SURYA GROUP OF INSTITUTION

VIKRAVANDI 605-652

PREDICTION HOUSE PRICES USING MACHINE LEARNING

NAAN MUDHALVAN

Subject code-SB3001

Course Name:Experience Based Practical learning

PREPARED BY:

ANJUGAM.C

REGNO: 42222116002

DEPARTMENT:ECE

3rd year

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

Problem Statement: The housing market is an important and complex sector that impacts people's lives in many ways. For many individuals and families, buying a house is one of the biggest investments they will make in their lifetime. Therefore, it is essential to accurately predict the prices of houses so that buyers and sellers can make informed decisions. This project aims to use machine learning techniques to predict house prices based on various features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors.

Project Steps

Problem Definition: The problem is to predict house prices using machine learning techniques. The objective is to develop a model that accurately predicts the prices of houses based on a set of features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking:

1. Data Source: Choose a dataset containing information about houses, including

features like location, square footage, bedrooms, bathrooms, and price.

2. Data Preprocessing: Clean and preprocess the data, handle missing values, and

convert categorical features into numerical representations.

3. Feature Selection: Select the most relevant features for predicting house prices.

4. Model Selection: Choose a suitable regression algorithm (e.g., Linear Regression,

Random Forest Regressor) for predicting house prices.

5. Model Training: Train the selected model using the preprocessed data.

6. Evaluation: Evaluate the model's performance using metrics like Mean Absolute

Error (MAE), Root Mean Squared Error (RMSE), and R-squared.

Data Source:

So to deal with this kind of issues Today we will be preparing a MACHINE LEARNING Based model, trained on the House Price Prediction Dataset.

You can download the dataset from the link

Dataset Link: https://www.kaggle.com/datasets/vedavyasv/usa-housing

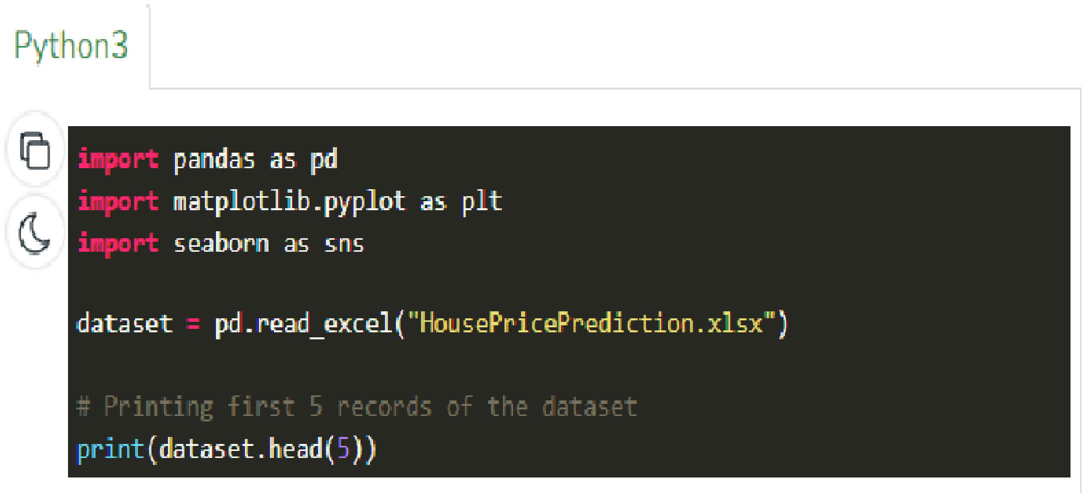
Importing Libraries and Dataset

Here we are using

 Pandas – To load the Dataframe

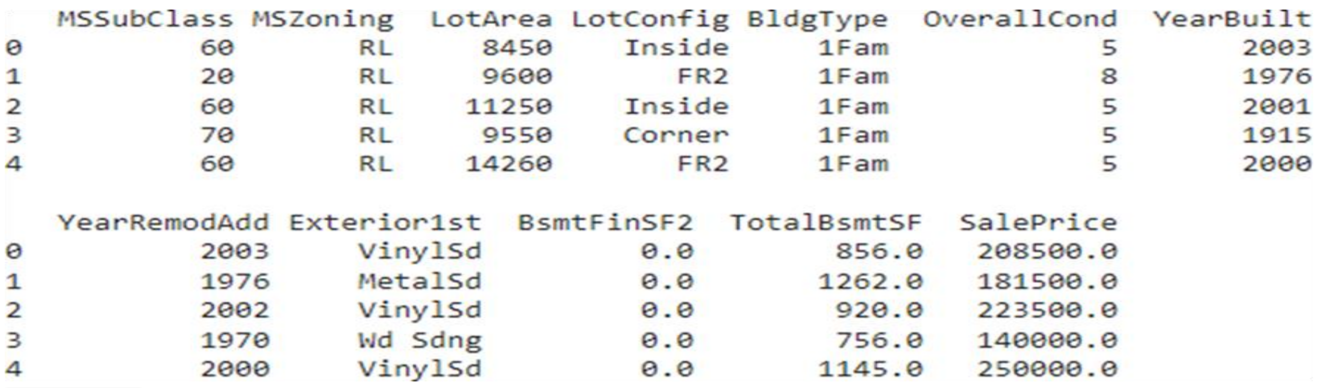
 Matplotlib – To visualize the data features i.e. barplot

 Seaborn – To see the correlation between features using heatmap

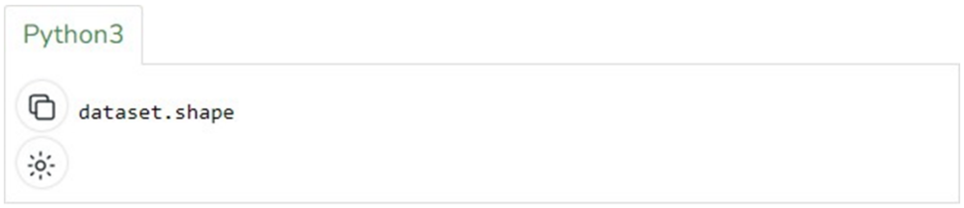


```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_excel("HousePricePrediction.xlsx")

# Printing first 5 records of the dataset
print(dataset.head(5))
```

Output:



```
   MSSubClass MSZoning  LotArea LotConfig BldgType  OverallCond  YearBuilt
0          60       RL     8450    Inside     1Fam            5       2003
1          20       RL     9600       FR2     1Fam            8       1976
2          60       RL    11250    Inside     1Fam            5       2001
3          70       RL     9550    Corner     1Fam            5       1915
4          60       RL    14260       FR2     1Fam            5       2000

   YearRemodAdd Exterior1st  BsmtFinSF2  TotalBsmtSF  SalePrice
0          2003     VinylSd         0.0        856.0   208500.0
1          1976     MetalSd         0.0       1262.0   181500.0
2          2002     VinylSd         0.0        920.0   223500.0
3          1970     Wd Sdng         0.0        756.0   140000.0
4          2000     VinylSd         0.0       1145.0   250000.0
```



Python3

```
dataset.shape
```

Output:

```
(2919,13)
```

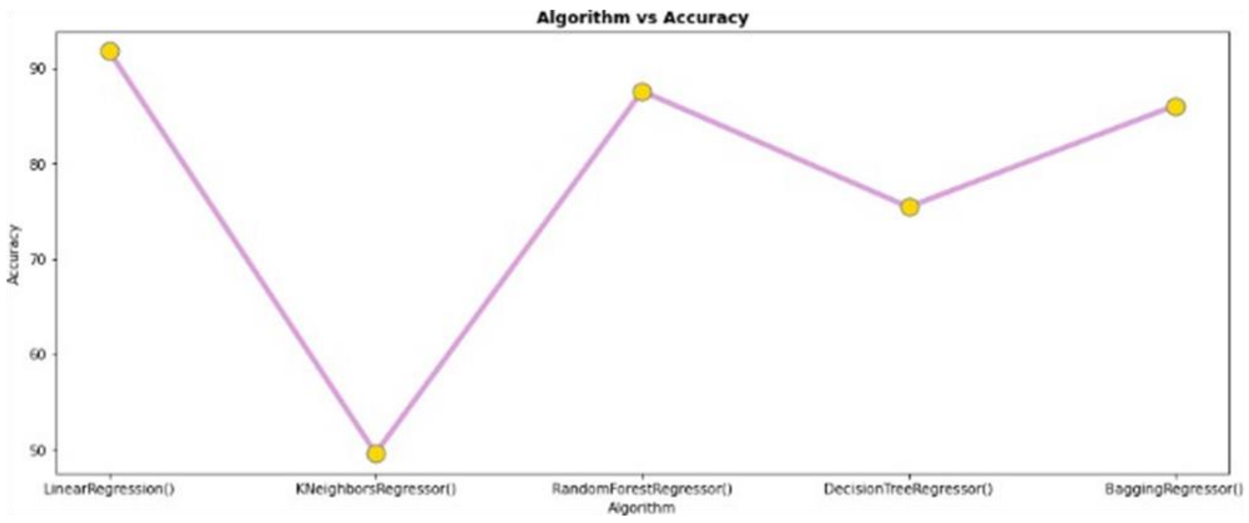
Data Preprocessing:

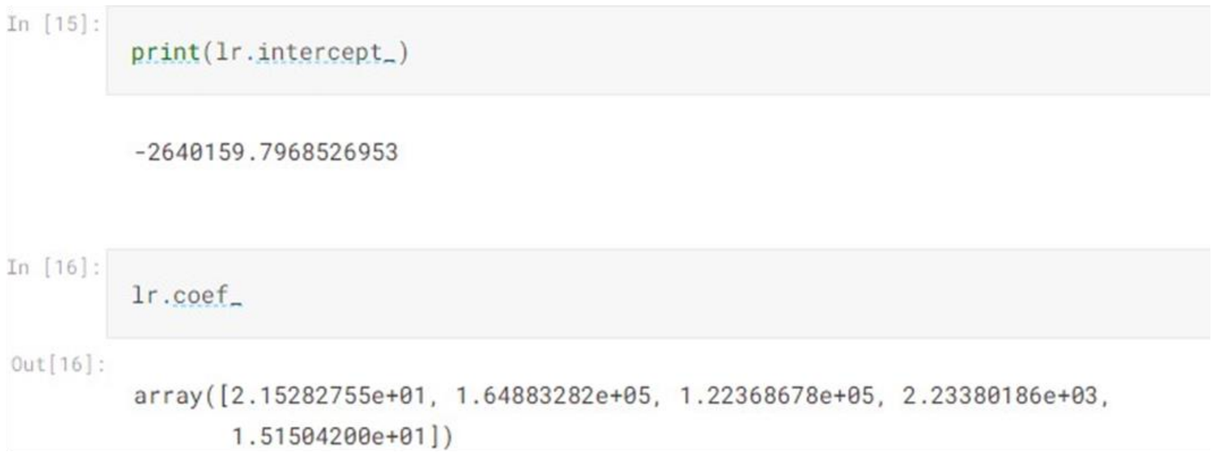
Now, we categorize the features depending on their datatype (int, float, object) and then calculate the number of them

Model Selection

```
lr = LinearRegression()
dt = DecisionTreeRegressor() rn = RandomForestRegressor() knn =
KNeighborsRegressor() sgd = SGDRegressor()
br = BaggingRegressor() li = [lr,knn,rn,dt,br] di = {}
for i in li:
i.fit(X_train,y_train) ypred = i.predict(X_test)
print(i,":",r2_score(ypred,y_test)*100)
di.update({str(i):i.score(X_test,y_test)*100})
plt.figure(figsize=(15, 6))
plt.title("Algorithm vs Accuracy", fontweight='bold') plt.xlabel("Algorithm")
plt.ylabel("Accuracy")
plt.plot(di.keys(),di.values(),marker='o',color='plum',linewidth=4,markersize=13,
         markerfacecolor='gold',markeredgecolor='slategray') plt.show()
```

```
LinearRegression() : 91.00355108791786
KNeighborsRegressor() : 20.484074470563286
RandomForestRegressor() : 83.83229998741602
DecisionTreeRegressor() : 73.30037072629774
BaggingRegressor() : 81.95467285948747
```





```
In [15]:  print(lr.intercept_)

          -2640159.7968526953

In [16]:  lr.coef_

Out[16]:  array([2.15282755e+01, 1.64883282e+05, 1.22368678e+05, 2.23380186e+03,
                 1.51504200e+01])
```

MODEL BULIDING AND EVALUTION OF PREDICATED DATA

```
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,

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.
```

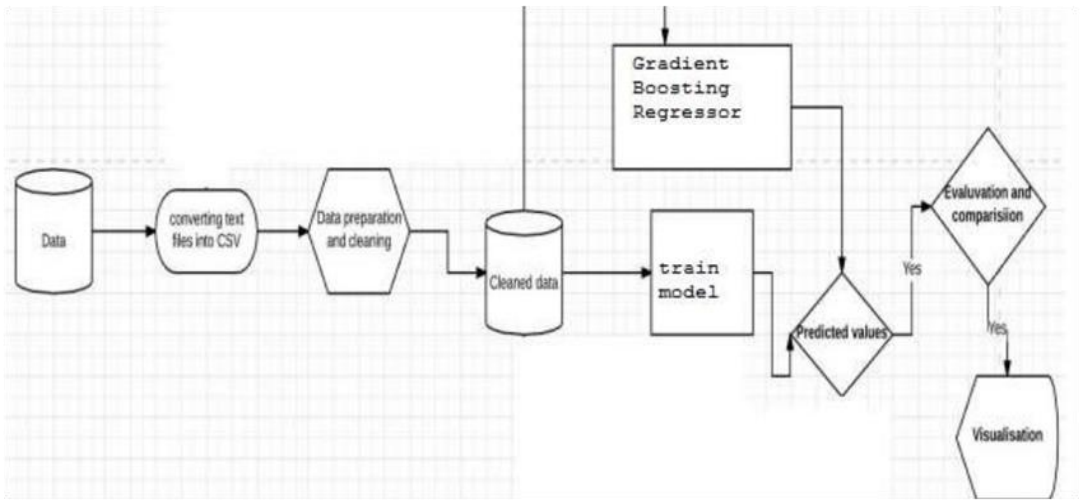
PROPOSED SYSTEM:

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

ARCHITECTURE:



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.

Step1: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

Step4: 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:

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

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

OUTPUT:

```
XGB Regressor
XGB Regressor (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)
```

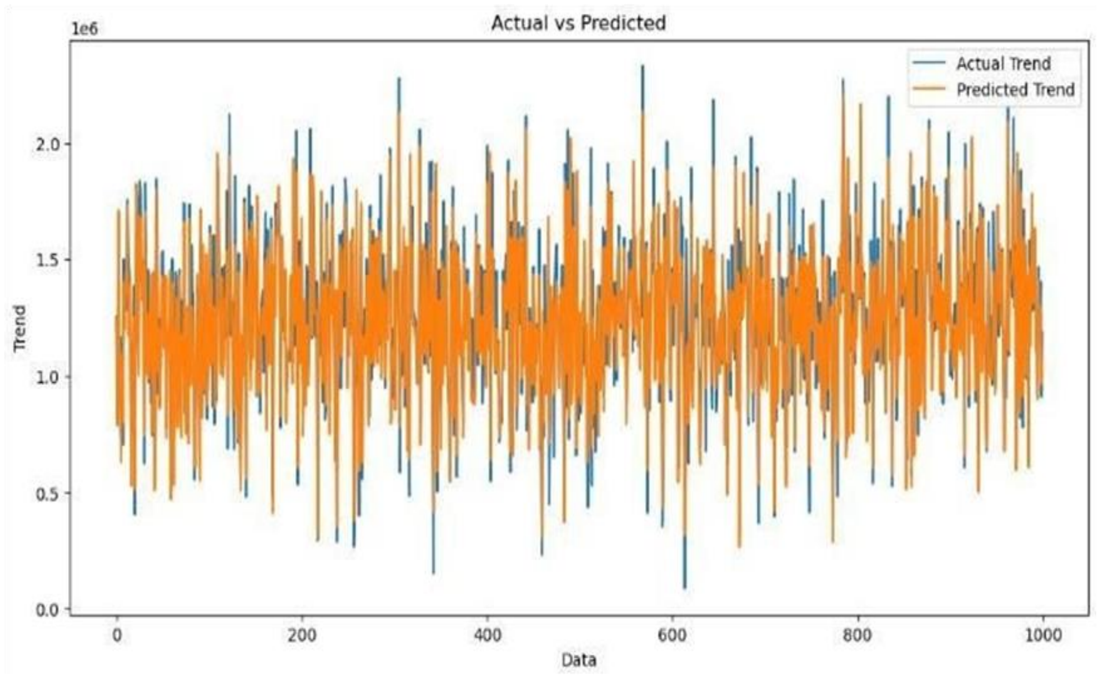
EVALUATION OF PREDICTING DATA:

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)), 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')
```



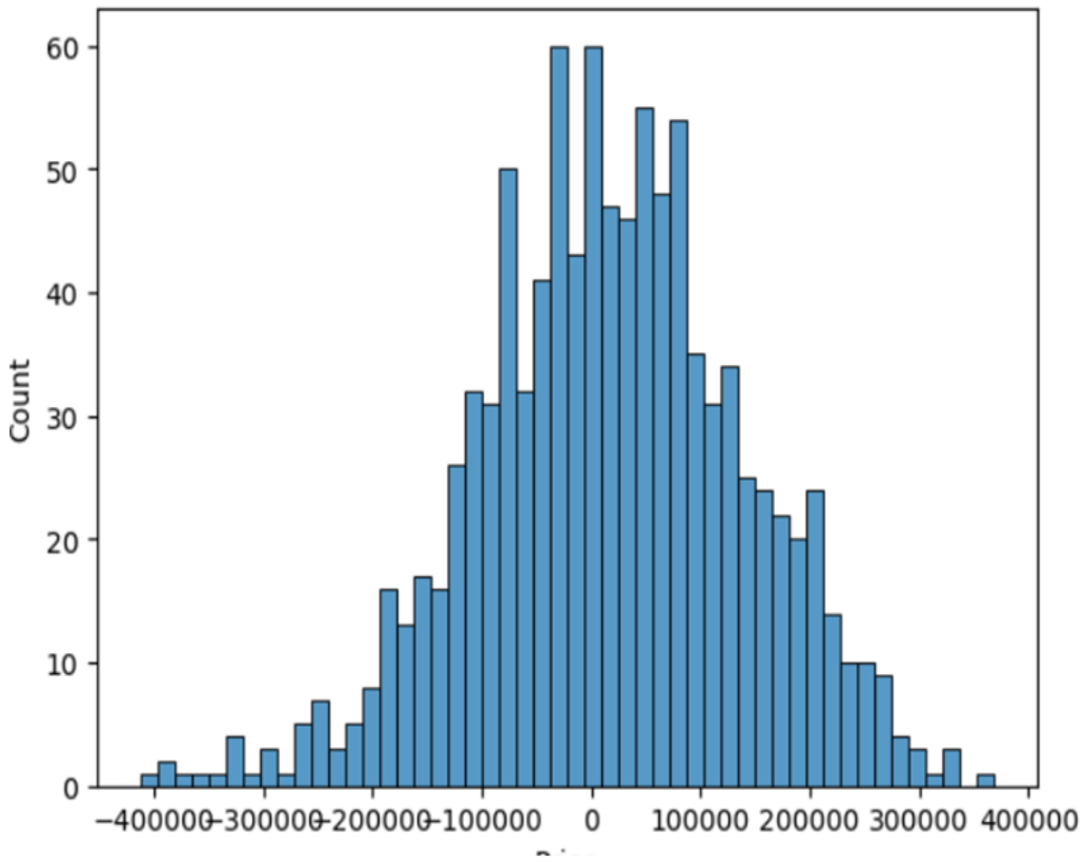
INPUT;

INPUT;

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

OUTPUT;

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



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.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

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.

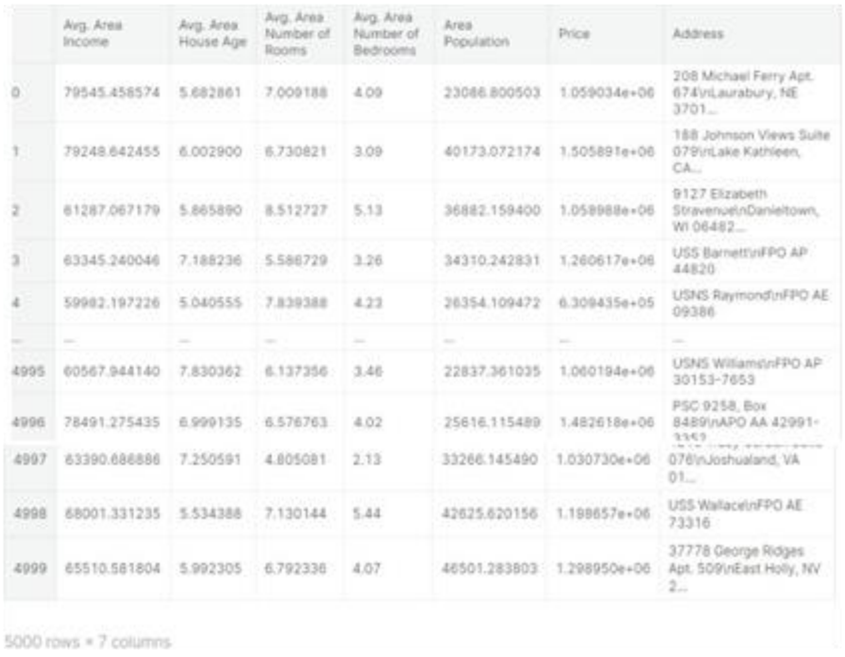
Data cleaning can be applied to filling in missing values, remove noise, resolving inconsistencies, identifying and removing outliers in the data.

Data integration merges data from multiple sources into a coherent data store, such as a data warehouse.

Data transformations, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.

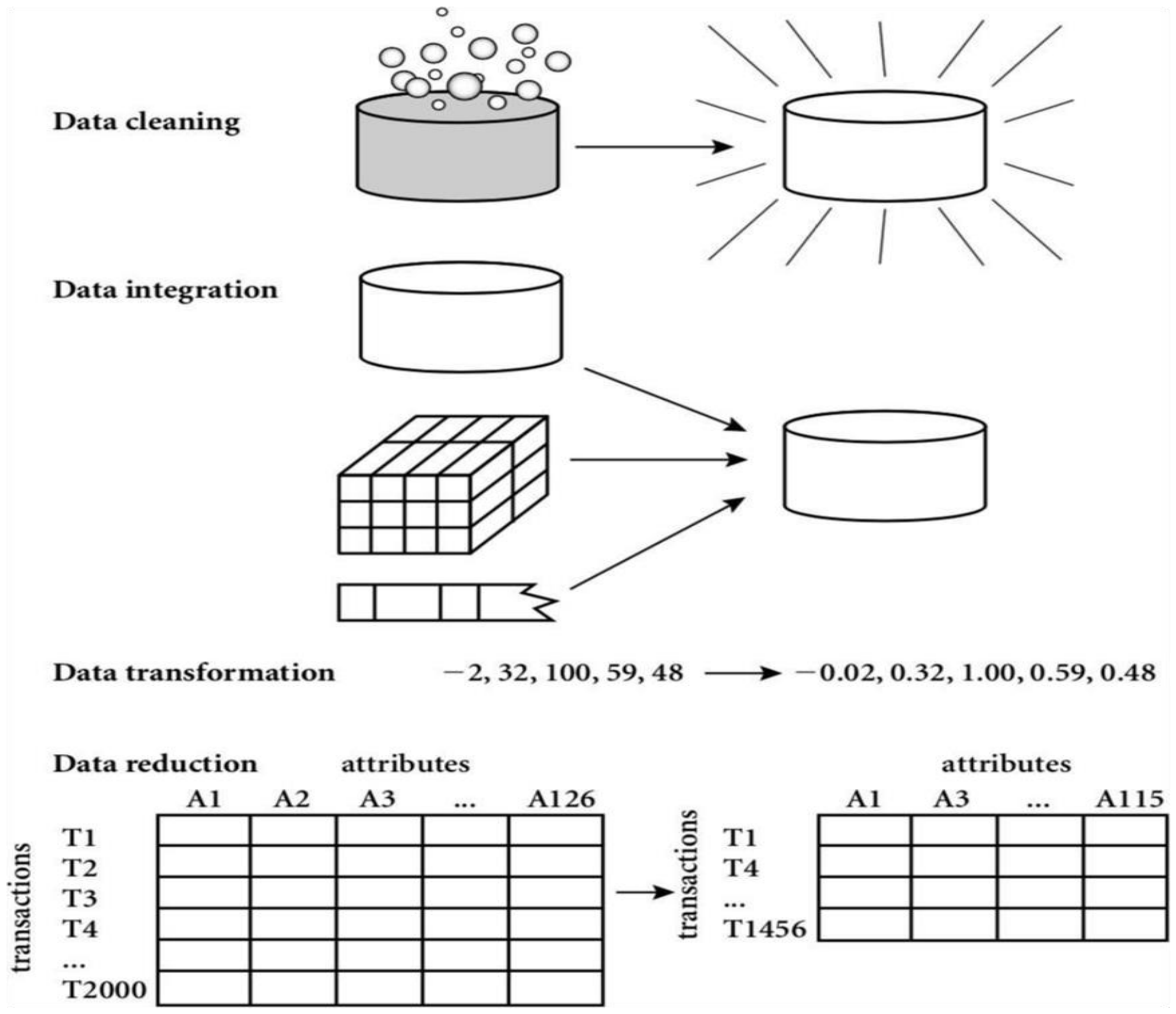
Data reduction can reduce the data size by eliminating redundant features, or clustering, for instance**.**

Given data set



	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	[illegible] 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns



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-regression-techniques/trai
n.csv')
```

```
test = pd.read_csv('../input/house-prices-advanced-regression-techniques/test.
csv')
```

1. 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 = 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]
```

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))

Ways to handle missing data:

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

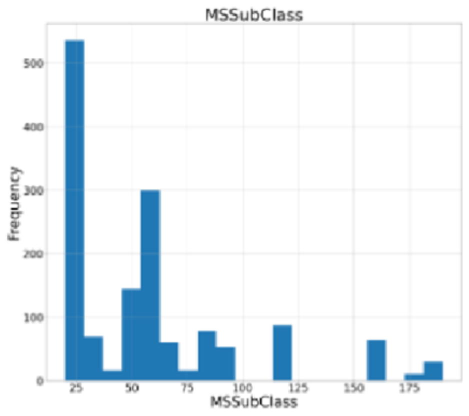
By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

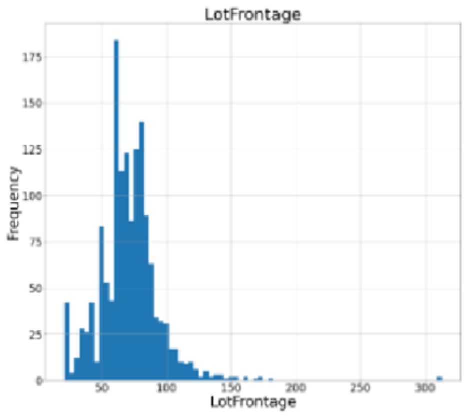
By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc. Here, we will use this approach.

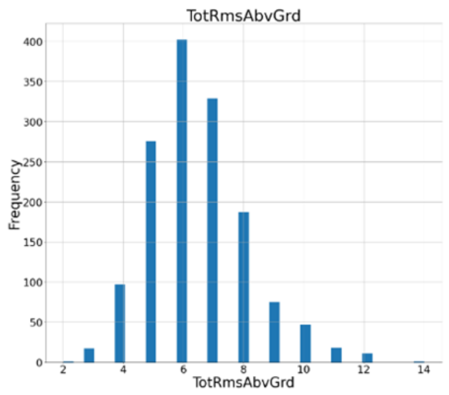
To handle missing values, we will use **Scikit-learn** library in our code, which contains various libraries for building machine learning models. Here we will use **Imputer** class of **sklearn.preprocessing** library. Below is the code for it:

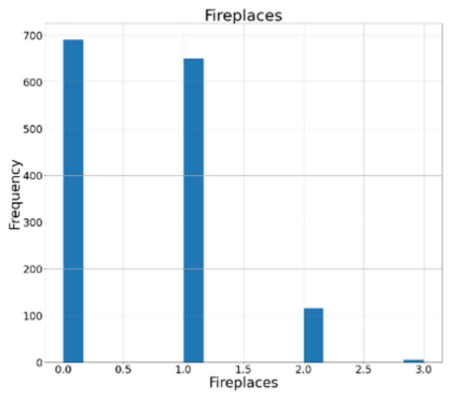
```
These columns have IQR equal to zero. Freedman Diaconis Rule doesn't work sign ificantly well for these columns.
Use sturges rule to find the optimal number of bins for the columns. '''
cols_list = ['LowQualFinSF','BsmtFinSF2','BsmtHalfBath','KitchenAbvGr',
'EnclosedPorch','3SsnPorch','ScreenPorch','PoolArea','MiscVal']

# 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
```

```
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
```









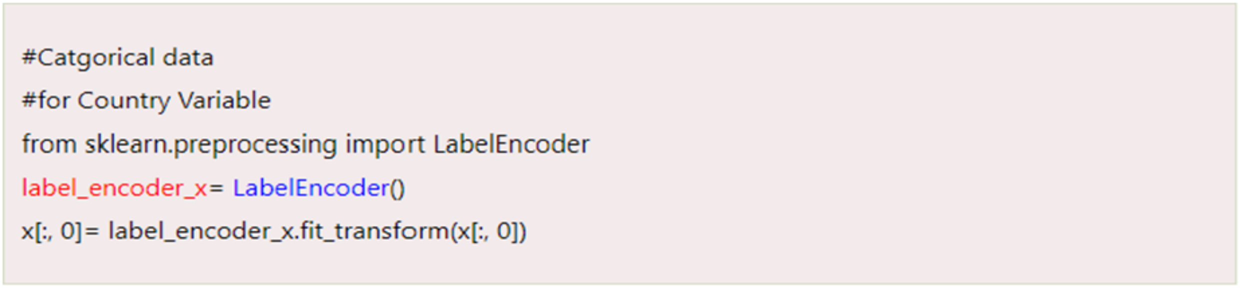
Encoding Categorical data:

Categorical data is data which has some categories such as, in our dataset; there are two categorical variable, **Country**, and **Purchased**.

Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

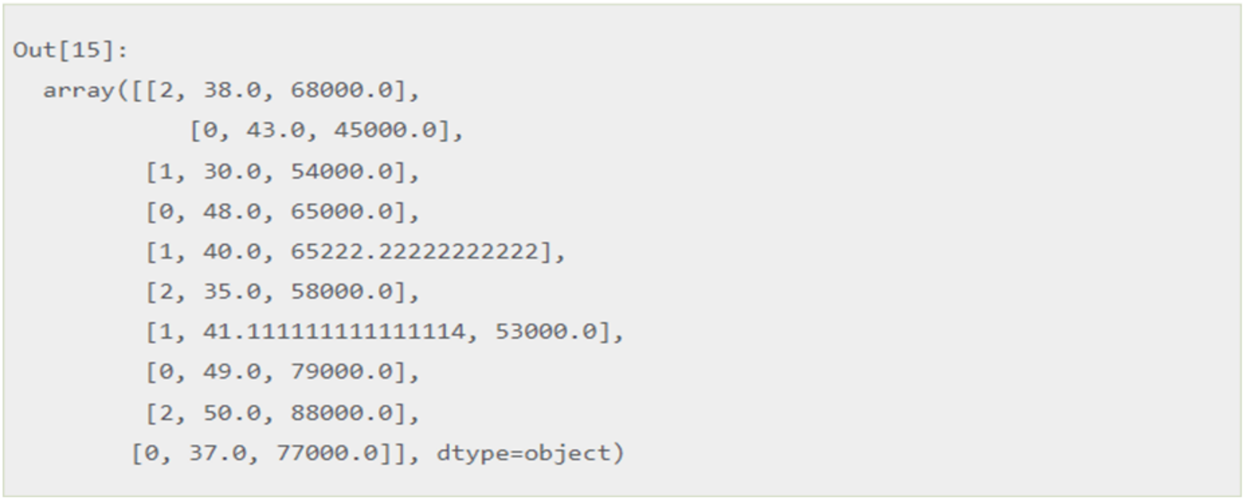
For Country variable:

Firstly, we will convert the country variables into categorical data. So to do this, we will use **LabelEncoder()** class from **preprocessing** library.



```
#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**.

Dummy Variables:

Dummy variables are those variables which have values 0 or 1. The 1 value gives the presence of that variable in a particular column, and rest variables become 0. With dummy encoding, we will have a number of columns equal to the number of categories.

In our dataset, we have 3 categories so it will produce three columns having 0 and 1 values. For Dummy Encoding, we will use **OneHotEncoder** class of **preprocessing** library.

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 . 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)

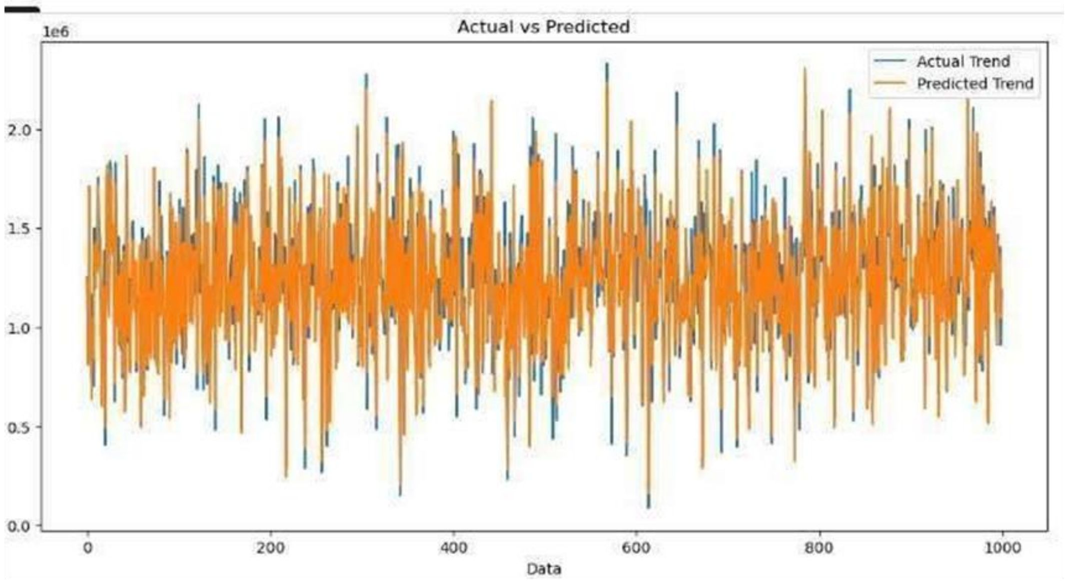
Evaluation of predicted data:

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,

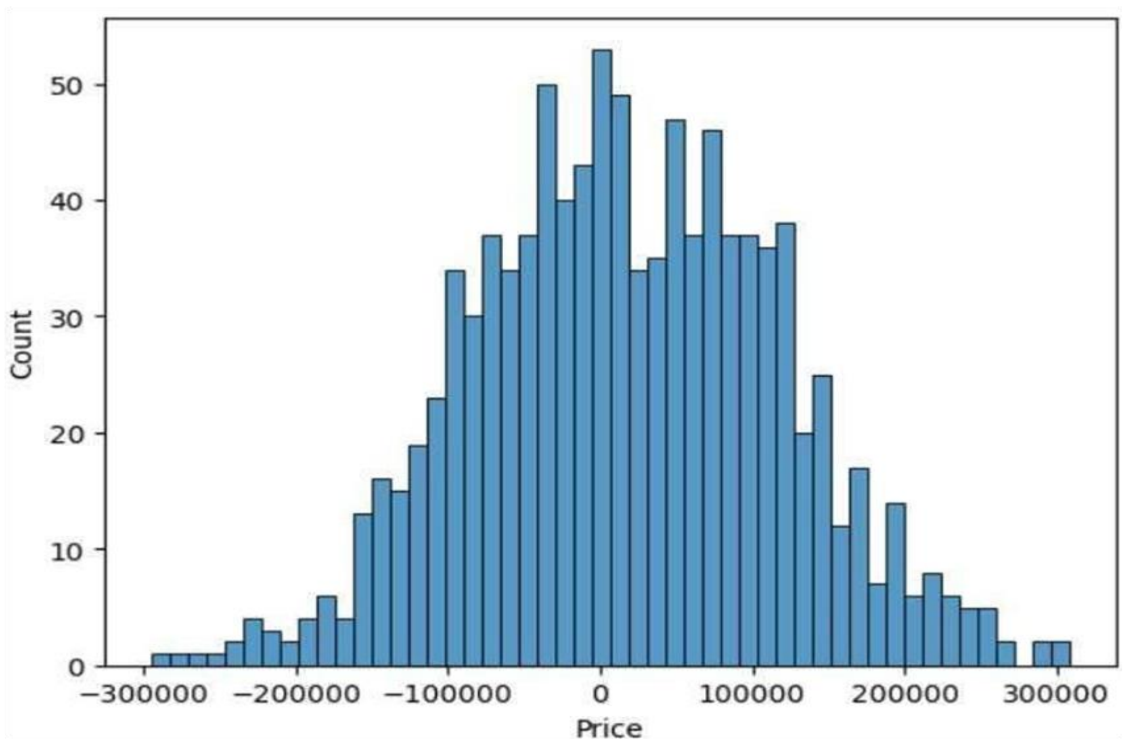
Prediction1)) print(mean_squared_error(Y_test, Prediction1))

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 :

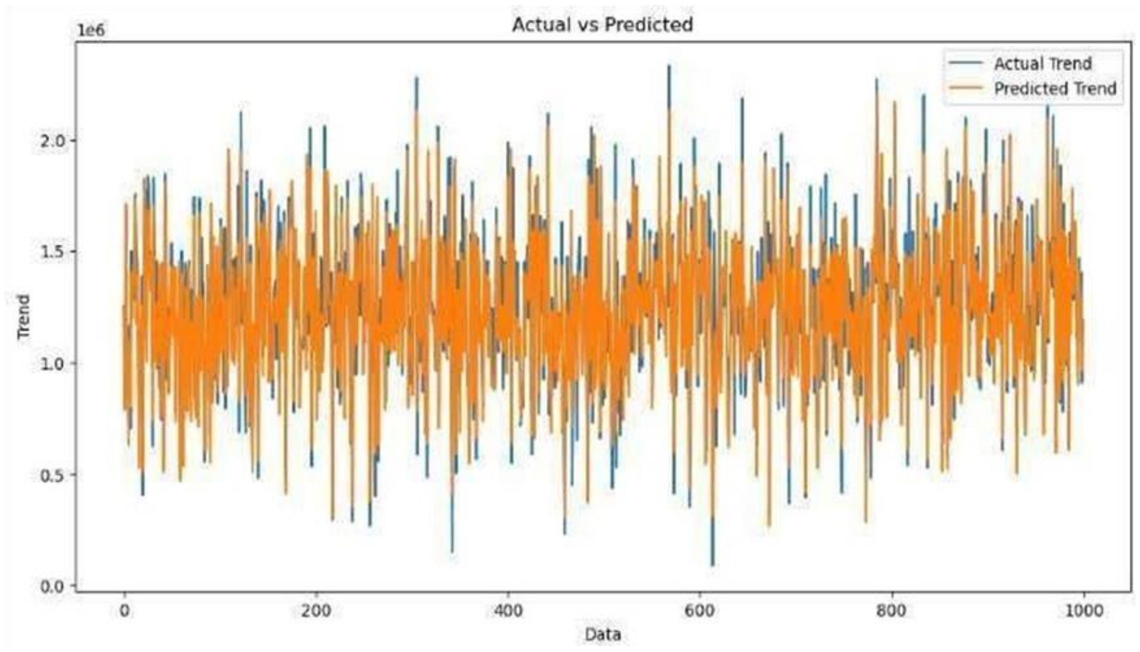
Prediction5 = model_xg.predict(X_test_scal) Evaluation of Predicting Prices:

Input:

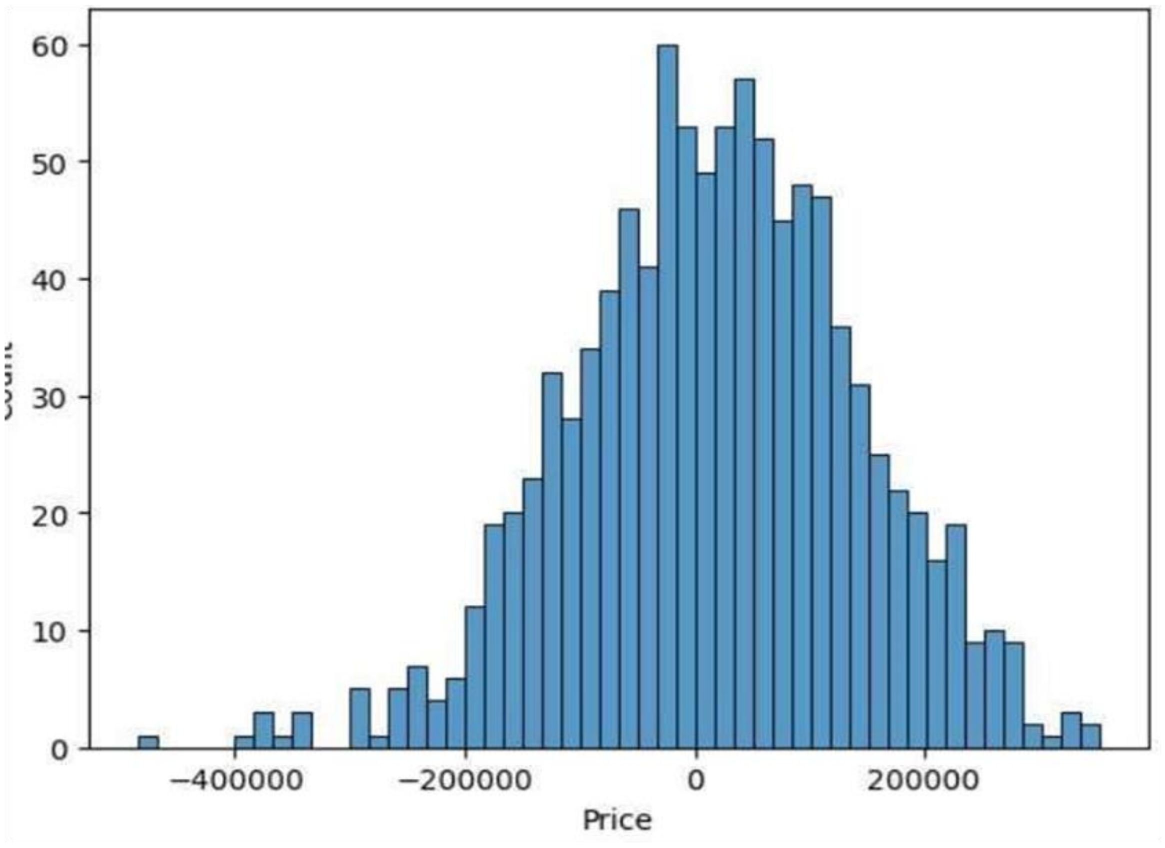
```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction5, label='Predicted Trend')
plt.xlabel('Data')                    plt.ylabel('Trend')              plt.legend()
```

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:

Predicting house prices using machine learning is a powerful and valuable tool in the real estate industry.through the utilization of advanced algorithms and vast datasets, it enables more accurate pricing estimations,benefiting both buyers and sellers.however,it’s essential yonremember thet no model is perfect ,and external factors can influence housing markets. Continous model refinement and expect human judgment are essential to supplement and enchance the predictions. Noneteless,machine learning has undoudtedly revolutionized the way we approach property valuation,making it more data-driven and precise.