4222 -SURYA GROUP OF INSTITUTION

VIKRAVANDI -605 652

SUBJECT CODE-SB3001

COURSE NAME-EXPERIENCE BASED PRACTICAL LEARNING

PREDICTION HOUSE PRICES USING MACHINE LEARNING

NAAN MUDHALVAN PROJECT

PREPARED BY

N.PERUMAL

REG NO :422221106012

ECE DEPARTMENT

HOUSE PRICE PREDICTION USING MACHINE LEARNING

INTRODUCTION:

House price prediction using machine learning is a common and well-established task in the field of data science and real estate. You can use various machine learning algorithms to predict house prices based on historical data and a set of relevant features. Here's a step-by-step guide on how to approach this task:

Data Collection:

Gather a dataset that includes historical information about houses, such as the number of bedrooms, bathrooms, square footage, location, age, and, most importantly, the actual selling prices.
Data Preprocessing:

Handle missing data: Identify and fill in missing values in the dataset.
Encode categorical variables: Convert categorical features like "location" into numerical values using techniques like one-hot encoding or label encoding.
Feature scaling: Normalize or standardize numerical features to ensure they are on a similar scale.
Feature Selection/Engineering:

Select relevant features that are likely to influence the house price. This may involve domain knowledge and statistical analysis.
Create new features if necessary, such as the price per square foot or the age of the house at the time of sale.
Split Data:

Divide the dataset into two parts: a training set and a testing set. A common split is 70-80% for training and the remainder for testing.
Choose a Machine Learning Algorithm:

Select a regression algorithm suitable for predicting house prices. Common choices include Linear Regression, Decision Trees, Random Forest, Support Vector Machines, and Gradient Boosting algorithms like XGBoost or LightGBM.
Train the Model:

Use the training data to train the chosen algorithm. The algorithm will learn the relationships between the features and the house prices in the training dataset.
Evaluate the Model:

Use the testing dataset to evaluate the model's performance. Common regression metrics for evaluation include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R^2).
Hyperparameter Tuning:

Fine-tune the model's hyperparameters to optimize its performance. This can be done using techniques like grid search or random search.
Make Predictions:

Once you are satisfied with the model's performance, you can use it to make predictions on new, unseen data.
Deployment:

If you plan to make your model available for others to use, you can deploy it as a web application or API.
Regular Maintenance:

Keep the model updated with new data if you want it to provide accurate predictions over time.
PROBLEM DEFINITION :

OBJECTIVE:

- People looking to buy a new home tend to be more conservative with their budgets and market strategies.
- This project aims to analyze various parameters like average income, average area etc. and predict the house price accordingly.
- This application will help customers to invest in an estate without approaching an agent
- To provide a better and fast way of performing operations. • To provide proper house price to the customers.
- To eliminate need of real estate agent to gain information regarding house prices.
- To provide best price to user without getting cheated.
- To enable user to search home as per the budget.
- The aim is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analyzing previous market trends and price ranges, and also upcoming developments future prices will be predicted.
- House prices increase every year, so there is a need for a system to predict house prices in the future.
- 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.
- We use Random forest regression algorithm in machine learning for predicting the house price trends

DATA SOURCE : Collection of data processing techniques and processes are numerous. We collected data for USA real estate properties from various real estate websites. The data would be having attributes such as Location, carpet area, built-up area, age of the property, zip code, price, no of bed rooms etc. We must collect the quantitative data which is structured and categorized. Data collection is needed before any kind of machine learning research is carried out. Dataset validity is a must otherwise there is no point in analyzing the data.

Data preprocessing :

Data preprocessing is the process of cleaning our data set. There might be missing values or outliers in the dataset. These can be handled by data cleaning. If there are many missing values in a variable we will drop those values or substitute it with the average value. `dataset = pd.read_csv('/kaggle/input/usa-housing/USA_Housing.csv')` `dataset.info()` `dataset.describe()`

```
dataset.columns
```

VISUALISATION AND PRE PROCESSING DATA

```
sns.histplot(dataset, x='Price', bins=50, color='y')
sns.boxplot(dataset, x='Price', palette='Blues') sns.jointplot(dataset,
x='Avg. Area House Age', y='Price', kind='hex') sns.jointplot(dataset,
x='Avg. Area Income', y='Price')
    plt.figure(figsize=(12,8))
sns.pairplot(dataset)

dataset.hist(figsize=(10,8))
dataset.corr(numeric_only=True
plt.figure(figsize=(10,5))
    sns.heatmap(dataset.corr(numeric_only = True), annot=True)
```

DIVIDE THE DATA SET

```
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()

Y_train.shape
Y_test.head()

Y_test.shape
 sc =
StandardScaler()
X_train_scal =
sc.fit_transform(X_tr
ain)
X_test_scal = sc.fit_transform(X_test)
```

MODEL BULIDING AND EVALUTION OF PREDICATED DATA

```
    model_lr=LinearRegression()

odel_lr.fit(X_train_scal, Y_train)

   Prediction1 = model_lr.predict(X_test_scal)

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')     ns.histplot((Y_test-Prediction1),
bins=50)

  print(r2_score(Y_test,

Prediction2))

 print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))


print(r2_score(Y_test, Prediction1)) print(mean_absolute_error(Y_test,
Prediction1)) print(mean_squared_error(Y_test, Prediction1))



 Model_rf = RandomForestRegressor(n_estimators=50)
```

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

Consider exploring advanced regression techniques like XG Boost for Improved prediction accuracy.

PROPOSED SYSTEM:

In this proposed system, we focus on predicting the house price values using machine learning

algorithms like XG Boost regression model. We proposed the system "House price prediction using Machine Learning" we have predicted the House price using XG boost regression model. In this proposed system, we were able to train the machine from the various attributes of data points from the past to make a future prediction. We took data from the previous year stocks to train the model .The data set we used was from the official organization. Some of data was used to train the machine and the rest some data is used to test the data. The basic approach of the supervised learning model is to learn the patterns and relationships in the data from the training set and then reproduce them for the test data. We used the python pandas library for data processing which combined different datasets into a data frame. The raw data makes us to prepare the data for feature identification. The attributes were stories, no. of bed rooms, bath rooms, Availability of garage, swimming pool, fire place, year built, area in soft, sale price for a particular house. We used all these features to train the machine on XG boost regression and predicted the house price, which is the price for a given day. We also quantified the accuracy by using the predictions for the test set and the actual values. The proposed system gives the Predicted price.

ALOGORITHM:

We used the python pandas library for data processing which combined different datasets into a data frame. The raw data makes us to prepare the data for feature identification. XG for regression builds an additive model in a forward stage wise fashion. It allows for the optimization of arbitrary differentiable loss functions. In each stage, a regression tree is fit on the negative XG of the given loss function. The idea of boosting came out of the idea of whether a weak learner can be modified to become better. A weak hypothesis is defined as one whose performance is at least slightly better than random chance. The Objective is to minimize the loss of the model by adding weak hypothesis using a XG descent like procedure. This class of algorithms was described as a stage-wise additive model. This is because one new weak learner is added at a time and existing weak learners in the model are frozen and left unchanged.

Step 1: Load the data set df = pd.read csv("ml_house_data_set.csv")

Step 2: Replace categorical data with one-hot encoded data

Step 3: Remove the sale price from the feature data

Step 4: Create the features and labels X and Y arrays.

Step 5: Split the data set in a training set (70%) and a test set (30%). Step 6: Fit regression model.

Step 7: Save the trained model to a file trained_house_classifier_model.pkl

Step 8: Predict house worth using predict function

MODEL XG BOOST REGRESSOR :

INPUT:

```
# Importing the libraries
from sklearn.metrics import confusion_matrix import
xgboost as xgb
from sklearn.model_selection import train_test_split from
sklearn.preprocessing import LabelEncoder, OneHotEncoder
import numpy as np import matplotlib.pyplot as plt import pandas
as pd

# Importing the dataset dataset =
pd.read_csv('Churn_Modelling.csv') X =
dataset.iloc[:, 3:13].values y = dataset.iloc[:,
13].values

# Encoding categorical data labelencoder_X_1
= LabelEncoder()

X[:, 1] = labelencoder_X_1.fit_transform(X[:, 1]) labelencoder_X_2
= LabelEncoder()

X[:, 2] = labelencoder_X_2.fit_transform(X[:, 2]) onehotencoder
= OneHotEncoder(categorical_features=[1])
```

```
X = onehotencoder.fit_transform(X).toarray()
X = X[:, 1:]


# Splitting the dataset into the Training set and Test set

X_train, X_test, y_train, y_test = train_test_split(

        X, y, test_size=0.2, random_state=0)


# Fitting XGBoost to the training data my_model

= xgb.XGBClassifier() my_model.fit(X_train,

y_train)


# Predicting the Test set results y_pred

= my_model.predict(X_test)


# Making the Confusion Matrix cm =

confusion_matrix(y_test, y_pred)
```

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

```
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')
```

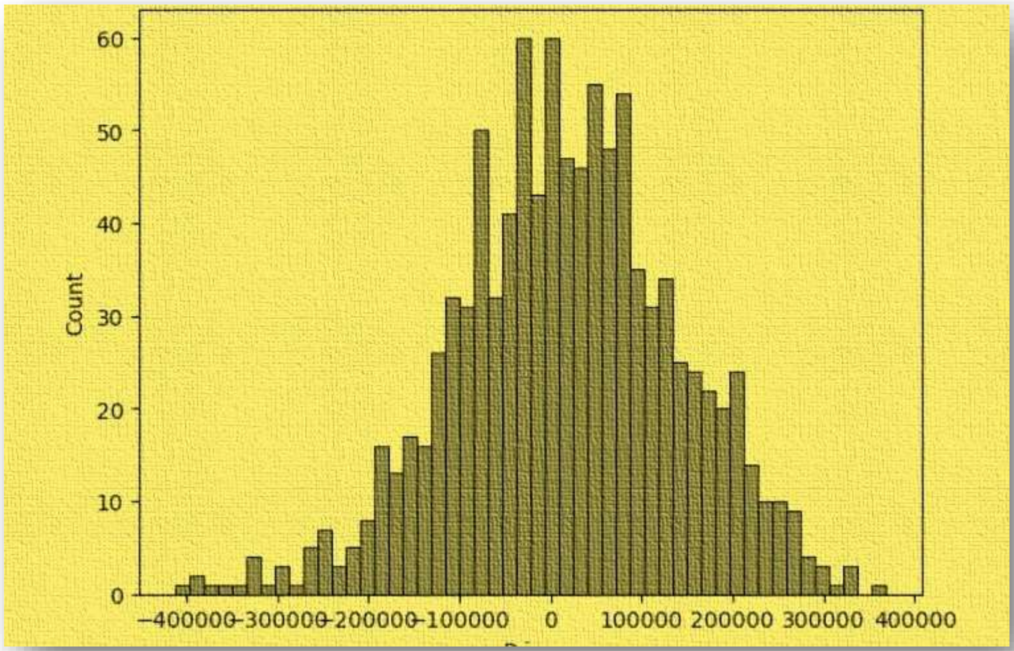
OUTPUT :

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

INPUT:

```
            sns.histplot((Y_test-Prediction4), bins=50) OUTPUT:
```

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



Count
0
10
20
30
40
50
60
−400000
−300000
−200000
−100000
0
100000
200000
300000
400000

In this part you will begin building your project by loading and preprocessing the dataset. Start building the house price prediction model by loading and preprocessing the dataset. Load the housing dataset and preprocess the data.

Data Preprocessing in Machine learning

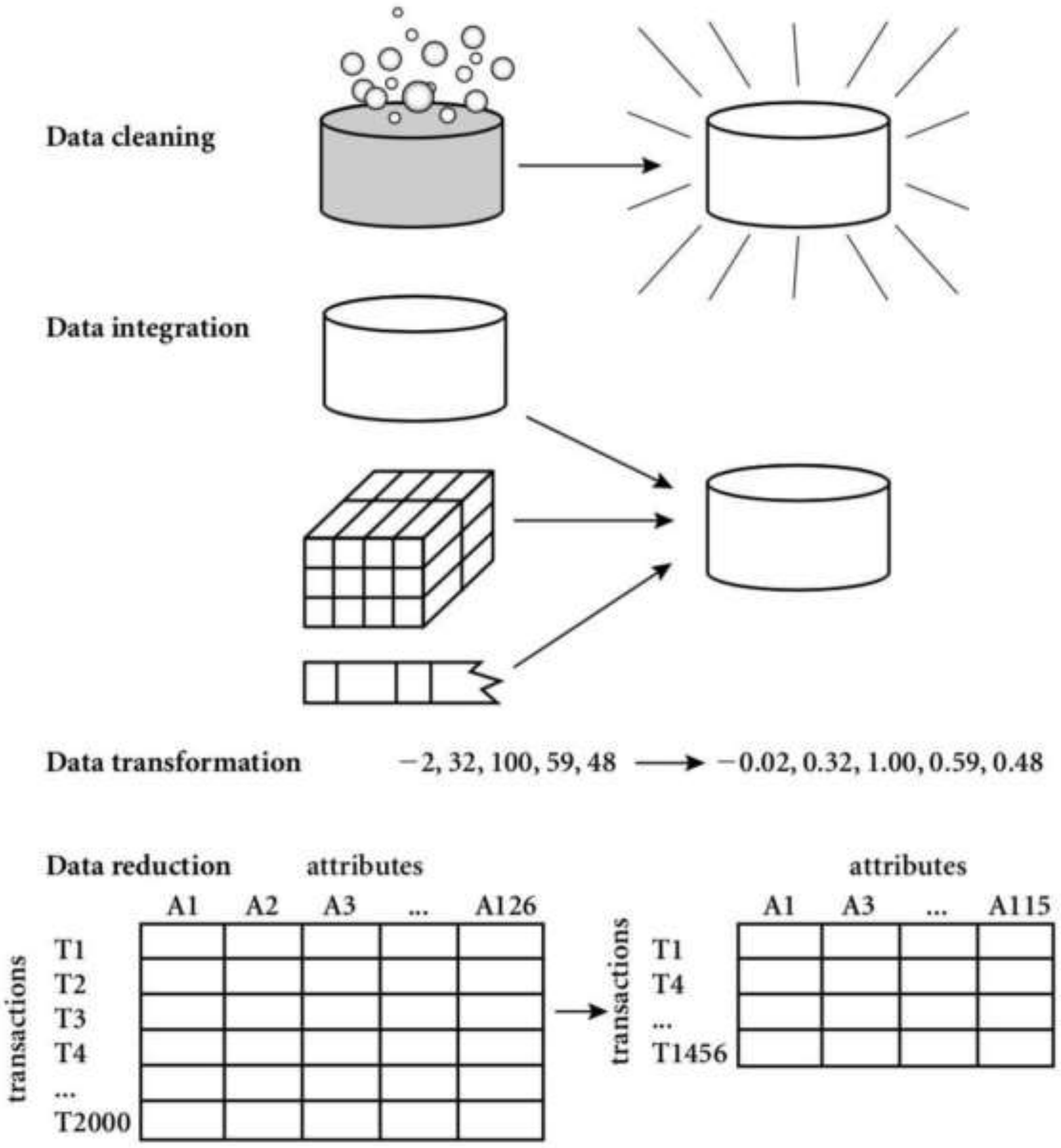
Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Data Preprocessing:
Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, greater is the reliance on the produced results. **Incomplete, noisy, and inconsistent data** are the properties of large real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.

- **Incomplete data** can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions.
- There are many possible reasons for **noisy data** (having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Errors in data transmission can also occur.

  Incorrect data may also result from inconsistencies in naming conventions or data codes used, or inconsistent formats for input fields, such as date.


Data cleaning
Data integration
Data transformation
−2, 32, 100, 59, 48 → −0.02, 0.32, 1.00, 0.59, 0.48
Data reduction
attributes
A1 A2 A3 ... A126
transactions
T1
T2
T3
T4
...
T2000
attributes
A1 A3 ... A115
transactions
T1
T4
...
T1456

Import the required libraries

```
import warnings
warnings.filterwarnings('ignore')
import numpy as np  import pandas
as pd  import matplotlib.pyplot as
plt from operator import itemgetter
from sklearn.experimental import
enable_iterative_imputer  from sklearn.impute import
IterativeImputer from sklearn.preprocessing import
OrdinalEncoder from category_encoders.target_encoder
import TargetEncoder from sklearn.preprocessing import
StandardScaler  from sklearn.ensemble import
(GradientBoostingRegressor,
GradientBoostingClass ifier) import xgboost
```

Load the dataset for training and testing

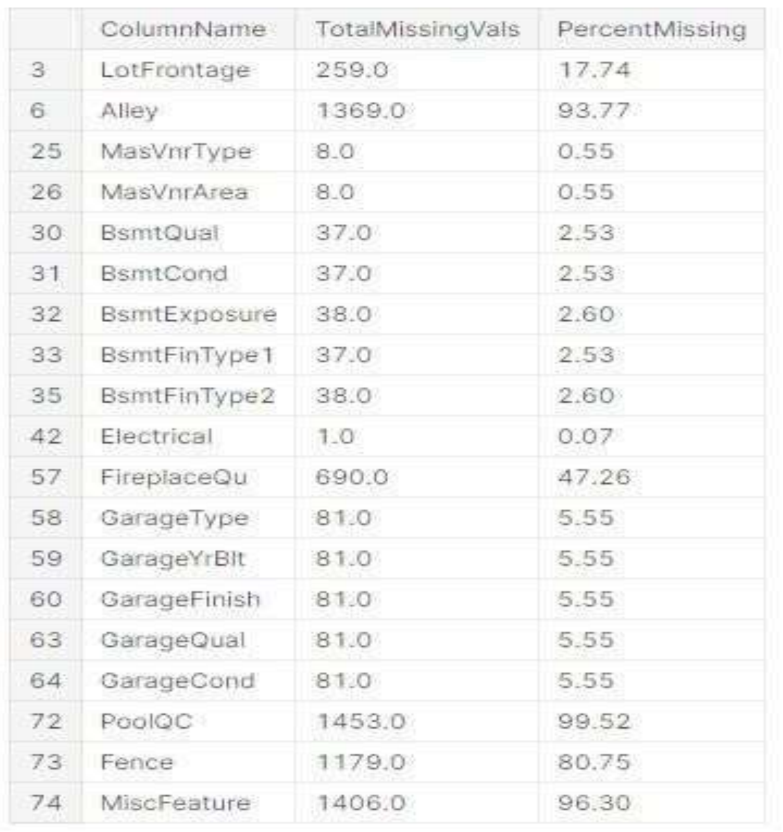
```
train = pd.read_csv('../input/house-prices-advanced-regressiontechniques/trai
n.csv') test = pd.read_csv('../input/house-prices-
advancedregressiontechniques/test. csv')
```

Data Cleaning

Find the missing percentage of each columns in training set.

```
def find_missing_percent(data):
    """
    Returns dataframe containing the total missing values and percentage of to
tal     missing values of a column.
    """     miss_df =
pd.DataFrame({'ColumnName':[],'TotalMissingVals':[],'PercentMiss
ing':[]})     for col in data.columns:         sum_miss_val =
data[col].isnull().sum()
        percent_miss_val = round((sum_miss_val/data.shape[0])*100,2)
miss_df = miss_df.append(dict(zip(miss_df.columns,[col,sum_miss_val,pe
rcent_miss_val])),ignore_index=True)     return miss_df
```

```
miss_df = find_missing_percent(train) '''Displays
columns with missing values'''
display(miss_df[miss_df['PercentMissing']>0.0])
print("\n")
print(f"Number of columns with missing
values:{str(miss_df[miss_df['PercentMis sing']>0.0].shape[0])}")
```



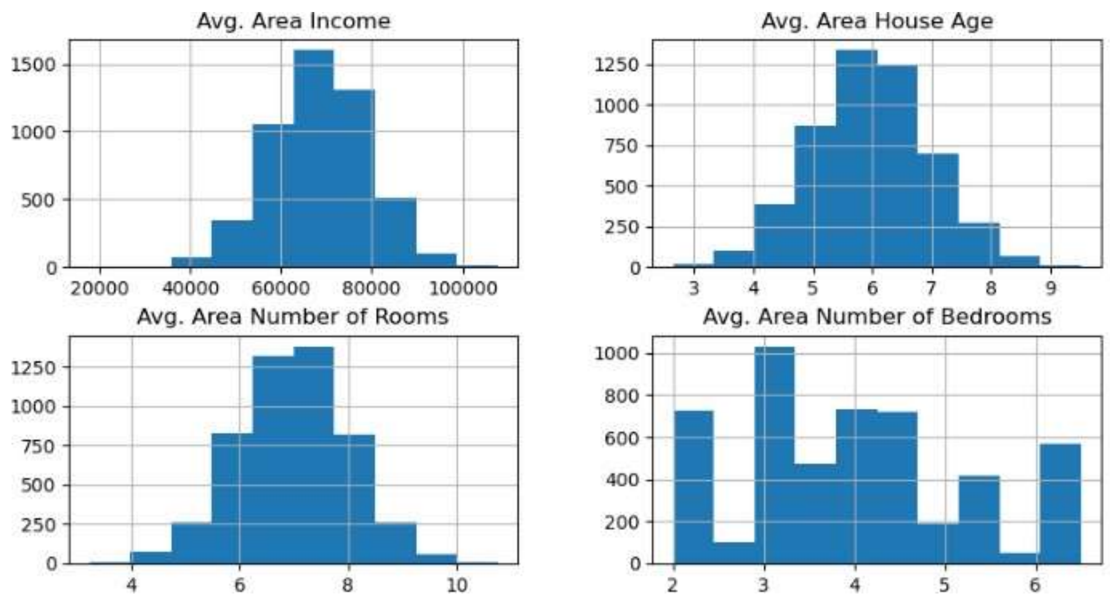
	ColumnName	TotalMissingVals	PercentMissing
3	LotFrontage	259.0	17.74
6	Alley	1369.0	93.77
25	MasVnrType	8.0	0.55
26	MasVnrArea	8.0	0.55
30	BsmtQual	37.0	2.53
31	BsmtCond	37.0	2.53
32	BsmtExposure	38.0	2.60
33	BsmtFinType1	37.0	2.53
35	BsmtFinType2	38.0	2.60
42	Electrical	1.0	0.07
57	FireplaceQu	690.0	47.26
58	GarageType	81.0	5.55
59	GarageYrBlt	81.0	5.55
60	GarageFinish	81.0	5.55
63	GarageQual	81.0	5.55
64	GarageCond	81.0	5.55
72	PoolQC	1453.0	99.52
73	Fence	1179.0	80.75
74	MiscFeature	1406.0	96.30

```
Number of columns with
missing values:19
```

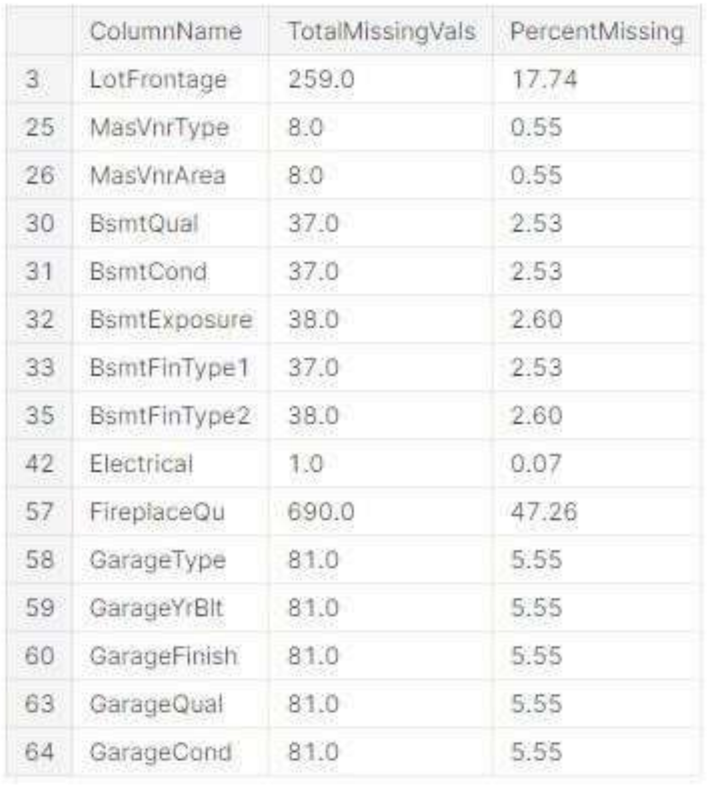
1.2

```
drop_cols = miss_df[miss_df['PercentMissing']
>70.0].ColumnName.tolist() print(f"Number of columns with more than
70%: {len(drop_cols)}") train = train.drop(drop_cols,axis=1) test =
test.drop(drop_cols,axis =1)

miss_df = miss_df[miss_df['ColumnName'].isin(train.columns)]
'''Columns to Impute'''
impute_cols = miss_df[miss_df['TotalMissingVals']>0.0].ColumnName.tolist()
miss_df[miss_df['TotalMissingVals']>0.0]
```



Out[5]:



	ColumnName	TotalMissingVals	PercentMissing
3	LotFrontage	259.0	17.74
25	MasVnrType	8.0	0.55
26	MasVnrArea	8.0	0.55
30	BsmtQual	37.0	2.53
31	BsmtCond	37.0	2.53
32	BsmtExposure	38.0	2.60
33	BsmtFinType1	37.0	2.53
35	BsmtFinType2	38.0	2.60
42	Electrical	1.0	0.07
57	FireplaceQu	690.0	47.26
58	GarageType	81.0	5.55
59	GarageYrBlt	81.0	5.55
60	GarageFinish	81.0	5.55
63	GarageQual	81.0	5.55
64	GarageCond	81.0	5.55

Now, we categorize the features depending on their datatype (int, float, object) and then calculate the number of them

```
 obj = (dataset.dtypes == 'object') object_cols =  list(obj[obj].index)
print("Categorical variables:",len(object_cols))
```

```
int_ = (dataset.dtypes == 'int') num_cols = list(int_[int_].index)   print("Integer
variables:",len(num_cols))

fl = (dataset.dtypes == 'float') fl_cols =
list(fl[fl].index)   print("Float
variables:",len(fl_cols))
```

Output:

```
Categorical variables : 4
Integer variables : 6
Float variables : 3
```

The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model.

Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

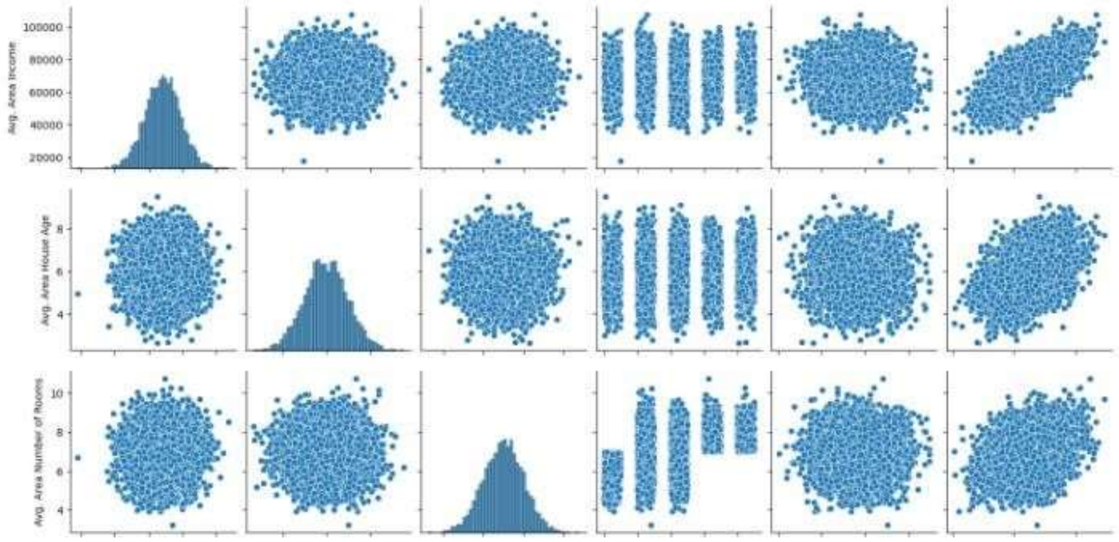
There are mainly two ways to handle missing data, which are:

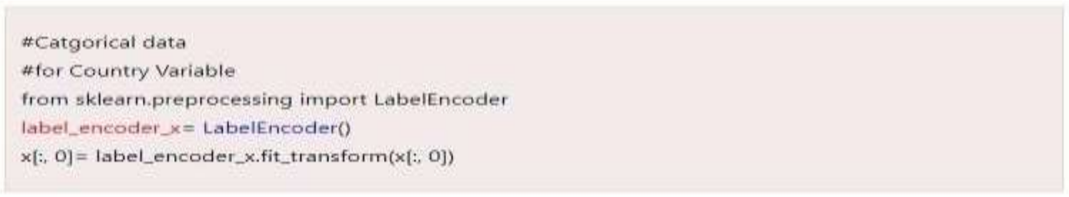
By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach.

To handle missing values, we will use **Scikit-learn** library in our code, which contains various libraries for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library. Below is the code for it:

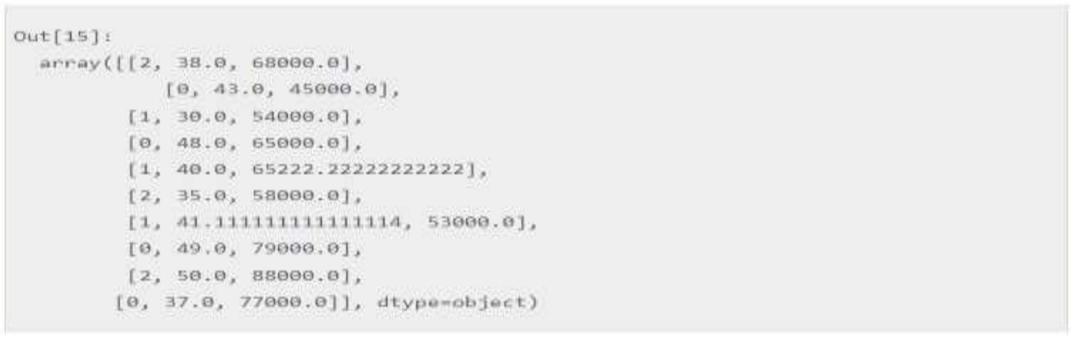
```
'''
These columns have IQR equal to zero. Freedman Diaconis Rule doesn't work sign ificantly well for these
columns.
Use sturges rule to find the optimal number of bins for the columns.
'''
cols_list = ['LowQualFinSF','BsmtFinSF2','BsmtHalfBath','KitchenAbvGr',
'EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal']
# Except ID hist_cols = numeric_cols[1:] for i
in range(0,len(hist_cols),2):
   if(i == len(hist_cols)-1):
      plot_histogram(train,hist_cols[i],hist_cols[i],cols_list,True)     else:
plot_histogram(train,hist_cols[i],hist_cols[i+1],cols_list)
```





```
#Catgorical data
#for Country Variable
from sklearn.preprocessing import LabelEncoder
label_encoder_x= LabelEncoder()
x[:, 0]= label_encoder_x.fit_transform(x[:, 0])
```

Output:



```
Out[15]:
  array([[2, 38.0, 68000.0],
             [0, 43.0, 45000.0],
         [1, 30.0, 54000.0],
         [0, 48.0, 65000.0],
         [1, 40.0, 65222.22222222222],
         [2, 35.0, 58000.0],
         [1, 41.111111111111114, 53000.0],
         [0, 49.0, 79000.0],
         [2, 50.0, 88000.0],
        [0, 37.0, 77000.0]], dtype=object)
```

Explanation:

In above code, we have imported **LabelEncoder** class of **sklearn library**. This class has successfully encoded the variables into digits.

But in our case, there are three country variables, and as we can see in the above output, these variables are encoded into 0, 1, and 2. By these values, the machine learning model may assume that there is some correlation between these variables which will produce the wrong output. So to remove this issue, we will use **dummy encoding**.

To Develop the project development part 2 is build a model evaluation, model training by using Linear Regression and XG boost regression.

In this proposed system, we focus on predicting the house price values using machine learning algorithms like XG Boost regression model and Linear Regression . We proposed the system "House price prediction using Machine Learning" we have predicted the House price using XG boost regression and linear regression model. In this proposed system, we were able to train the machine from the various attributes of data points from the past to make a future prediction. We took data from the previous year stocks to train the model .The data set we used was from the official organization. Some of data was used to train the machine and the rest some data is used to test the data. The basic approach of the supervised learning model is to learn the patterns and relationships in the data from the training set and then reproduce them for the test data. We used the python pandas library for data processing which combined different datasets into a data frame. The raw data makes us to prepare the data for feature identification. The attributes were stories, no. of bed rooms, bath rooms, Availability of garage, swimming pool, fire place, year built, area in soft, sale price for a particular house. We used all these features to train the machine on XG boost regression and predicted the house price, which is the price for a given day. We also quantified the accuracy by using the predictions for the test set and the actual values. The proposed system gives the Predicted price

ALOGORITHM:

We used the python pandas library for data processing which combined different datasets into a data frame. The raw data makes us to prepare the data for feature identification. XG for regression builds an additive model in a forward stage wise fashion. It allows for the optimization of arbitrary differentiable loss functions. In each stage, a regression tree is fit on the negative XG of the given loss function. The idea of boosting came out of the idea of whether a weak learner can be modified to become better. A weak hypothesis is defined as one whose performance is at least slightly better than random chance. The Objective is to minimize the loss of the model by adding weak hypothesis using a XG descent like procedure. This class of algorithms was described as a stage-wise additive model. This is because one new weak learner is added at a time and existing weak learners in the model are frozen and left unchanged.

Step 1: Load the data set df = pd.read csv("ml_house_data_set.csv")

Step 2: Replace categorical data with one-hot encoded data

Step 3: Remove the sale price from the feature data

Step 4: Create the features and labels X and Y arrays.

Step 5: Split the data set in a training set (70%) and a test set (30%).

Step 6: Fit regression model.

Step 7: Save the trained model to a file trained_house_classifier_model.pkl

Step 8: Predict house worth using predict function

Model Building and evaluation:

Model 1 : Linear regression

Input;

model_lr=LinearRegression() model_lr.fit(X_train_scal,

Y_train)

Output:

LinearRegression()

Predicting Prices

Prediction1 = model_lr.predict(X_test_scal)

Evaluation of predicted data:

Input : plt.figure(figsize=(12,6)) plt.plot(np.arange(len(Y_test)), Y_test,

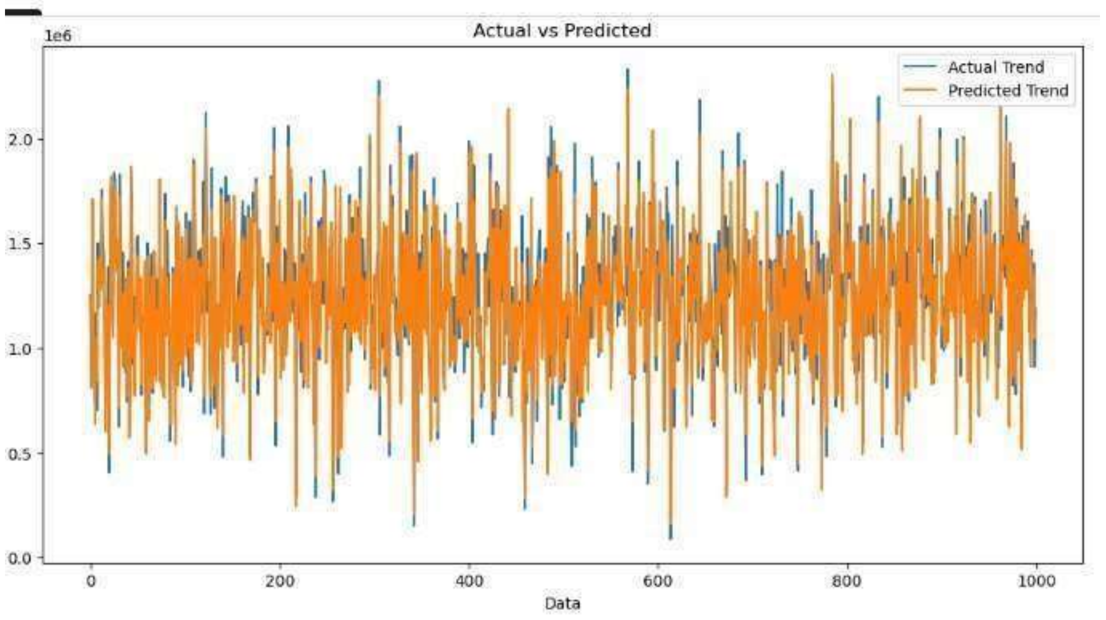
label='Actual Trend') plt.plot(np.arange(len(Y_test)), Prediction1,

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')
```



To Find a Mean absolute Error:

Input:

```
sns.histplot((Y_test-Prediction1), bins=50)
```

Output:

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

Input: print(r2_score(Y_test, Prediction1))

print(mean_absolute_error(Y_test,

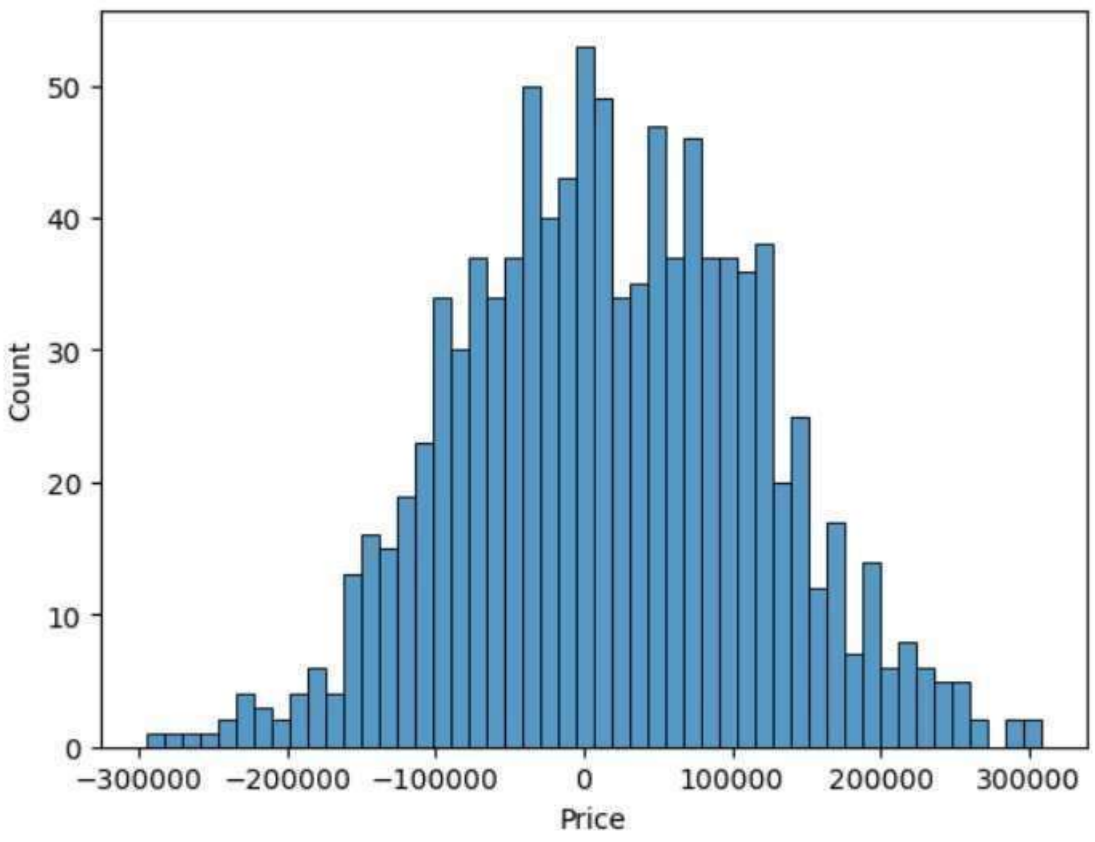
Prediction1)) print(mean_squared_error(Y_test, Prediction1))

Output:

```
0.9182928179392918

82295.49779231755

10469084772.975954
```



Model 2: XG Boost Regressor :

Input:

```
model_xg = xg.XGBRegressor()  model_xg.fit(X_train_scal,

Y_train)
```

Output:

```
  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)
```

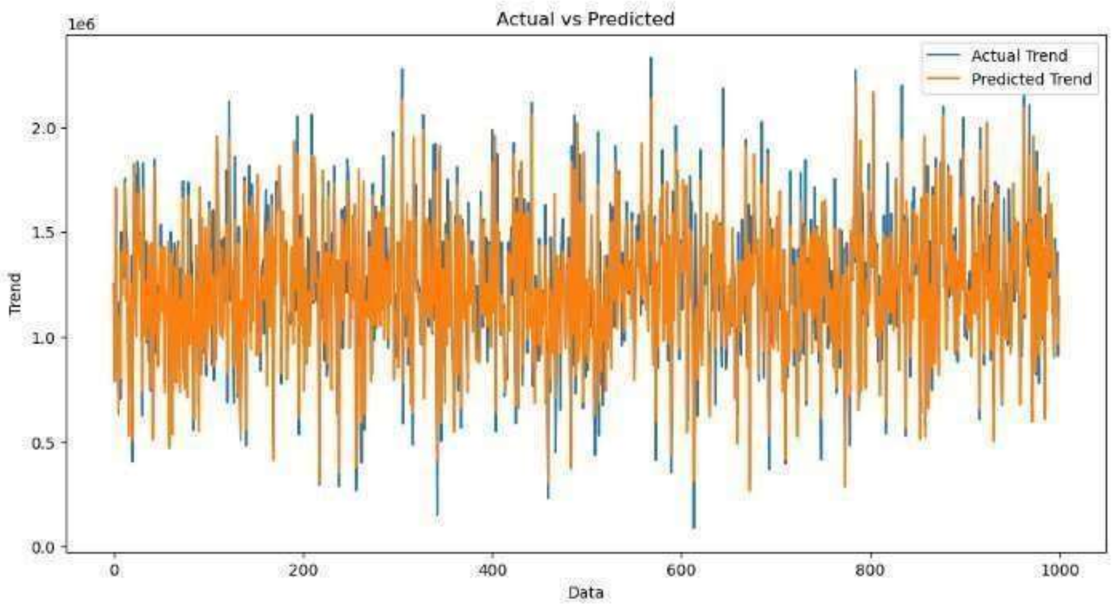
Evaluation of Predicting Prices:

Input:

```
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')
```

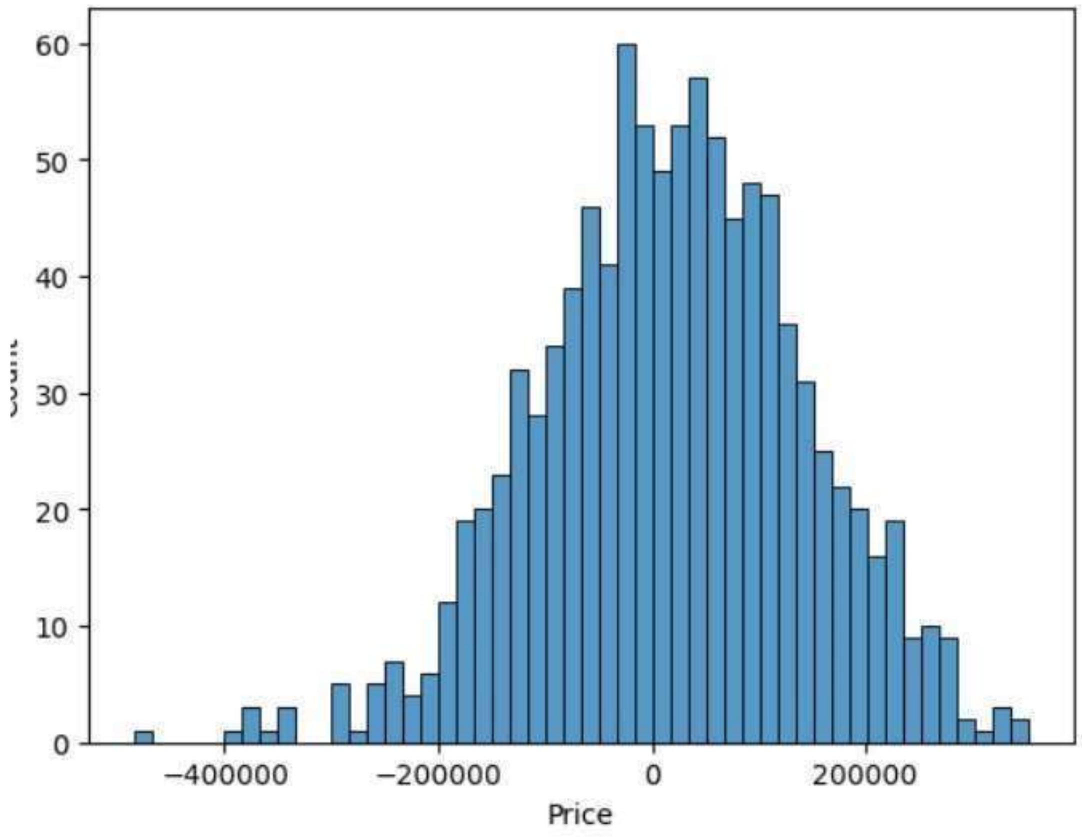
Output:

```
Text(0.5, 1.0, 'APrediction5 = model_xg.predict(X_test_scal)
```



```
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 sns.histplot((Y_test-Prediction4), bins=50)
<Axes: xlabel='Price', ylabel='Count'>
```



CONCLUSION:

This project entitled “House Price Prediction algorithm had all other algorithms regarding to model building and evaluation in this project analyze the importance of features in predicting house prices to gain insights into the factors that influence the market.
Remember that house price prediction is a complex task influenced by many factors, and no model will be perfect. It's important to continuously improve and update your model to ensure its accuracy. Additionally, ethical considerations and fairness in pricing should be taken into account when developing such models to avoid bias and discrimination.