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.

- 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 HYPERLINK "http://www.kaggle.com/datasets/vedavyasv/usa-housing)")

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 multicollinearity and overfitting.

- **Lasso Regression:** Employ L1 regularization to perform 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 technique to handle non- linearity and capture complex relationships in the data.

- **Gradient Boosting Regressors (e.g., XGBoost, LightGBM):** Leverage 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 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 features and receive price predictions.

Program:

House Price Prediction

Importing

Dependenci

es 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.filterwarnin

gs("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)
```

In [4]:

valuation 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')
```

Out[4]:

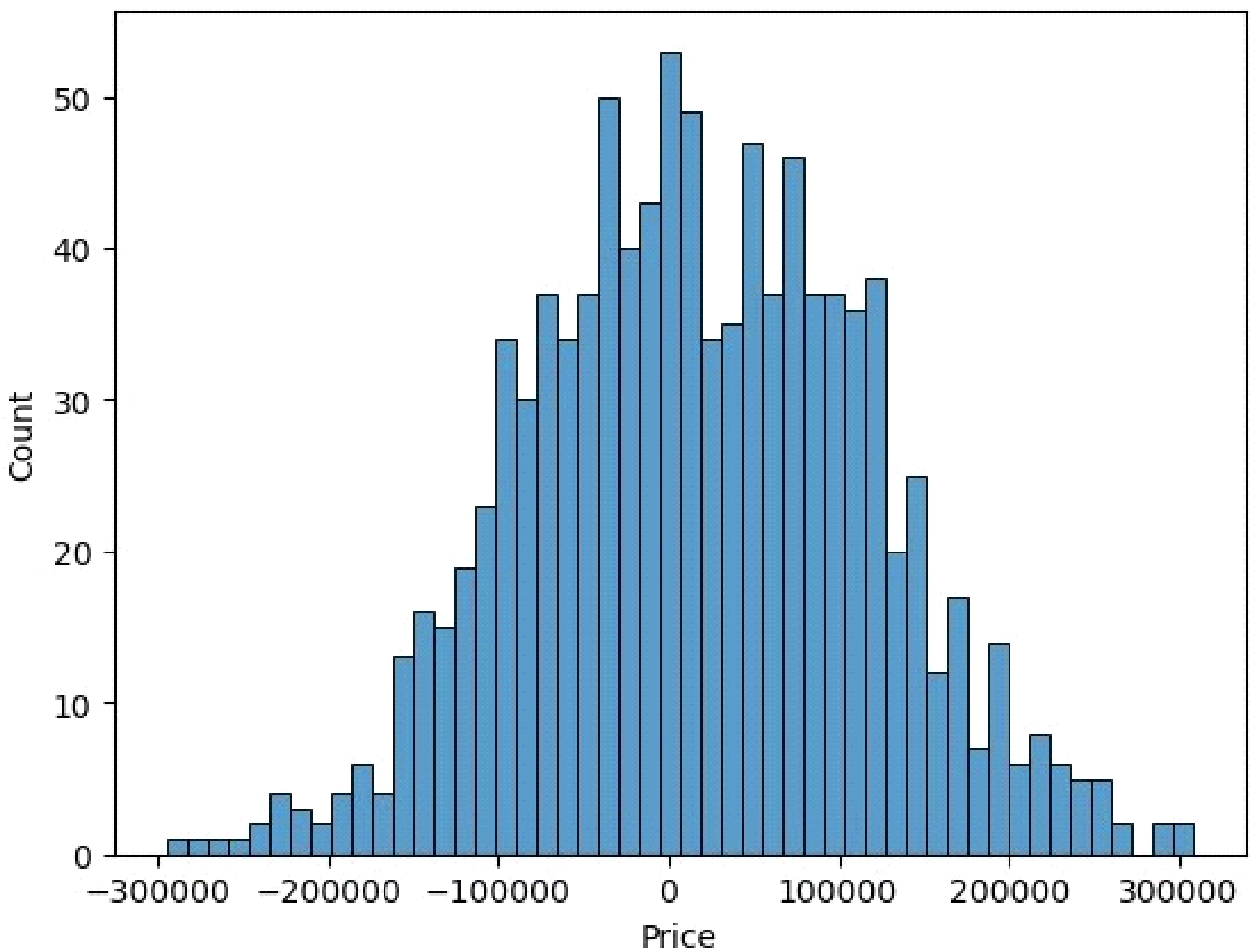
Text(0.5, 1.0, 'Actual vs Predicted')

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]:

```
SVR

SVR()
```

Predicting Prices

In [9]:

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

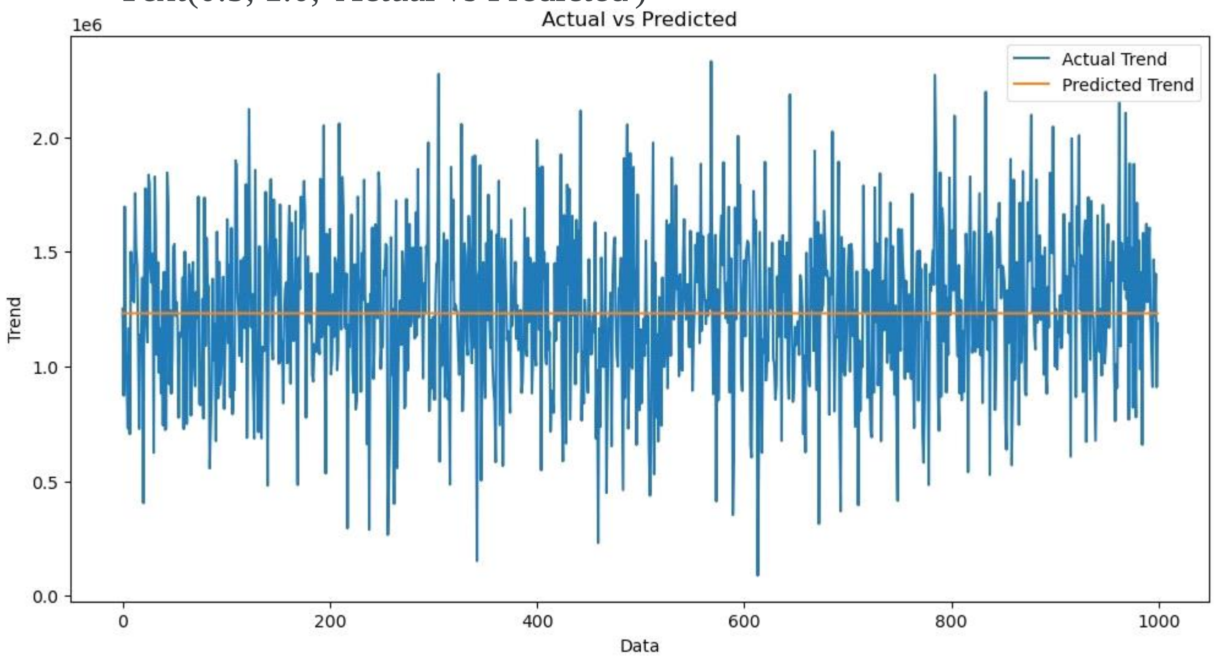
Evaluation of Predicted Data

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
```

<u>Model 3 - Lasso Regression</u>

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)
```

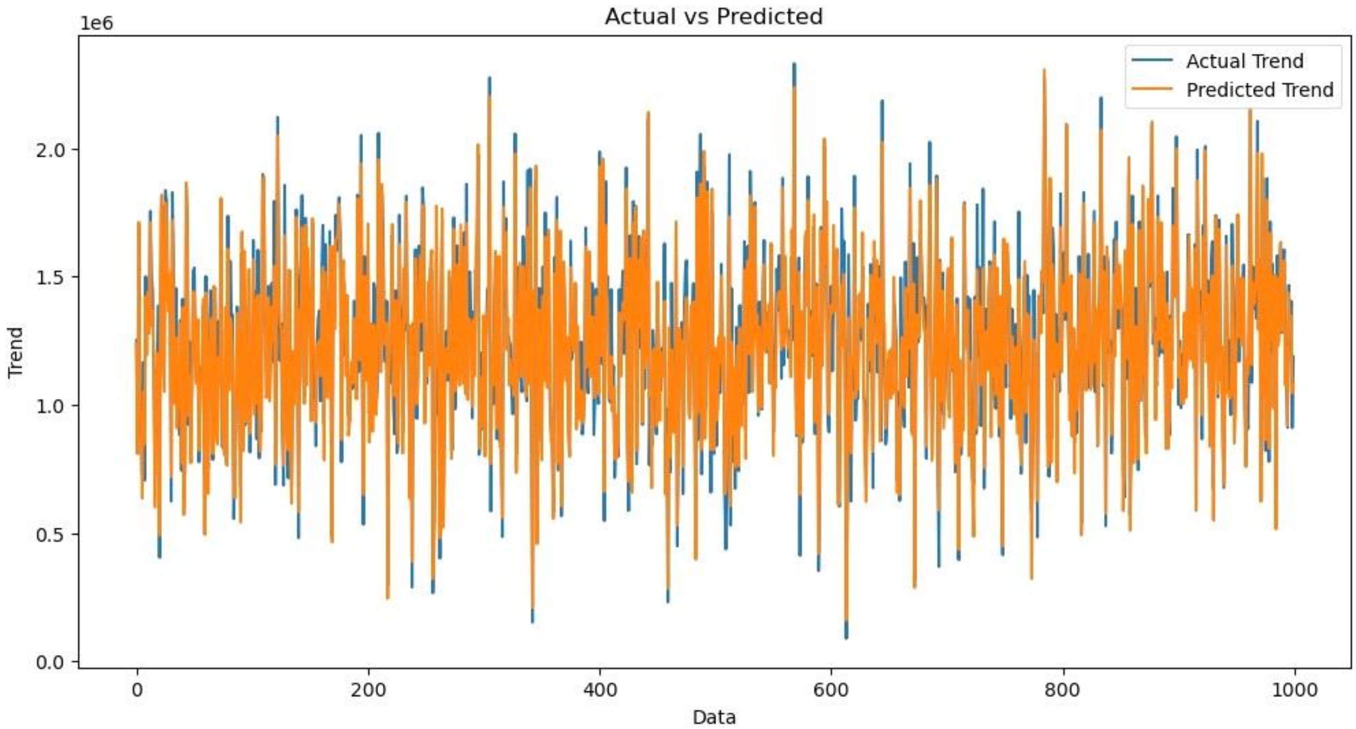
Evaluation of Predicted Data

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')
```

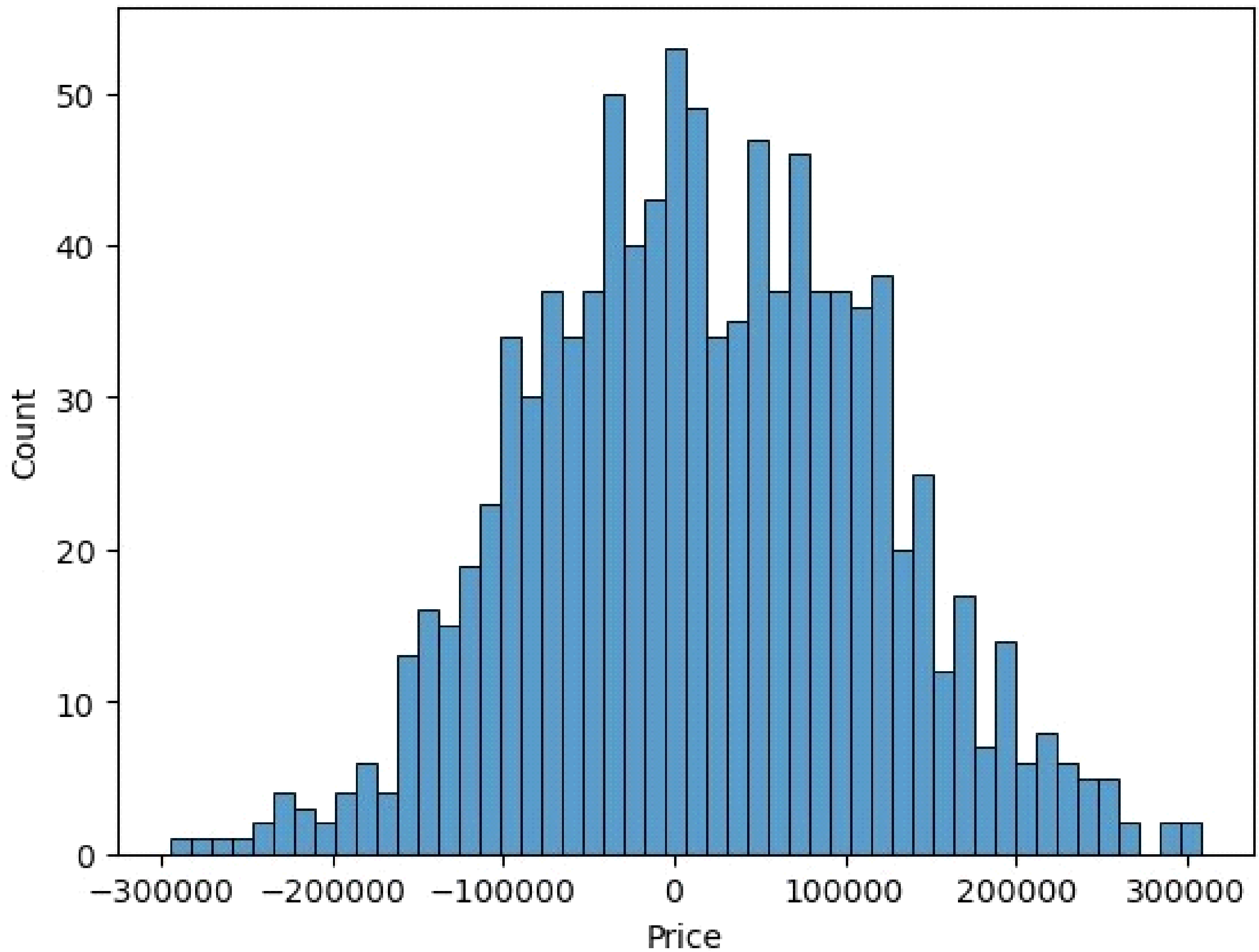


In [17]:

```
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)
```

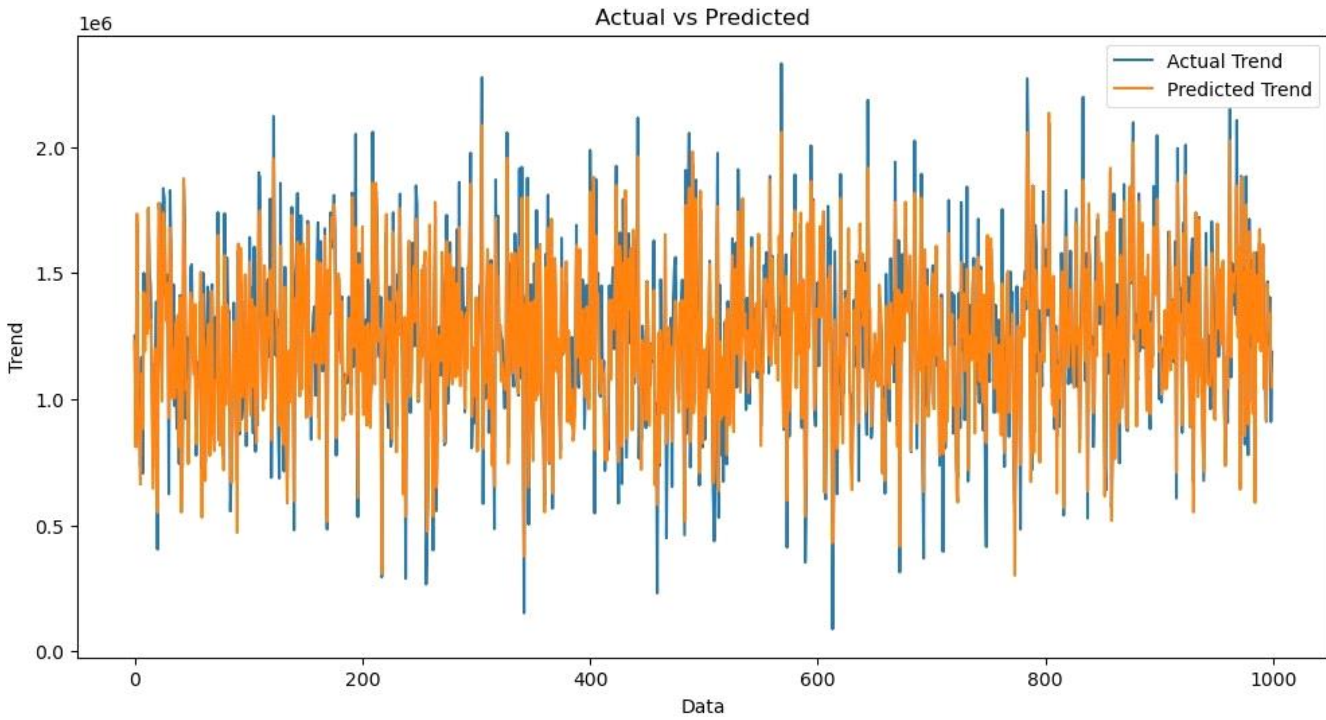
Evaluation of Predicted Data

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

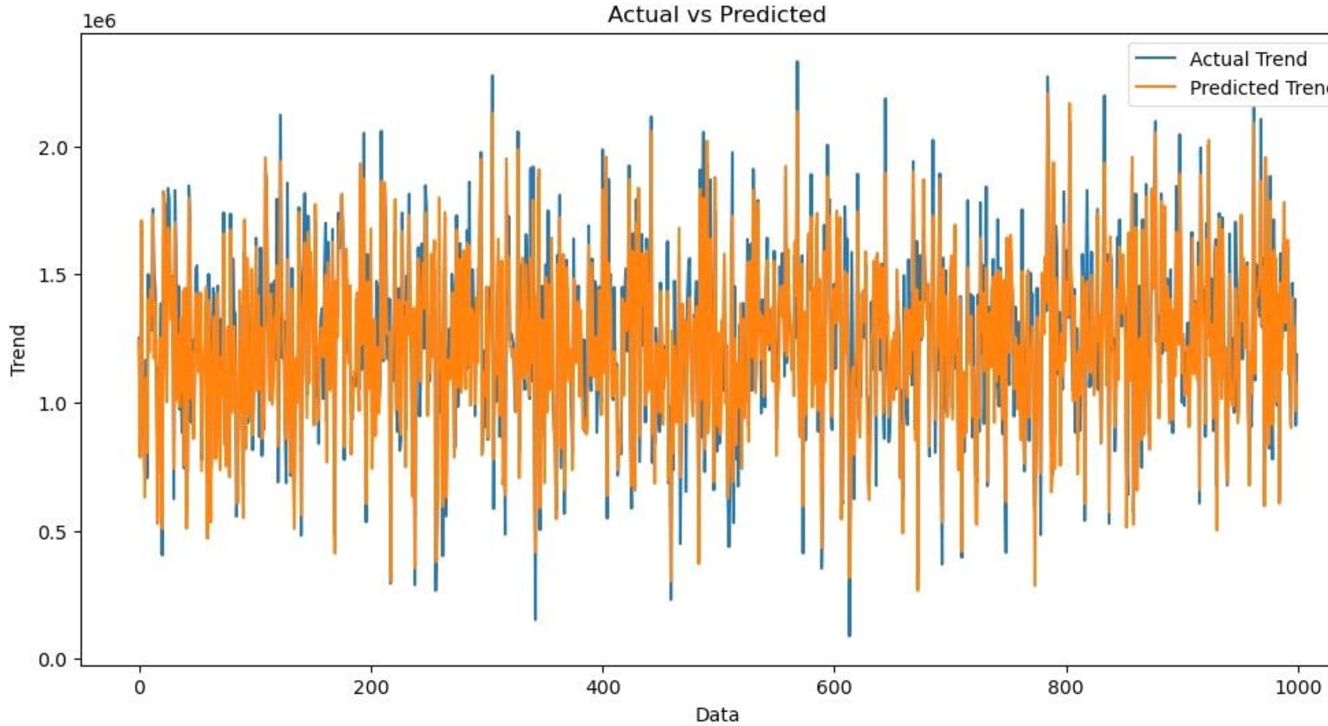
In [28]:

```
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.