**TITLE : Predicting House Prices using Machine Learning**

**Phase 4: Development Part 2**

**Continue building the house price prediction model by feature selection, model training and evaluation.**

Certainly! Continuing with the development of a house price prediction model, let’s proceed with Phase 4: Development Part 2, which includes feature selection, model training, and evaluation:

1.Feature Selection:

- Begin by assessing the relevance of each feature in your dataset. Consider using methods like correlation analysis, feature importance scores, or domain knowledge to select the most important features.

- Remove irrelevant or redundant features to simplify the model.

2. Data Preprocessing:

- Handle missing values by imputing or removing them based on the nature of the data.

- Encode categorical variables using techniques like one-hot encoding or label encoding.

3. Split Data:

- Split your dataset into a training set and a testing set. The common split ratio is 80% for training and 20% for testing.

4. Model Selection:

- Choose a regression algorithm suitable for your problem. Common choices include Linear Regression, Random Forest, XGBoost, or Neural Networks.

5. Model Training:

- Train the selected model on the training data.

6. Model Evaluation:

- Evaluate the model’s performance on the testing data using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2).

- Utilize cross-validation techniques to assess the model’s robustness.

7. Hyperparameter Tuning:

- If needed, fine-tune the model’s hyperparameters to improve its performance. You can use techniques like grid search or random search for this.

8. Model Interpretation:

- Interpret the model’s coefficients or feature importances to understand which factors have the most significant impact on house prices.

9. Visualization:

- Create visualizations, such as scatter plots or regression plots, to gain insights into the model’s predictions and actual values.

10. Model Deployment:

- If you plan to deploy the model for real-world use, set up a deployment pipeline and integrate it into your application or system.

Remember to iterate and refine the model as needed to achieve the best performance. It’s also important to keep track of the entire development process to ensure reproducibility and maintainability.

**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()

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\_train.shape

(4000,)

Y\_test.head()

1718 1.251689e+06

2511 8.730483e+05

345 1.696978e+06

2521 1.063964e+06

54 9.487883e+05

Name: Price, dtype: float64

Y\_test.shape

(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**

\_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')

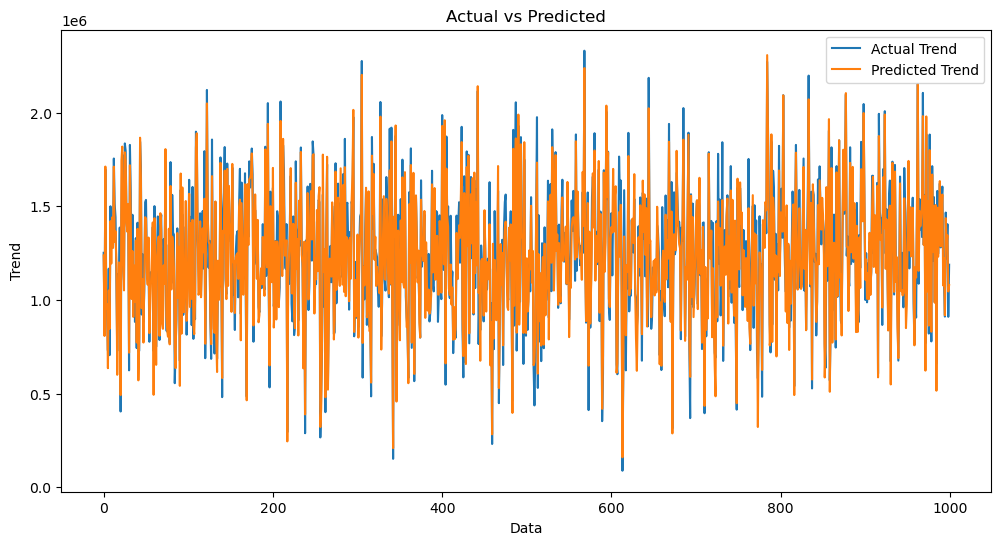
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

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

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

<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))

0.9182928179392918

82295.49779231755

10469084772.975954

**Model 2- 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')

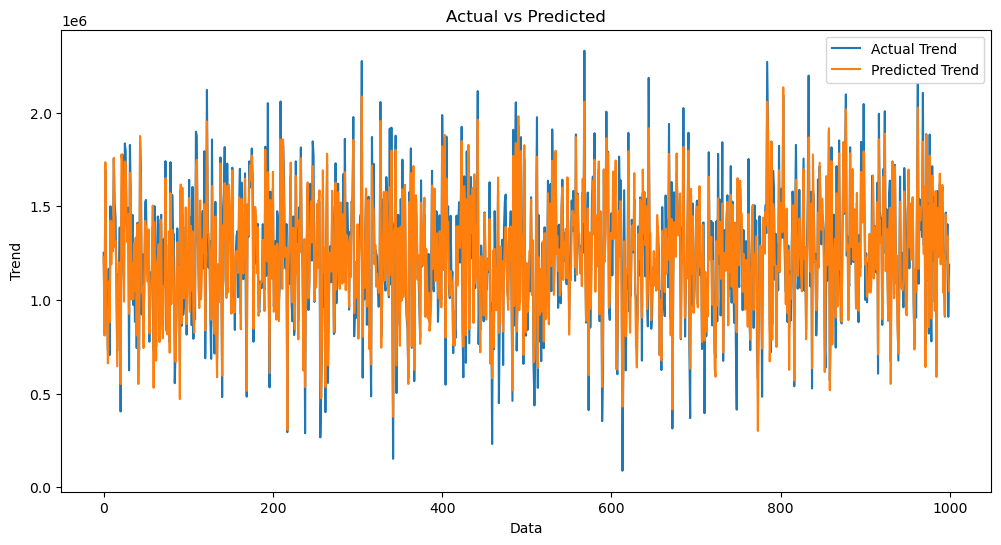
plt.xlabel('Data')

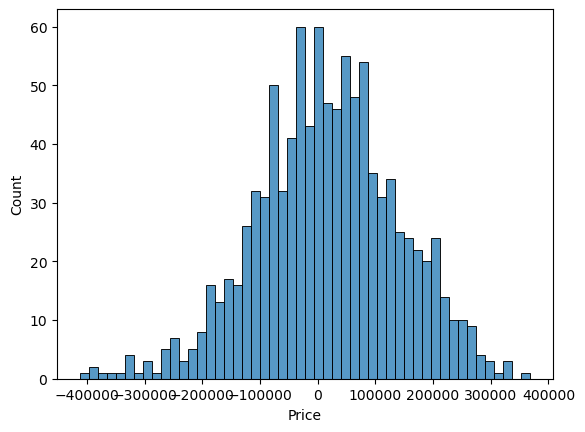
plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

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

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

<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))

-0.0006222175925689744

286137.81086908665

128209033251.4034

**Model 3:**

**XGboost Regressor**

model\_xg = xg.XGBRegressor()

model\_xg.fit(X\_train\_scal, Y\_trai

**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')

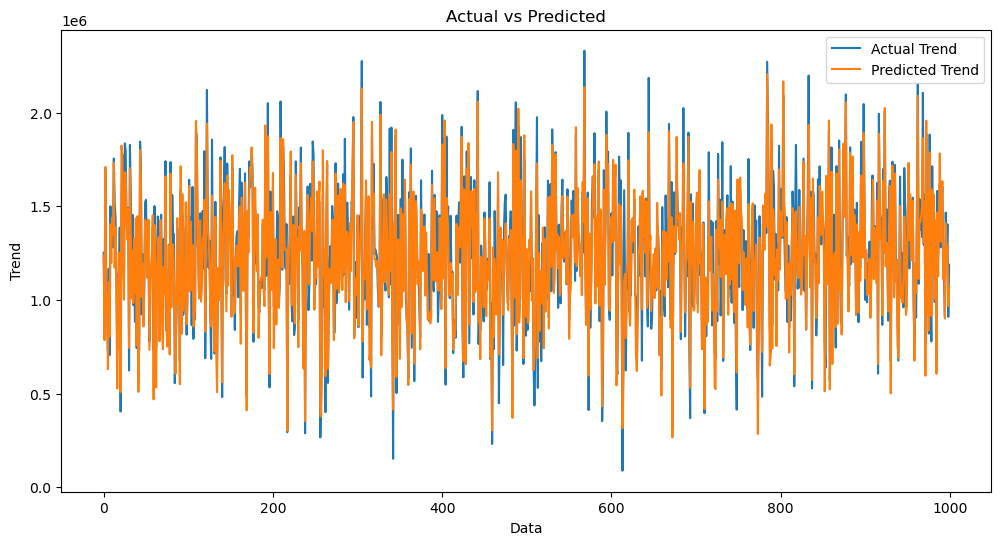
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

plt.title('Actual vs Predicted')

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

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

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

