**PREDICTING HOUSE PRICES USING MACHINE LEARNING**

**INTRODUCTION:**

* Predicting house prices using Machine Learning is an application of artificial intelligence and data analysis that aims to forecast the market value of residential properties. This process involves leveraging various algorithms and techniques to analyze a multitude of factors, such as property features (e.g., size, location, number of bedrooms), economic indicators, and historical sales data.
* Machine Learning models are trained on datasets containing information about these factors, allowing them to learn patterns and relationships that influence housing prices. Once trained, these models can make predictions on the value of a house given a set of input features, providing valuable insights to real estate professionals, homebuyers, and sellers.
* The ability to predict house prices accurately Is essential for making informed investment decisions, assisting in property valuation, and understanding market trends. Various techniques, including regression models, decision trees, and deep learning, can be applied to this problem, offering different levels of accuracy and interpretability. Overall, predicting house prices with Machine Learning plays a pivotal role in the real estate industry, facilitating better decision-making and improving market understanding.

**FEATURE SELECTION:**

1. **Understand Your Data:**

Begin by understanding your dataset. Know the features available, their data types, and their potential significance in predicting house prices.

1. **Correlation Analysis:**

Calculate the correlation between each feature and the target variable (house price). Features with higher absolute correlation values are generally more important.

1. **Univariate Feature Selection:**

Use statistical tests like chi-squared or ANOVA to select features with the most significant impact on the target variable.

1. **Recursive Feature Elimination (RFE):**

Train a model and recursively remove the least important features. This method helps you find the optimal subset of features for your model.

1. **Feature Importance from Models:**

Some machine learning algorithms, like decision trees and random forests, provide feature importance scores. You can use these scores to identify important features.

**6. Cross-Validation:**

Perform feature selection within a cross-validation framework to ensure the stability and reliability of your selected features.

**7.Evaluate Model Performance:**

After selecting features, evaluate your machine learning model’s performance using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE). If the performance is not satisfactory, consider revisiting your feature selection choices.

**MODEL TRAINING:**

**1. Data Preprocessing:**

* **Handle missing values:** Decide on a strategy to deal with missing data, such as imputation or removal.
* **Data scaling:** Normalize or standardize numerical features to ensure all features have a similar scale.
* **Categorical data**: Encode categorical variables using techniques like one-hot encoding or label encoding.

**2. Data Splitting:**

* Divide your dataset into training, validation, and test sets. Common splits are 70-30 or 80-20 for training and testing, with a separate validation set for hyper parameter tuning.

**3. Feature Selection:**

* Apply the feature selection techniques discussed in the previous response to choose the most relevant features for your model.

**4. Model Training:**

* Fit the chosen algorithm on the training data using the selected features.
* The algorithm learns the relationships between the input features and the target variable (house prices) during this step.

**5. Model Evaluation:**

* Assess your model’s performance on the validation set using appropriate evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), or R-squared (R2).

**6. Monitoring and Maintenance:**

* Continue to monitor the model’s performance over time and retrain it if necessary to adapt to changing data patterns.

**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.head()

**Output:**

3413 1.305210e+06

1610 1.400961e+06

3459 1.048640e+06

4293 1.231157e+06

1039 1.391233e+06

Name: Price, dtype: float64

Y\_test.shape

**Output:**

(1000,)

**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 1 – Linear Regression**

Model\_lr=LinearRegression()

Model\_lr.fit(X\_train\_scal, Y\_train)

**Output:**

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’)

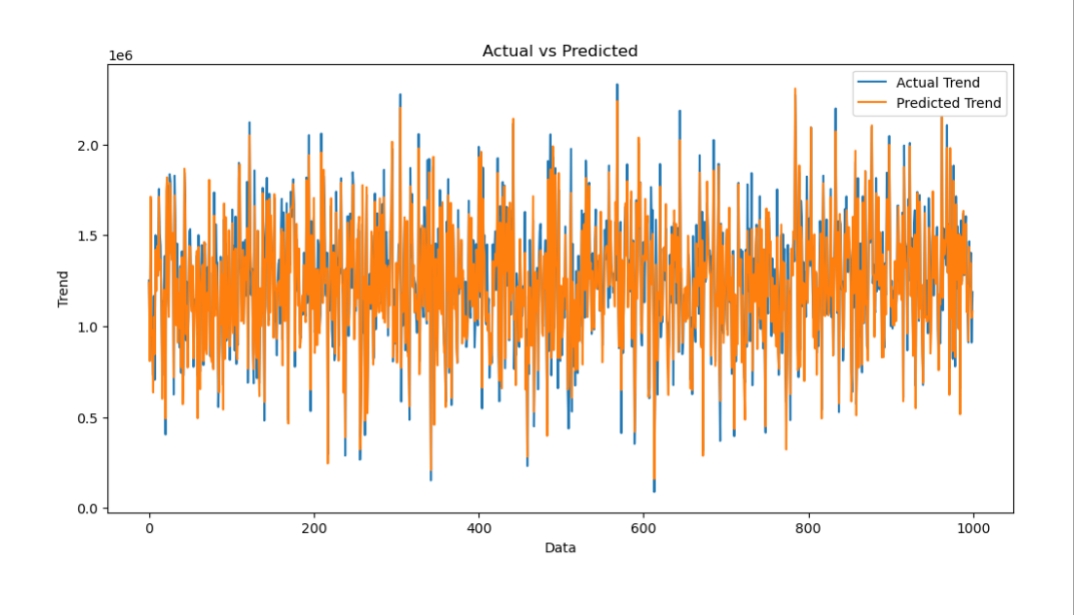
Plt.ylabel(‘Trend’)

Plt.legend()

Plt.title(‘Actual vs Predicted’)

**Output:**

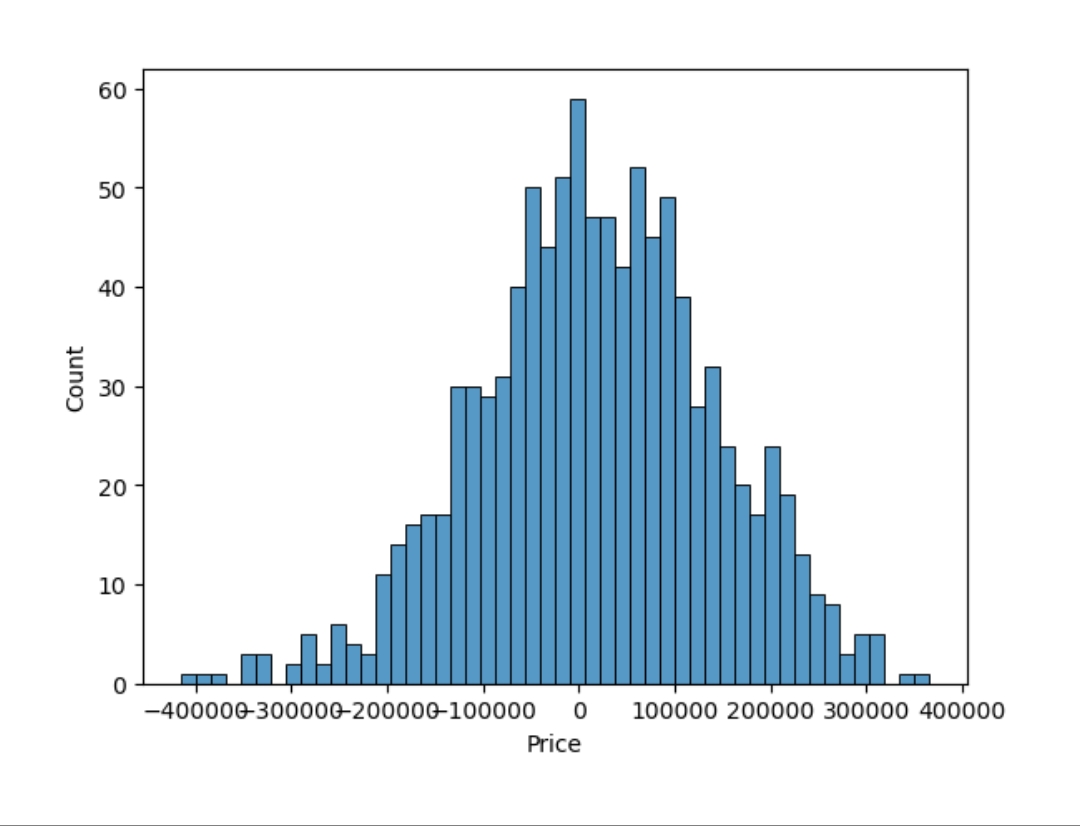
Text(0.5, 1.0, ‘Actual vs Predicted’)



NiSns.histplot((Y\_test-Prediction1), bins=50)

**Output:**

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



Print(r2\_score(Y\_test, Prediction1))

Print(mean\_absolute\_error(Y\_test, Prediction1))

Print(mean\_squared\_error(Y\_test, Prediction1))

**Output:**

0.9182928179392918

82295.49779231755

10469084772.975954

**Model 2 – Support Vector Regressor**

Model\_svr = SVR()

Model\_svr.fit(X\_train\_scal, Y\_train)

**Output:**

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’)

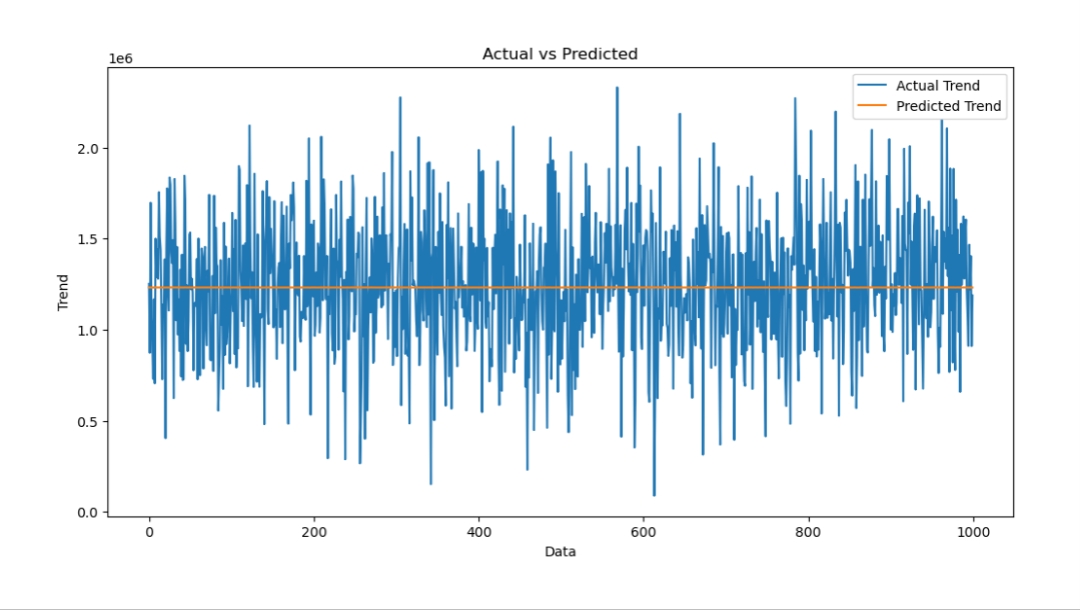
Plt.xlabel(‘Data’)

Plt.ylabel(‘Trend’)

Plt.legend()

Plt.title(‘Actual vs Predicted’)

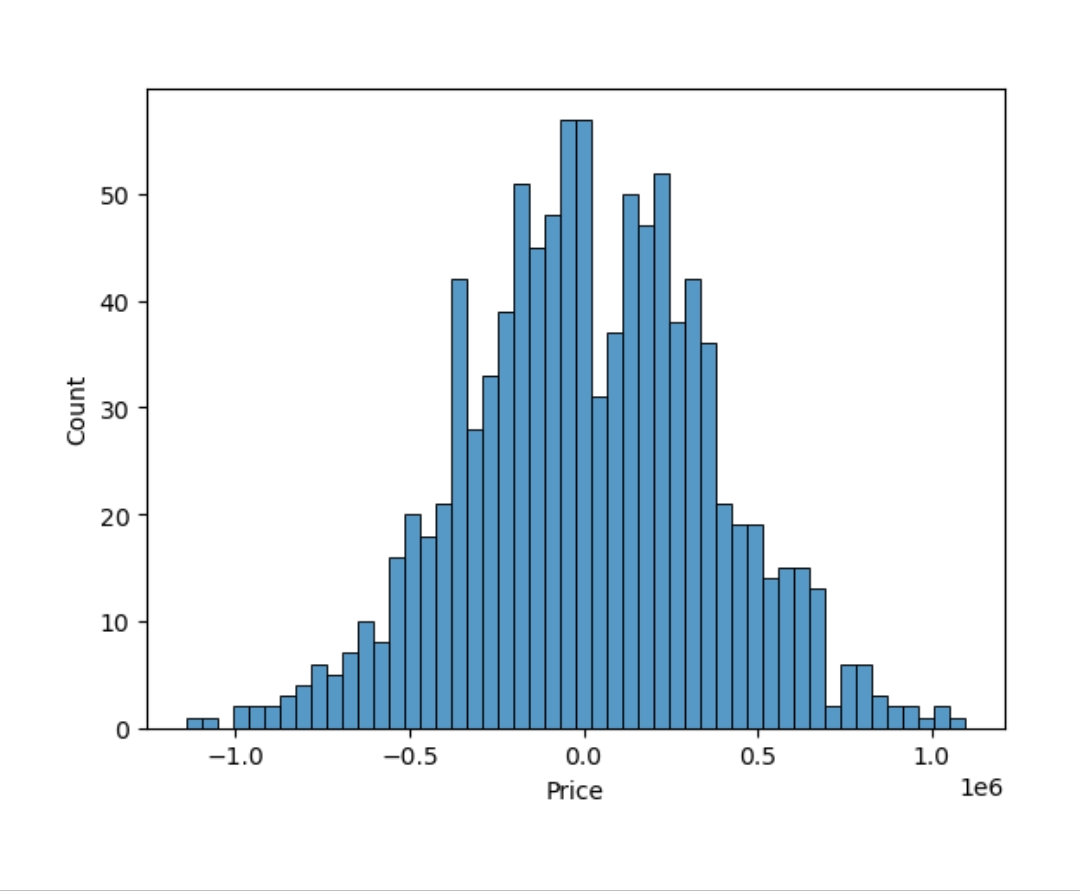
**Output:**

Text(0.5, 1.0, ‘Actual vs Predicted’)

Sns.histplot((Y\_test-Prediction2), bins=50)

**Output:**

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



print(r2\_score(Y\_test, Prediction2))

print(mean\_absolute\_error(Y\_test, Prediction2))

print(mean\_squared\_error(Y\_test, Prediction2))

**Output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

Model 3 – Lasso Regression

Model\_lar = Lasso(alpha=1)

Model\_lar.fit(X\_train\_scal,Y\_train)

**Output:**

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’)

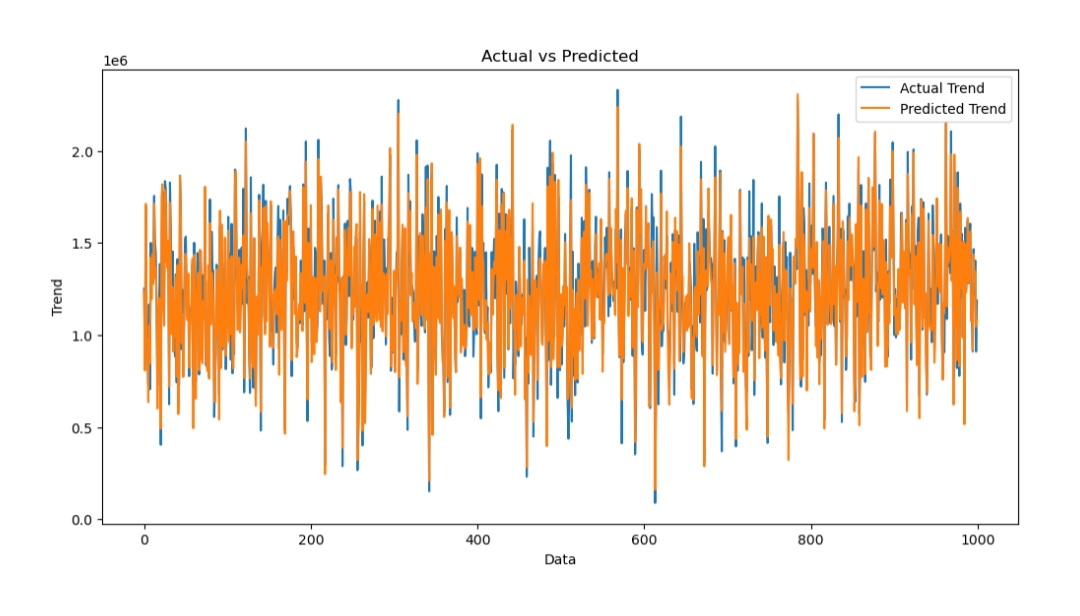
Plt.ylabel(‘Trend’)

Plt.legend()

Plt.title(‘Actual vs Predicted’)

**Output:**

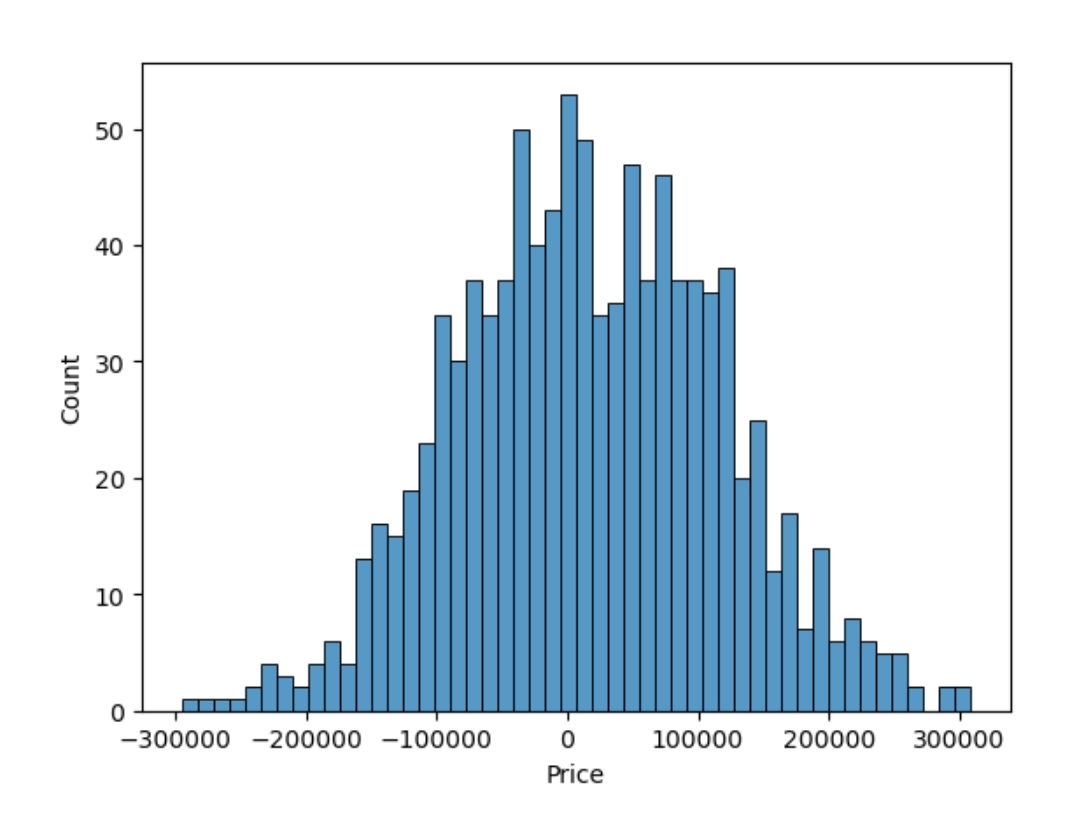
Text(0.5, 1.0, ‘Actual vs Predicted’)



Sns.histplot((Y\_test-Prediction3), bins=50)

**Output:**

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



Print(r2\_score(Y\_test, Prediction2))

Print(mean\_absolute\_error(Y\_test, Prediction2))

Print(mean\_squared\_error(Y\_test, Prediction2))

**Output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

**Model 4 – Random Forest Regressor**

Model\_rf = RandomForestRegressor(n\_estimators=50)

Model\_rf.fit(X\_train\_scal, Y\_train)

**Output:**

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’)

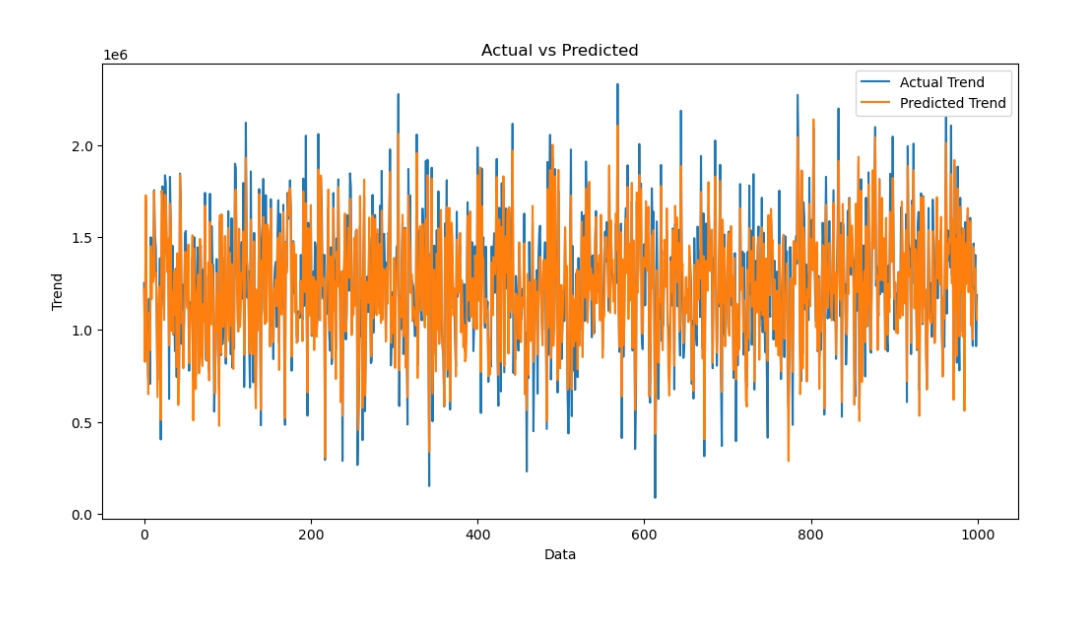
Plt.ylabel(‘Trend’)

Plt.legend()

Plt.title(‘Actual vs Predicted’)

**Output:**

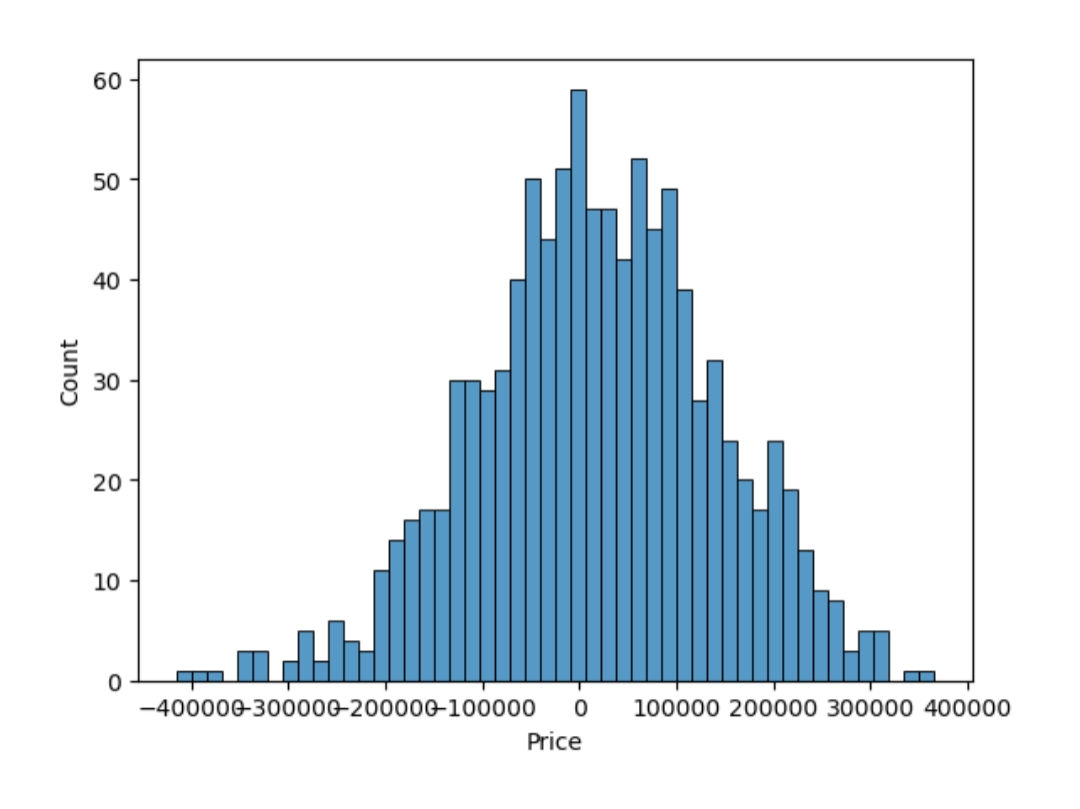
Text(0.5, 1.0, ‘Actual vs Predicted’)



Sns.histplot((Y\_test-Prediction4), bins=50)

**Output:**

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



Print(r2\_score(Y\_test, Prediction2))

Print(mean\_absolute\_error(Y\_test, Prediction2))

Print(mean\_squared\_error(Y\_test, Prediction2))

**Output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

**Model 5 – Xgboost Regressor**

Model\_xg = xg.XGBRegressor()

Model\_xg.fit(X\_train\_scal, Y\_train)

**Output:**

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**

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’)

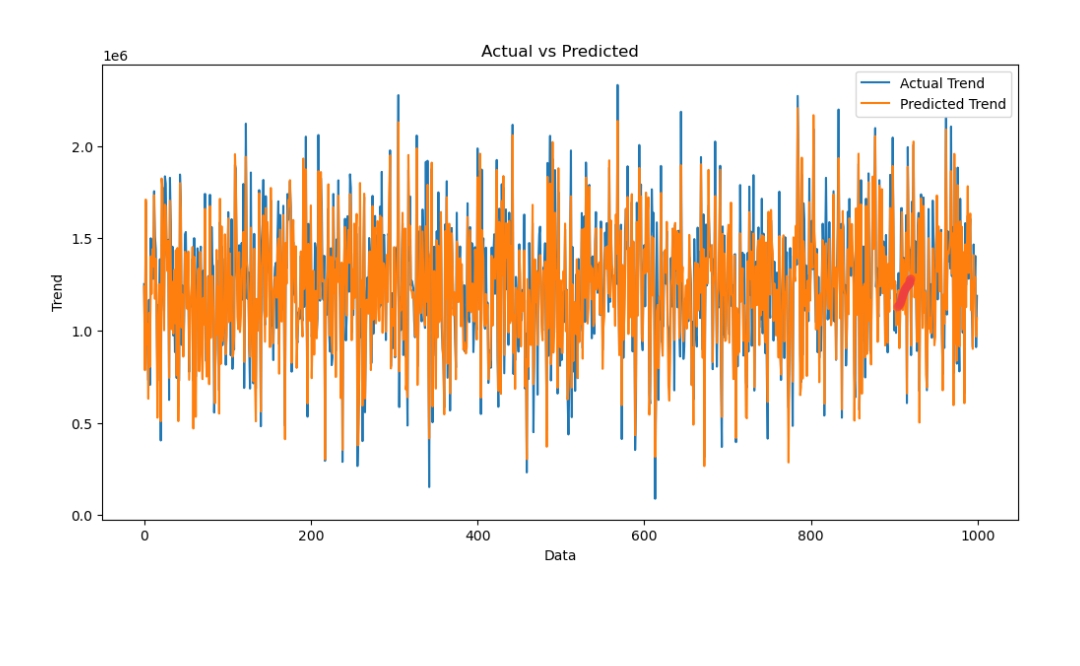
Plt.ylabel(‘Trend’)

Plt.legend()

Plt.title(‘Actual vs Predicted’)

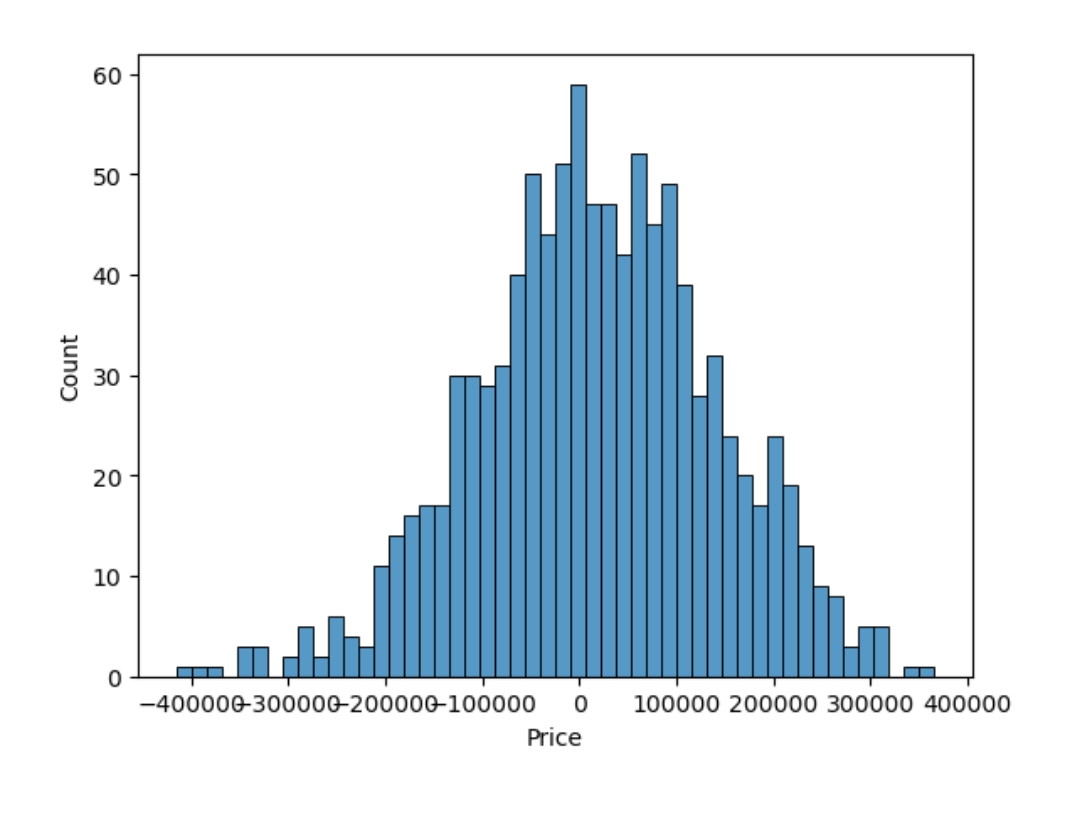
**Output:**

Text(0.5, 1.0, ‘Actual vs Predicted’)



Sns.histplot((Y\_test-Prediction4), bins=50)

**Output:**

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

Print(r2\_score(Y\_test, Prediction2))

Print(mean\_absolute\_error(Y\_test, Prediction2))

Print(mean\_squared\_error(Y\_test, Prediction2))

**Output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

### Linear Regression is giving us best Accuracy.

**CONCLUSION:**

* In the end, the success of predicting house prices using machine learning relies on a combination of careful feature selection, effective model training, and rigorous evaluation.
* It’s important to strike a balance between model complexity and performance, adapting to the specific dataset and problem requirements.
* Continuous monitoring and maintenance are often necessary to ensure that the model remains accurate as data distributions change over time.