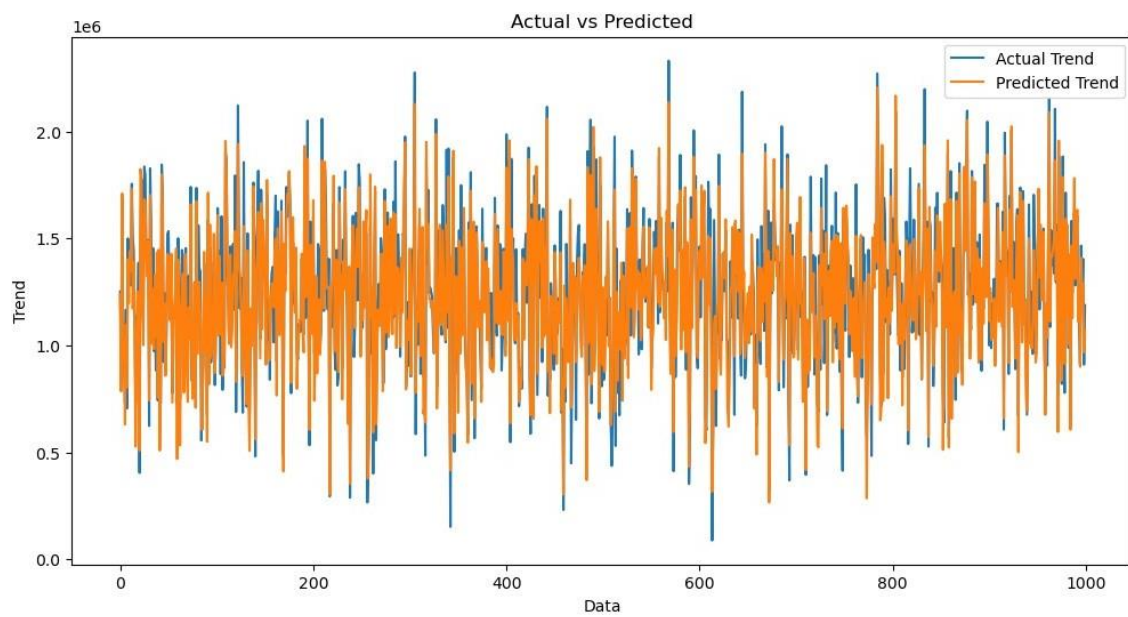
```
plt.ylabel('Trend')
```

```
plt.legend()
```

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

Out[28]:

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

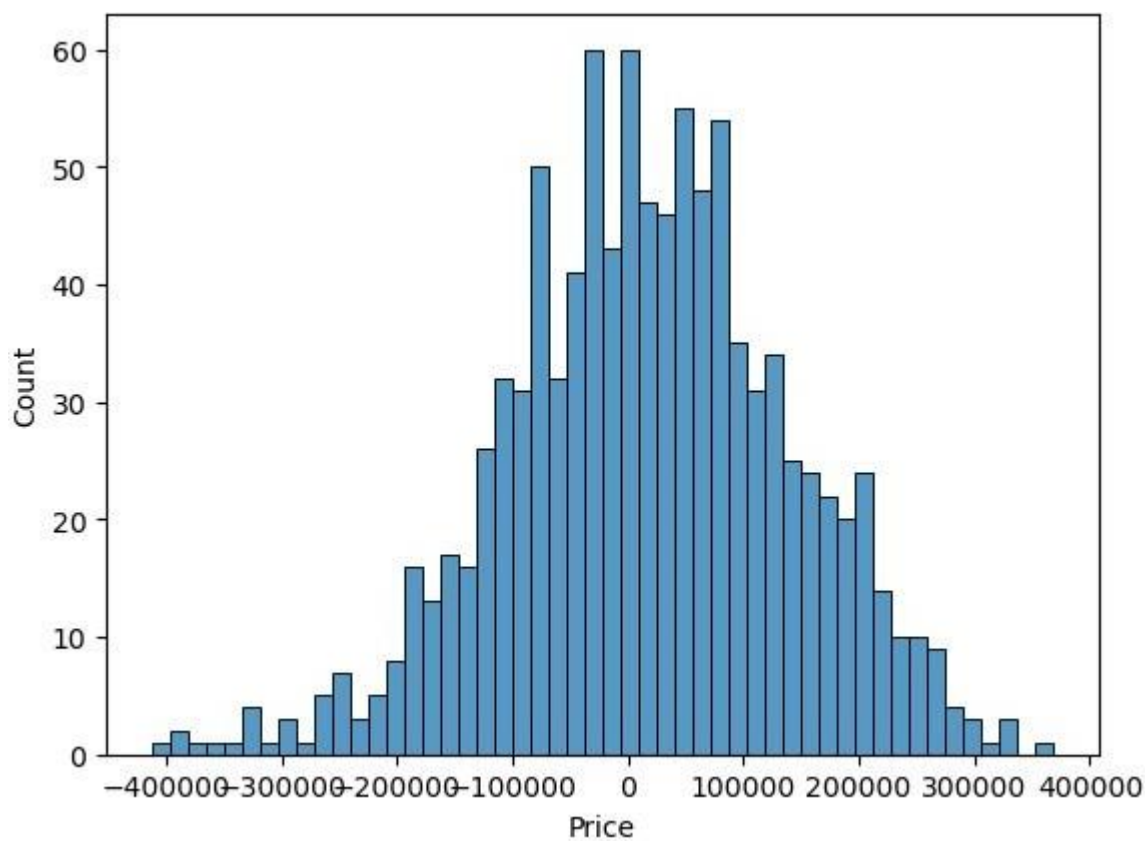



In [29]:

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

Out[29]:

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



In [30]:

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

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

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

Out [30] :

```
-0.0006222175925689744
```

```
286137.81086908665
```

```
128209033251.4034
```

Conclusion and Future Work (Phase 2):

Project Conclusion:

— In the Phase 2 conclusion, we will summarize the key findings and insights from the advanced regression techniques. We will reiterate the impact of these techniques on improving the accuracy and robustness of house price predictions.

— Future Work: We will discuss potential avenues for future work, such as incorporating additional data sources (e.g., real-time economic indicators), exploring deep learning models for prediction, or expanding the project into a web application with more features and interactivity.