# PHASE II

# PREDITING HOUSE PRICES USING ARTIFICIAL INTELLIGENCE

BY M.KANISH PRETHIVE

## INTRODUCTION

- Artificial intelligence (AI) is rapidly transforming many industries, and real estate is no exception. AI-powered house price prediction models are becoming increasingly popular, as they can help buyers, sellers, and investors make more informed decisions.

- These models work by analyzing large datasets of historical home sales data. This data includes a variety of factors, such as the property's location, size, condition, and amenities. The AI model then uses this data to learn the relationships between these factors and home prices.
- Once the model is trained, it can be used to predict the price of a new property. This can be done by simply providing the model with the property's features. The model will then output a prediction of the property's value.
- AI-powered house price prediction models have several advantages over traditional methods. First, they can consider a much larger number of factors than traditional methods. This allows them to produce more accurate predictions. Second, AI models can be updated with new data as it becomes available. This means that their predictions can improve over time.

## Content for Project phase 2

Consider Exploring advanced regression techniques like Gradient Boosting or XGBoost for improved predioction accuracy.

## Data Source

A good data source for house price prediction using machine learning should be accurate, complete, Covering geographic area of interest, Accessible

Dataset link(https://www.kaggle.com/datasets/vedavyasv/usa-housing/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 |

## 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

# 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('/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)
```

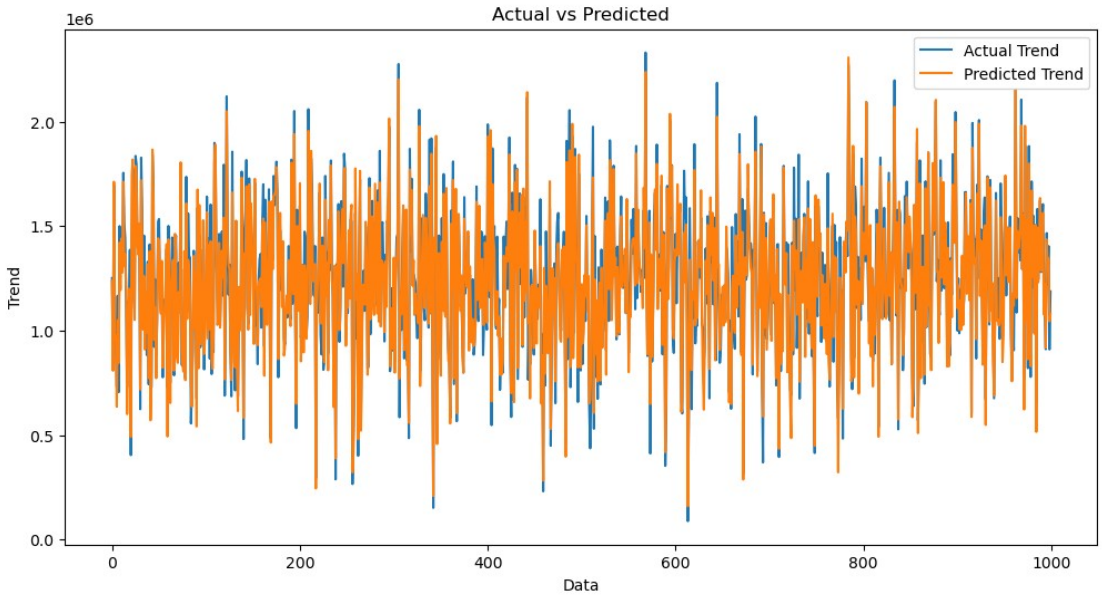
## Evaluation of Predicted Data

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')
```

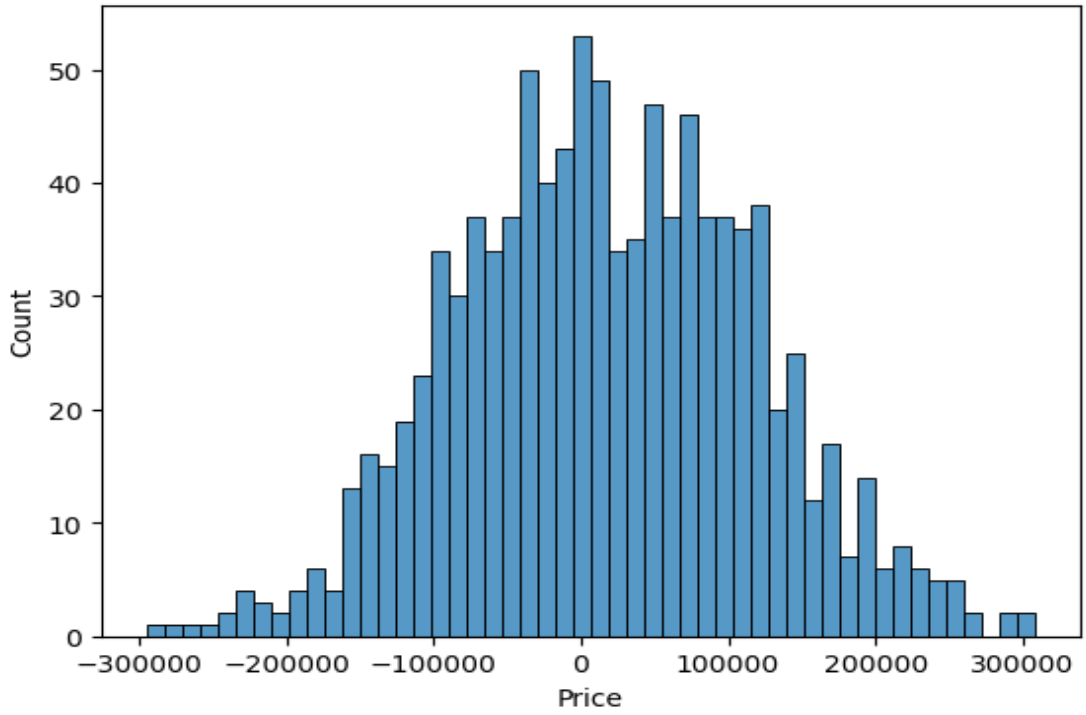



In [26]:

```
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)
```

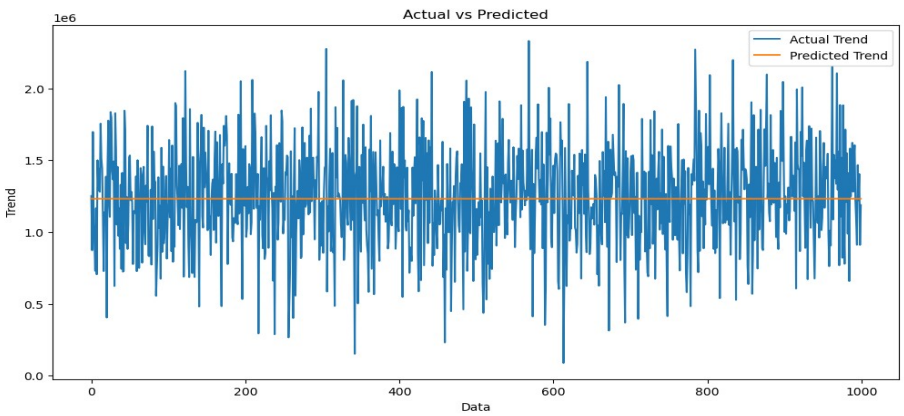
## Evaluation of Predicted Data

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')
```

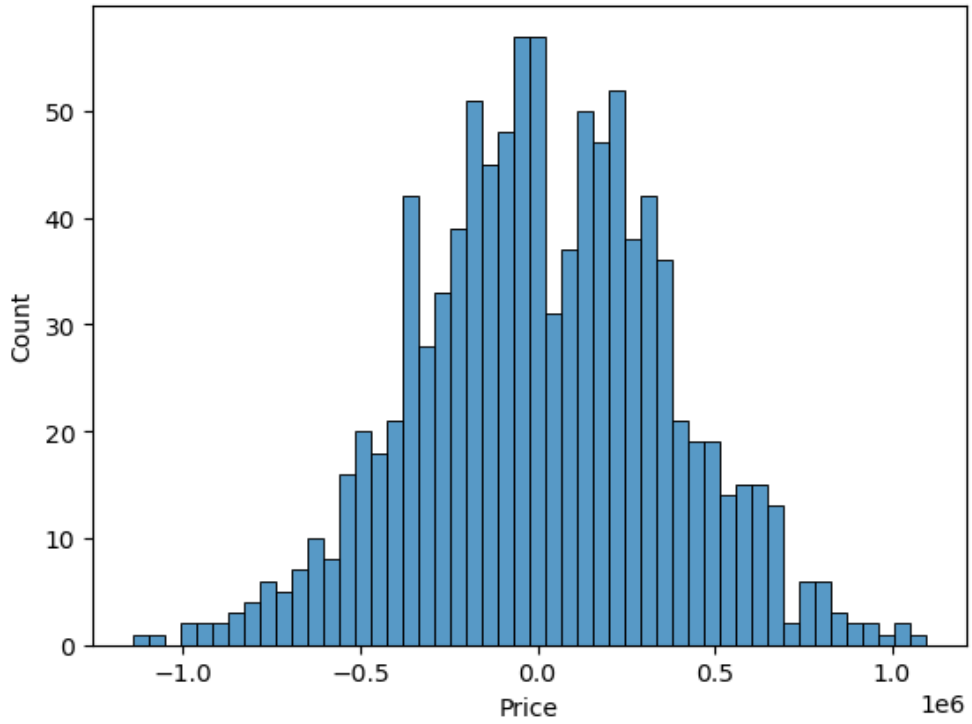


In [32]:

```
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

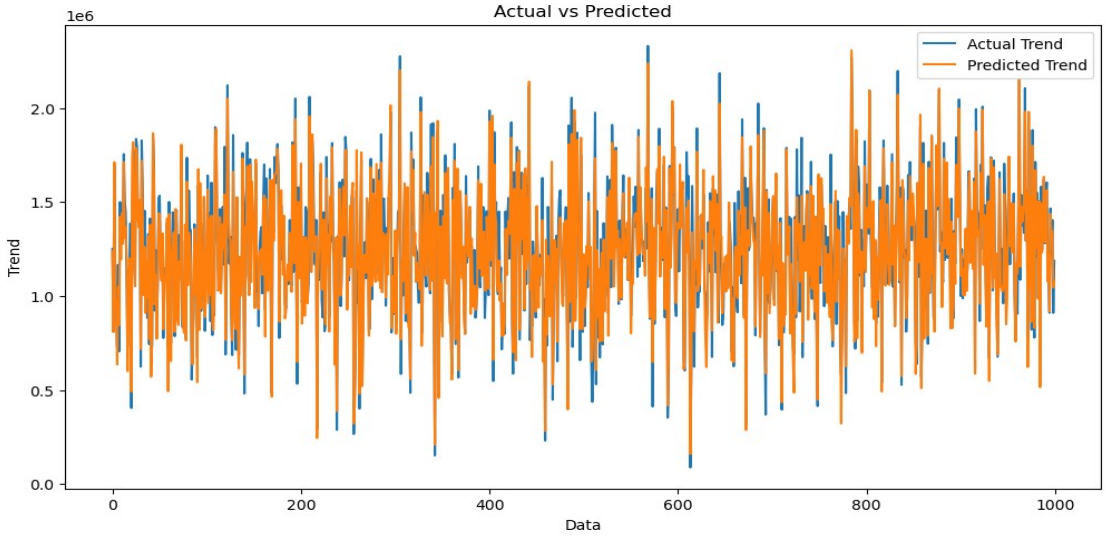
In [36]:

```
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]:

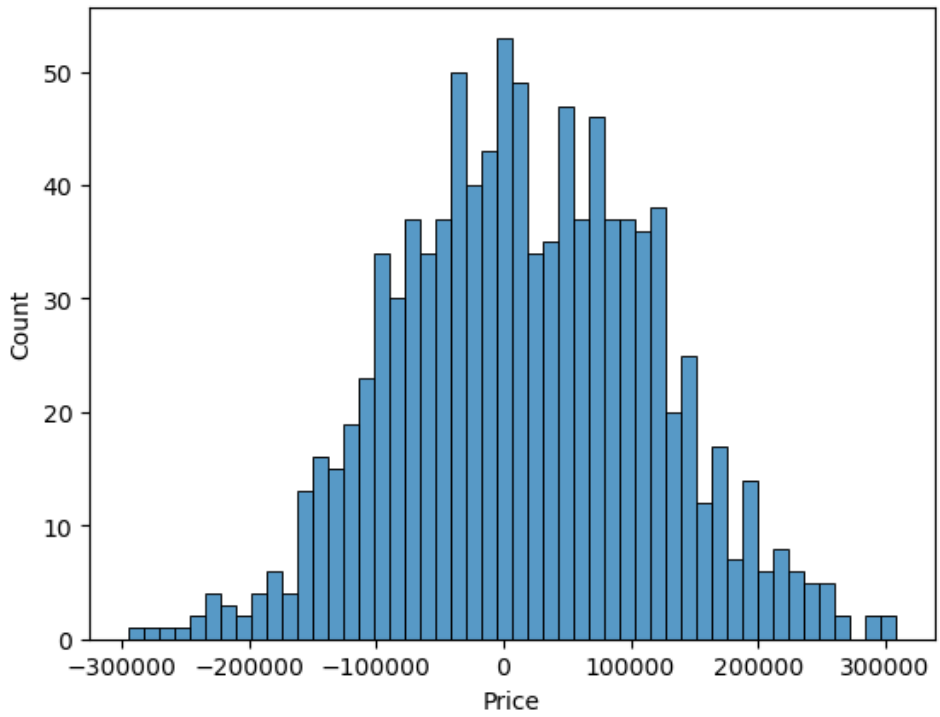
```
Text(0.5, 1.0, 'Actual vs Predicted')
```

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)
```

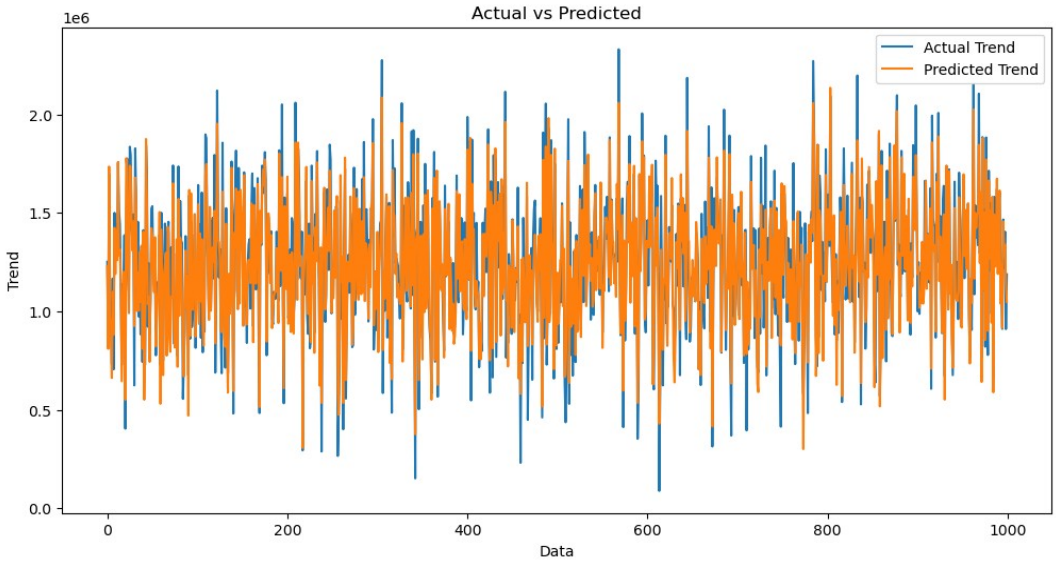
## Evaluation of Predicted Data

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')
```

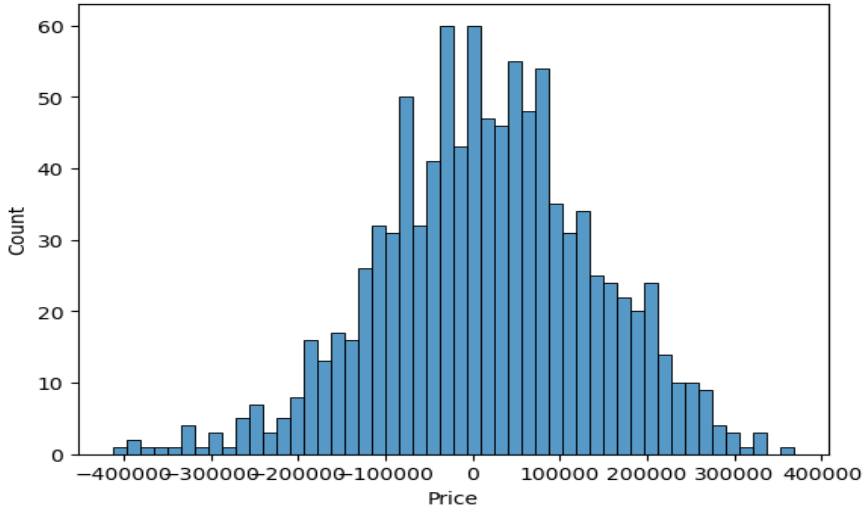



In [44]:

```
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

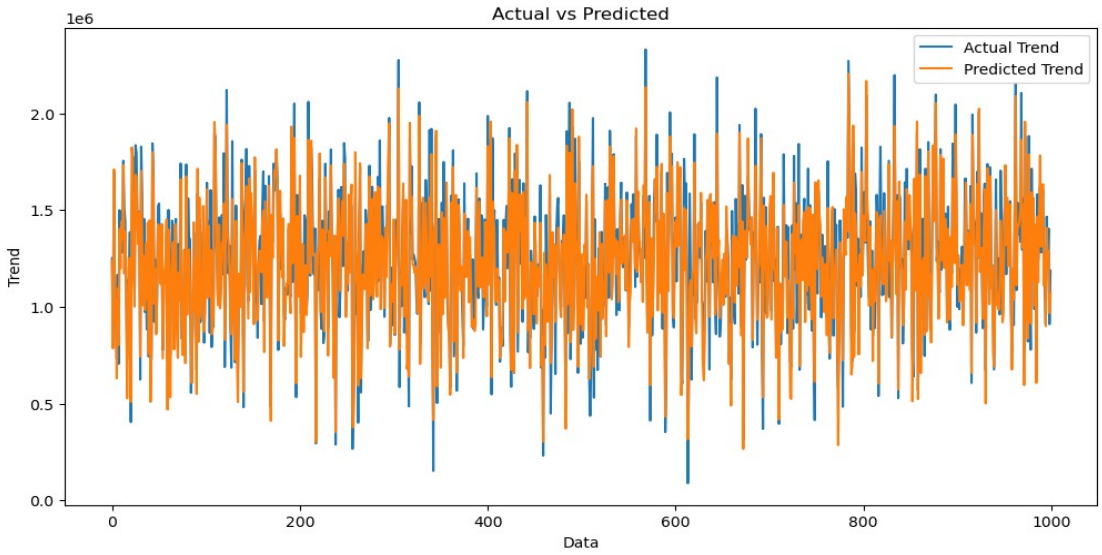
In [49]:

```
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')
```

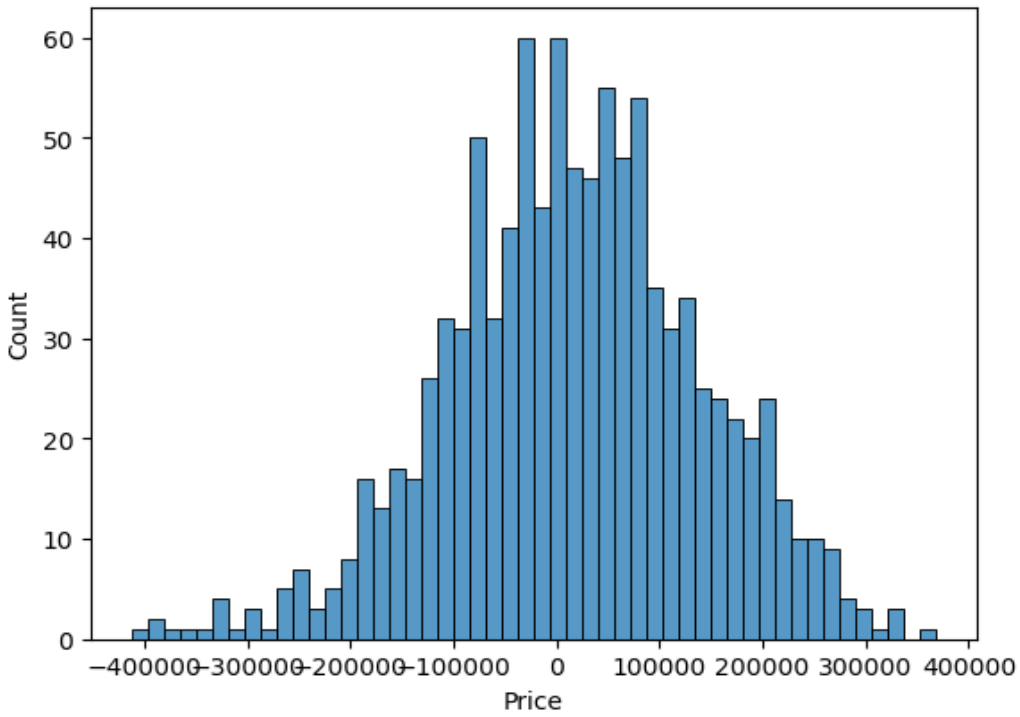



In [50]:

```
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
```