PREDICTING HOUSE PRICE USING MACHINE LEARNING

REG NO: 950621104089
NAME: SAVARI JEEVA JEBASTIN P

PHASE 2: INNOVATION

Date	11 October 2023
Team ID	proj_212169_Team_3
Project Name	HOUSE PRICE PREDICITON USING MACHINE LEARNING

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, andeconomic 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.

CONTENT FOR PROJECT PHASE 2:

- Consider exploring advanced regression techniques like Gradient Boosting or XGBOOST forimproved 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)

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, andcategorical 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.

ADVANCED REGRESSION TECHNIQUES:

- **RIDGE REGRESSION :** Introduce L2 regularization to mitigate multicollinearity and over fitting.
- **LASSO REGRESSION:** Employ L1 regularization to perform feature selection and simplify the model.
- **ELASTIC NET REGRESSION:** Combine both L1 and L2 regularization to benefit from their respective advantages.
- **RANDOM FORSET REGRESSION:** Implement an ensemble technique to handle non-linearity and capture complex relationships in the data.
- **GRADIENT BOOSTING REGRESSOR (e.g., XGBOOST):** Utilize gradient boosting algorithms for improved accuracy.

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")
```

LOADING DATASET

```
dataset = pd.read_csv('E:/USA_Housing.csv')
```

MODEL1: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)
```

EVALUATION OF PREDICTING DATA:

In [4]:

```
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]:

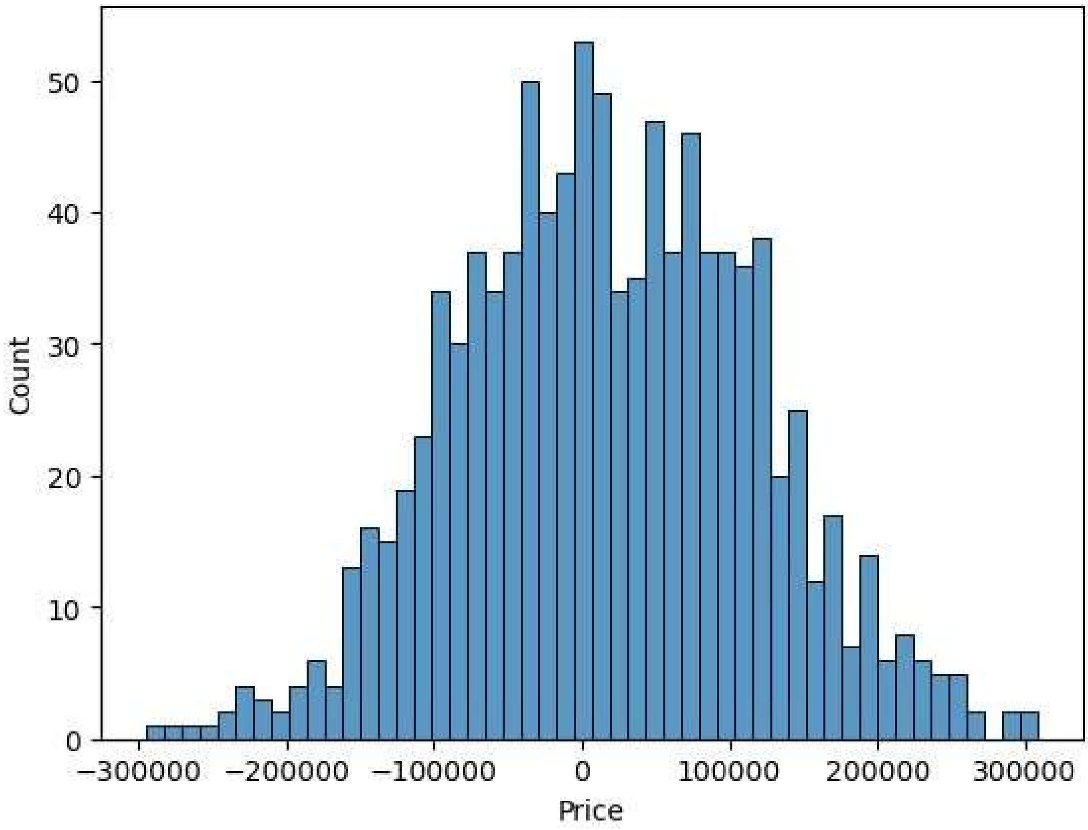
```
Text(0.5, 1.0, 'Actual vs Predicted')
```

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

MODEL2: SUPPORT VECTOR REGRESSOR:

In [7]:

```
model_svr = SVR()
```

In [8]:

```
model_svr.fit(X_train_scal, Y_train)
```

Out[8]:

```
▾ SVR
SVR()
```

In [9]:

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

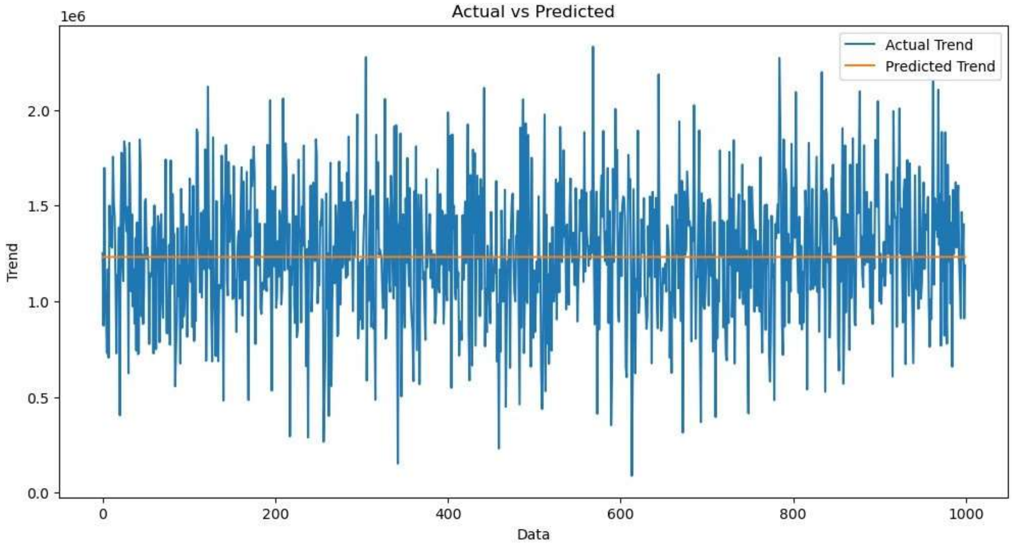
EVALUATION OF PREDICTING DATA

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

MODEL3:

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)
```

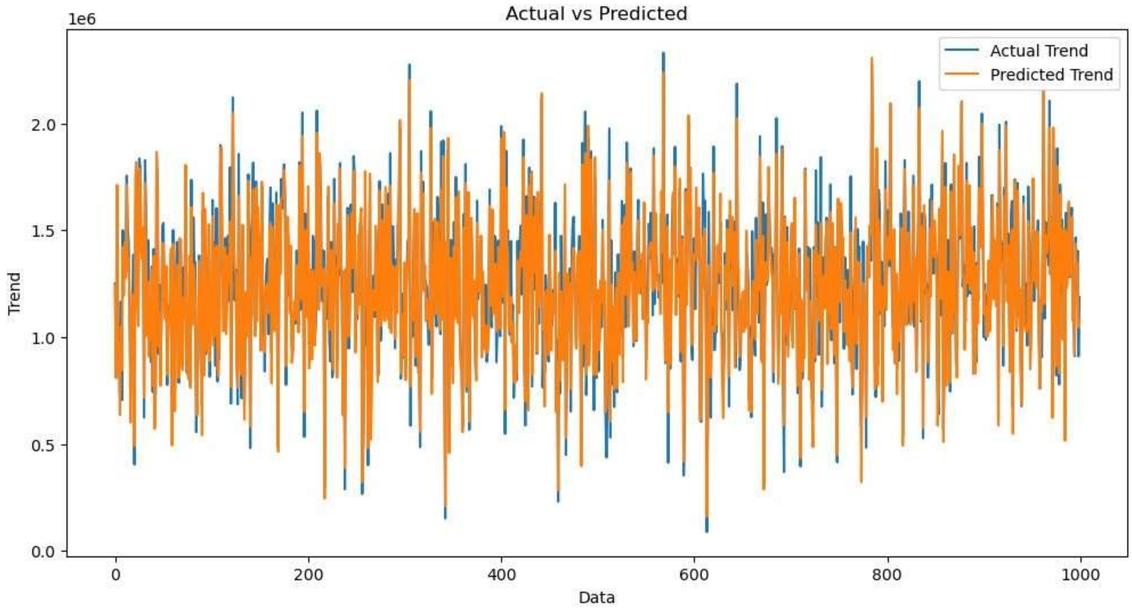
EVALUATION OF PREDICTING DATA

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')

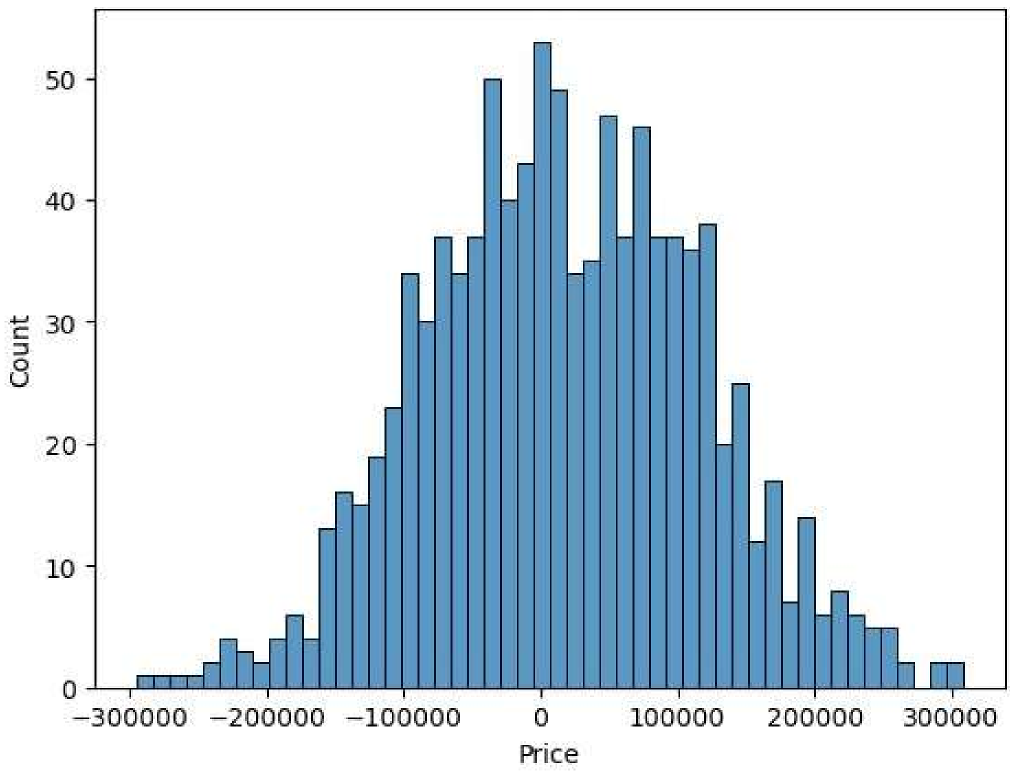


In [17]:

```
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

MODEL4: 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)
```

In [21]:

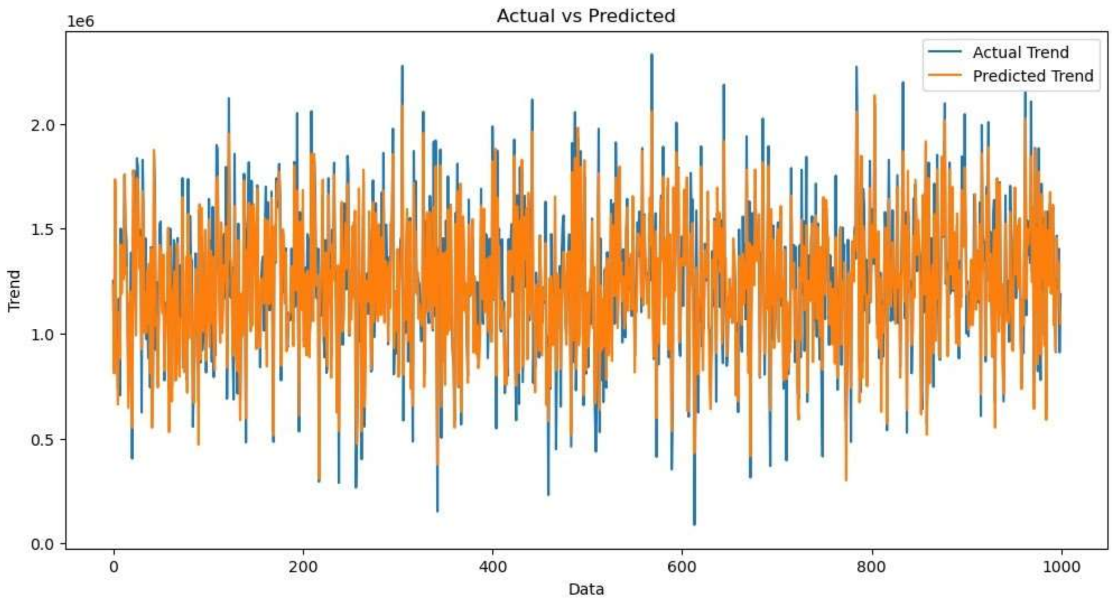
```
Prediction4 = model_rf.predict(X_test_scal)
```

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

In [25]:

```
model_xg = xg.XGBRegressor()
```

In [26]:

```
model_xg.fit(X_train_scal, Y_train)
```

Out[26]:

```
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, ...)
```

In [27]:

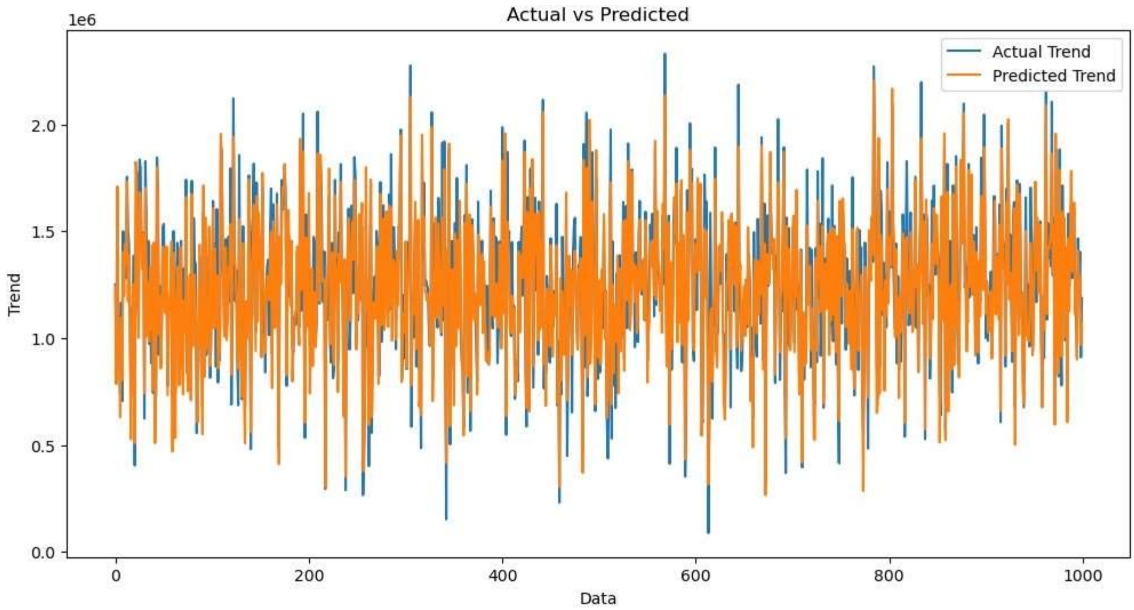
```
Prediction5 = model_xg.predict(X_test_scal)
```

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

- 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.