Predicting House Prices using Machine Learning

# INTRODUCTION

House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house. There are three factors that influence the price of a house which include physical conditions, concept and location.In the case of house price prediction, we can use historical data on various features of a house, such as its location, size, and amenities, to train a machine-learning model. Once the model is trained, it can analyze new data on a given house and make a prediction of its market value

In this phase loading and preprocessing of house price prediction is going to be done.

# DATASET

The data is obtained from <https://www.kaggle.com/datasets/vedavyasv/usa-housing>

# COLUMNS USED

##### 

##### From USA\_Housing.csv data the following columns are used

* **Avg. Area Income**
* **Avg. Area House Age**
* **Avg. Area Number of Rooms**
* **Avg. Area Number of Bedrooms**
* **Area Population**
* **Price**
* **Address**

# LIBRARIES USED

The Python 3 environment comes with many helpful analytics libraries installed and several helpful packages to load.

The essential libraries used in this project are :

* Importing OS (for kaggle inputs)
* Numpy and Pandas libraries
* Matplotlib
* seaborn

**ALGORITHM USED**

* Here we are going use some algorithms which is used for predicting house prices , they are :

1.Linear regression.

2.support vector regressor.

3.Random Forest Regressor.

4.XGboost Regressor

## **Model 1 - Linear Regression**

model\_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')

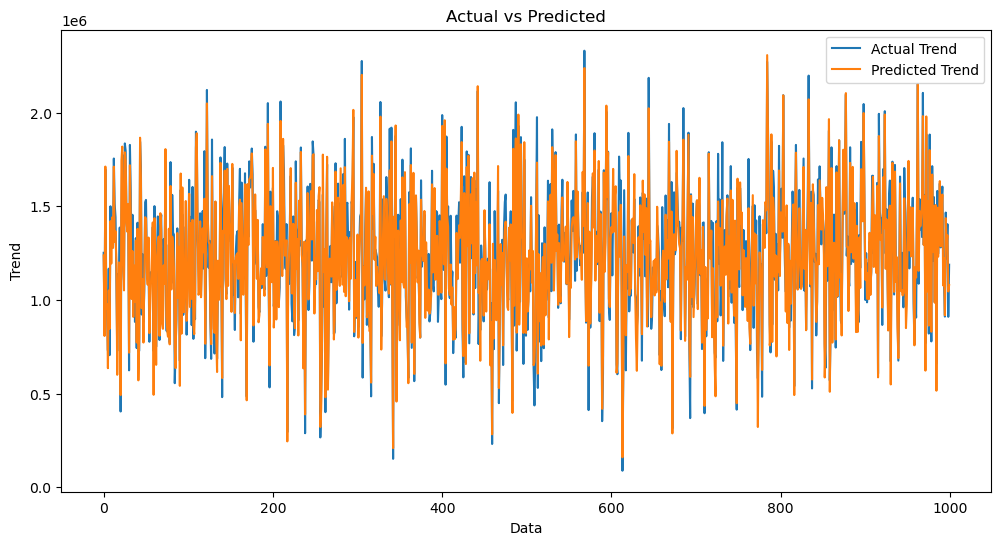
plt.xlabel('Data')

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 - Support Vector Regressor**

model\_svr = SVR()

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

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')

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



sns.histplot((Y\_test-Prediction2), 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 - 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')

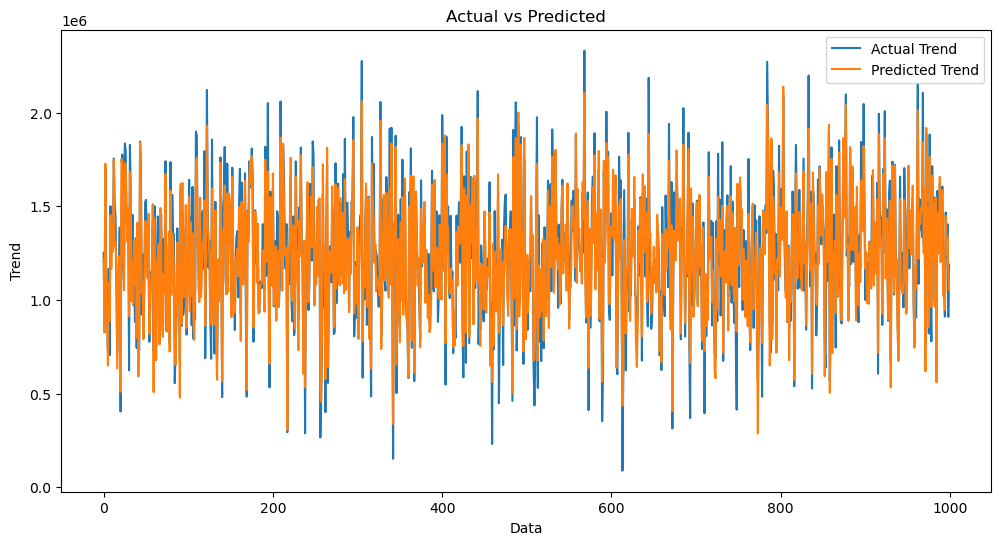
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

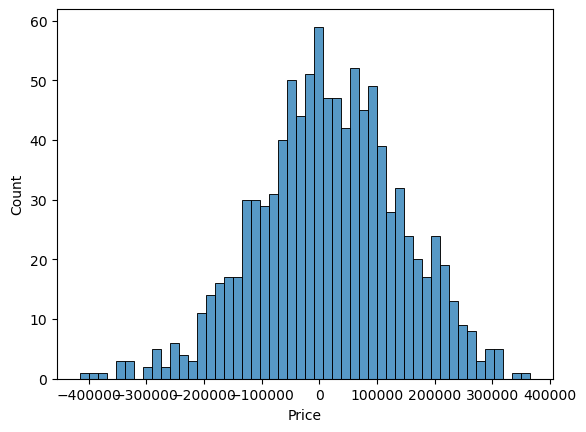
plt.title('Actual vs Predicted')

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 4 - XGboost Regressor**

model\_xg = xg.XGBRegressor()

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

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')

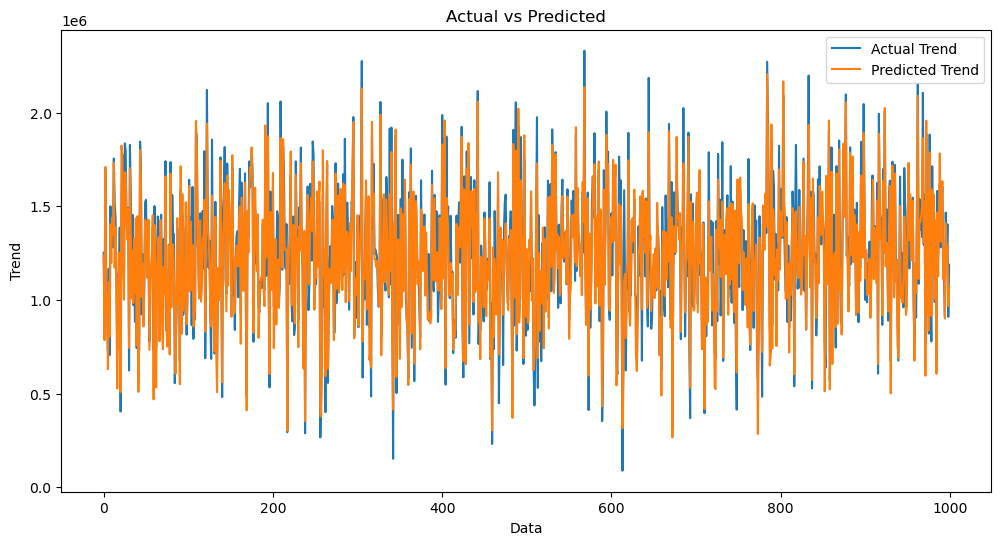
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

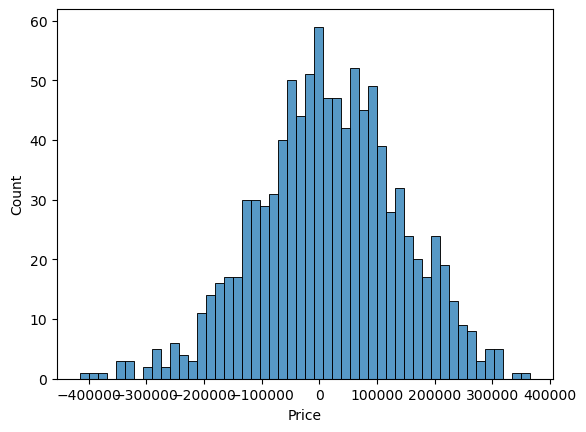
plt.title('Actual vs Predicted')

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



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

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



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

### **Linear Regression is giving us best Accuracy.**