Predicting House Prices Using Machine Learning

Problem Statement :-

The objective of this project is to develop a machine learning model that can accurately predict house prices based on various features and attributes. Predicting house prices is a valuable task in the real estate industry, as it can assist buyers and sellers in making informed decisions and provide insights for property investment.

The problem at hand is to develop a machine learning model that can accurately predict house prices based on a set of input features. This task involves creating a predictive regression model to estimate the market value of residential properties. The primary objective is to minimize prediction errors and provide a reliable tool for both buyers and sellers to make informed decisions in the real estate market.

The main problem we aim to address is to build a predictive model that can estimate the selling price of a house given a set of input features. These features may include but are not limited to:

1. Size of the house (in square feet)
2. Number of bedrooms
3. Number of rooms
4. Location or neighborhood
5. Age of the property
6. Proximity to schools, public transportation, and other amenities
7. Area of Population
8. Price

Key Challenges:

Several challenges are associated with predicting house prices:

1. Data Quality: Ensuring the quality and accuracy of the dataset is crucial. Incomplete or erroneous data can lead to unreliable predictions.

2. Feature Selection: Choosing the most relevant features that significantly impact house prices and avoiding overfitting or underfitting is essential.

3. Model Selection: Determining the most suitable machine learning algorithms and regression techniques for this specific prediction task.

4. Scalability: Ensuring that the model can handle a large and diverse dataset, which may include various types of housing properties.

5. Ethical Considerations: Addressing potential biases and ethical issues in predicting house prices, such as discrimination in housing.

Objectives:

The primary objectives of this project are as follows:

1. Collect and preprocess a comprehensive dataset of historical house prices and relevant features.

2. Explore and analyze the data to gain insights into the factors affecting house prices.

3. Select appropriate machine learning algorithms and regression techniques for building a predictive model.

4. Train and evaluate the model's performance using relevant metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

5. Deploy the model for practical use, allowing users to input house features and receive accurate price predictions.

6. Continuously monitor and update the model as new data becomes available, ensuring its long-term relevance and accuracy.

Design Thinking Process :-

Design thinking is a human-centered approach to problem-solving that can be applied to the task of predicting house prices using machine learning. Here's a simplified design thinking process tailored to this problem.

STEP 1: EMPATHIZE

Understand the needs and pain points of potential users, such as homebuyer, sellers, and real estate agents.Conduct interviews, surveys, or observations to gather insights into what information and features are most important when predicting the houseprices.

STEP 2: DEFINE

Clearly articulate the problem you are trying to solve, taking into account the user perspectives .Define the goals and success criteria for your machine learning model, such as prediction accuracy or user satisfaction.

STEP 3: IDEATE

Brainstorm potential features and data sources that could be valuable for predicting house prices. Consider different machine learning algorithms and techniques that can be applied to regression problems like this one.Explore creative ways to handle missing data and outliers.

STEP 4: PROTOTYPE

Create a prototype of your machine learning model. This could be simplified version using a subset of the data. Develop a user interface or application through which users can input house details and receive price predictions

STEP 5: TEST

Gather a small set of test users and let them interact with your prototype Collect feedback on the usability of your application and the usefulness of the predictions. Use this feedbackto refine your model and user interface

STEP 6: ITERATE

Based on user feedback and testing results, make iterative improvements to your machine learning model and the user interface. Continuously refine your feature selection, data preprocessing, and model tuning based on real-world performance.

STEP 7: IMPLEMENT

Once you have a robust machine learning model and a user-friendly interface, implement the full-scale application.Ensure that data is collected and updated regularly to keep the model up to date with the latest real estate market trends.

STEP 8: LAUNCH

Deploy the house price prediction tool to your target audience, whether homebuyers, sellers, or real estate professionals.Monitor its performance and collect real-world usage data.

STEP 9: FEEDBACK AND IMPROVE

Encourage users to provide feedback and ratings. Continuously analyze user feedback and application usage to identify areas for improvement.
Keep refining the model & user interface to meet changing user needs and market conditions.

STEP 10: SCALE

If successful, consider scaling your application to reach a larger audience.Explore opportunities for partnerships with real estate agencies or online property platforms.Throughout this design thinking process, the key is to remain user-centric, adapting your model and application based on user feedback and real-world usage.

Phase of development :-

This iterative approach ensures that your solution is not only technically sound but also valuable and user-friendly.
Predicting house prices using machine learning typically involves several steps in the problem-solving process.

Phase 1:

Problem Definition:

The problem of predicting house prices using machine learning for phase 1 is described. A house price prediction problem involves the task of forecasting or estimating the market value or selling price of a residential property. The goal is to create a model that can accurately predict the price of a house given its characteristics.

Design Thinking:

Applying design thinking principles to this problem provides a structured and user-centric approach that focuses on data analysis and encompasses the human aspects involved in buying, selling, and investing in real estate.

Data collection:

Gather a dataset containing historical housing data. This dataset should include features like square footage, number of bedrooms, location, etc., as well as the corresponding sale prices.

Data pre-processing:

Data pre-processing is important because it prepares the data in the most meaningful way for the subsequent detailed analysis. Data preprocessing holds significant importance in the realm of machine learning, particularly when predicting house prices. Additionally, we offer techniques developed from our experience in data processing.

In Phase 1, we describe the problem, our design thinking process, data collection methods, and data preprocessing techniques. This is the basics to solve the problem

Phase 2:

Innovation:

Our idea for predicting house prices has been successfully integrated in this phase. With this innovation, we are confident in our ability to accurately forecast the value of homes.

Implementation:

House price prediction implementation involves using machine learning algorithms to analyse and predict the value of houses based on various features such as location, size, number of bedrooms, amenities, and other relevant factors.

Features:

The features used in house price prediction are specific characteristics or attributes of a house that are taken into account when estimating its market value. These features help machine learning models make accurate predictions. Here are some common

In this phase, we generated and implemented our ideas. As a result, the actual problem is implemented based on our ideas arises. We implemented our ideas and features to solve the problem.

Phase 3:

Python code implementation is the crucial next step where our ideas and data will come to life. With the power of Python, we will bring our vision to reality and create impactful solutions.

Import dependencies:

Importing dependencies refers to bringing external libraries or modules into your code to utilize their functionalities. In the context of house price prediction using machine learning, several libraries are often employed in a programming environment like Python.

Loading Dataset:

The Python code was used to implement the collected data. A code for house price prediction needs development or a dataset can be obtained from an open platform.

Visualisation & Pre-processor:

In the context of house price prediction, data visualization, and preprocessing are crucial steps in understanding and preparing your dataset for machine learning.

Data Visualization:

Data visualization involves creating graphs, charts, and plots to represent the information in your dataset visually. In-house price prediction and data visualization serve several purposes: scatter plots, Histograms, etc.

In this phase, we developed a code and visualized the output. By investing time and effort into this phase, we were able to develop a robust code that accurately reflects our objectives. Furthermore, we also visualized the output, helping us to better understand and interpret our results. This critical step allowed us to identify any potential issues and make informed decisions moving forward.

Phase 4:

Model Training and Evaluation:

Assessing model performance and accuracy is paramount in house price prediction. Don't leave anything to chance, make sure to evaluate predicted data.

Model training in house price prediction involves several iterative steps to develop a robust and accurate predictive model. The evolution of the house price prediction model typically has several steps that have been implemented.

Random forest:

Random Forest is a well-known machine learning algorithm that falls under the category of supervised learning techniques. It can be effectively used for solving

both classification and regression problems in the field of Machine Learning. The algorithm predominantly relies on the concept of ensemble learning which is a technique of combining multiple classifiers to solve complex problems and to improve model performance.

Model Training:

Model training for house price prediction involves the process of building and fine-tuning a predictive model using a dataset with information about houses and their corresponding prices.

When using Linear Regression, Random Forest, and Decision Tree in Python, we obtain an output.

In this phase, we trained our data using multiple algorithms to obtain an output. By utilizing various algorithms, we have successfully trained our data to produce an output in this phase. Our diligent efforts have resulted in a successful outcome that has brought us one step closer to achieving our goal.

Describing the dataset:-

The dataset given for our project predicting house prices is https://www.kaggle.com/datasets/vedavyasv/usa-housing

Code:

```
df=pd.read_csv("/kaggle/input/usa-housing/USA_Housing.csv")
veri = df.copy()
veri.describe()
```

Output:

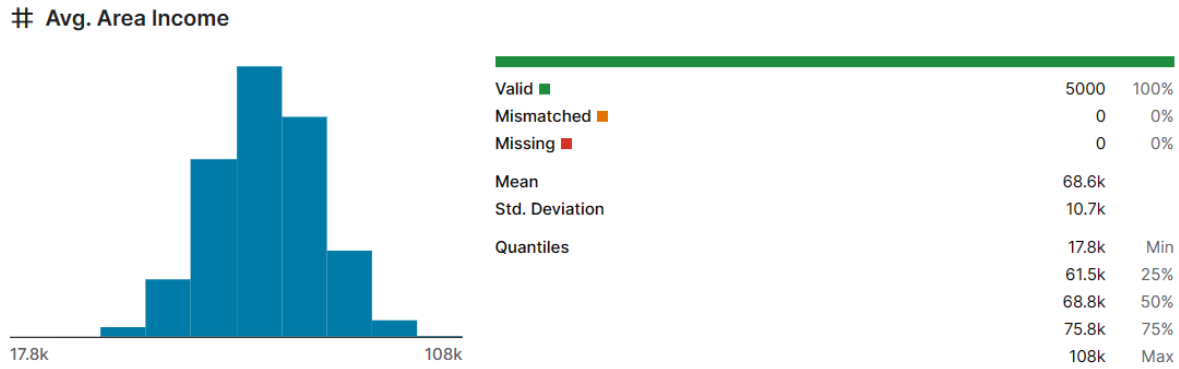
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 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

Describing the column in our dataset

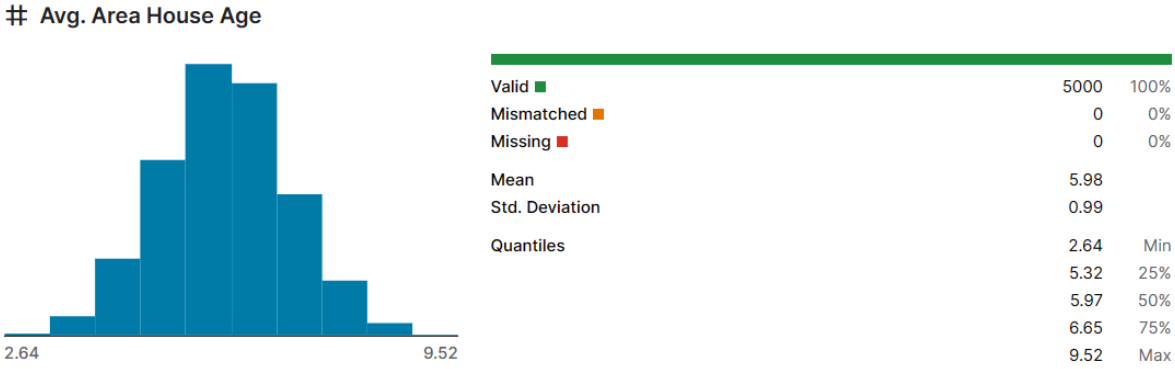
A. Average Area Income
B. Average Area House Age

C. Average Area Number of Rooms
D. Average Area Number of BedRooms
E. Area Population
F. Price
G. Address

A. Average Area Income

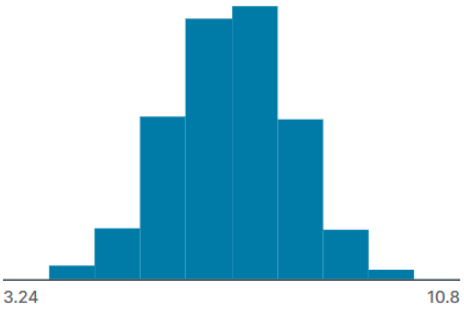


B. Average Area House Age



C.Average Area Number of Rooms

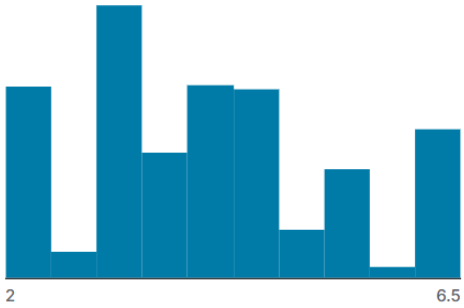
Avg. Area Number of Rooms



Valid	5000	100%
Mismatched	0	0%
Missing	0	0%
Mean	6.99	
Std. Deviation	1.01	
Quantiles	3.24	Min
	6.3	25%
	7	50%
	7.67	75%
	10.8	Max

D.Average Area Number of BedRooms

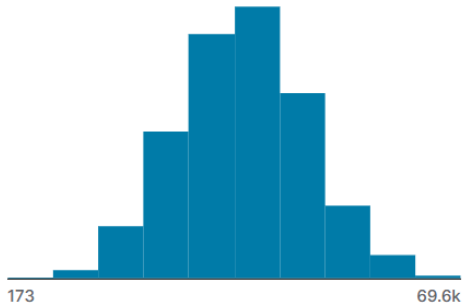
Avg. Area Number of Bedrooms



Valid	5000	100%
Mismatched	0	0%
Missing	0	0%
Mean	3.98	
Std. Deviation	1.23	
Quantiles	2	Min
	3.14	25%
	4.05	50%
	4.49	75%
	6.5	Max

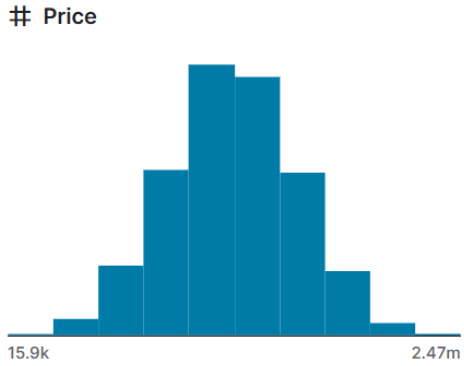
E. Area Population

Area Population



Valid	5000	100%
Mismatched	0	0%
Missing	0	0%
Mean	36.2k	
Std. Deviation	9.92k	
Quantiles	173	Min
	29.4k	25%
	36.2k	50%
	42.9k	75%
	69.6k	Max

F.Price



Valid	5000	100%
Mismatched	0	0%
Missing	0	0%
Mean	1.23m	
Std. Deviation	353k	
Quantiles	15.9k	Min
	998k	25%
	1.23m	50%
	1.47m	75%
	2.47m	Max

G. Address

Valid	5000	100%
Mismatched	0	0%
Missing	0	0%
Unique	5000	
Most Common	208 Michael...	0%

Data Pre-Processing Steps:-

Data pre-processing in Machine Learning is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. Data preprocessing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In simple words, data preprocessing in Machine Learning is a data mining technique that transforms raw data into an understandable and readable format.

The major tasks involved are Data cleaning, Data transformation, Data reduction, and Data integration

1.Data Cleaning:

Data cleaning, one of the major preprocessing steps in machine learning, locates and fixes errors or discrepancies in the data. From duplicates and outliers to missing numbers, it fixes them all. Methods like transformation, removal, and imputation help ML professionals perform data cleaning

seamlessly. One of the primary tasks in data cleaning is addressing missing values. Missing data can result from various factors, including human error, sensor malfunctions, or incomplete data collection.

The handling of missing values is essential to prevent them from negatively impacting analysis or model performance. Common approaches to address missing data include imputation, where missing values are filled in using statistical techniques or by making educated guesses, and deletion, where entire records or columns with missing data are removed.

2.Data Integration :

Data integration is among the major responsibilities of data preprocessing in machine learning. This process integrates (merges) information extracted from multiple sources to outline and create a single dataset. The fact that you need to handle data in multiple forms, formats, and semantics makes data integration a challenging task for many ML developers.Data integration can take different forms, such as batch processing, real-time integration, or stream processing. Batch processing involves periodic updates and consolidation of data, while real-time integration processes data as it arrives, allowing for up-to-the-minute insights. Stream processing takes real-time integration further by enabling the continuous processing of data as it flows into the system. The choice of integration method depends on the specific business needs and the nature of the data sources.

Regardless of the approach, effective data integration is essential for organizations to harness the full potential of their data and gain a competitive edge in today's data-driven landscape.

3.Data Transformation:

ML programmers must pay close attention to data transformation when it comes to data preprocessing steps. This process entails putting the data in a format that will allow for analysis. Normalization, standardization, and discretisation are common data transformation procedures.While standardization transforms data to have a zero mean and unit variance, normalization scales data to a common range. Continuous data is discretized into discrete categories using this technique.

4.Data Reduction:

Data reduction is the process of lowering the dataset's size while maintaining crucial information. Through the use of feature selection and feature extraction algorithms, data reduction can be accomplished. While feature extraction entails translating the data into a lower-dimensional space while keeping the crucial information, feature selection requires choosing a subset of pertinent characteristics from the dataset. One common approach to data reduction is dimensionality reduction, which is often used when dealing with high-dimensional datasets. Dimensionality reduction techniques like Principal Component Analysis (PCA) or t-Distributed Stochastic Neighbour Embedding (t-SNE) aim to reduce the number of features while preserving the most significant variance in the data. By eliminating less informative features, these methods make it easier to visualize and analyse data, as well as to train machine learning models without overfitting.

Steps in Data Pre-processing:

There are 7 significant steps in Data Pre-processing :

- Acquire the dataset
- Import all the crucial libraries
- Import the dataset
- Identifying and handling the missing values
- Encoding the categorical data
- Splitting the dataset
- Feature scaling

1.Acquire the Data set:

Acquiring the dataset is the first step in data preprocessing in machine learning. To build and develop Machine Learning models, you must first acquire the relevant dataset. This dataset will be comprised of data gathered from multiple and disparate sources which are then combined in a proper format to form a dataset. Dataset formats differ according to use cases. For instance, a business dataset will be entirely different from a medical dataset. While a business dataset will contain relevant industry and business data, a medical dataset will include healthcare-related data.

There are several online sources from where you can download datasets.The dataset that we work on is (https://www.kaggle.com/datasets/vedavyasv/usa-housing).

2.Import all the crucial libraries:

Since Python is the most extensively used and also the most preferred library by Data Scientists around the world, we'll show you how to import Python libraries for data preprocessing in Machine Learning. The predefined Python libraries can perform specific data preprocessing jobs. Importing all the crucial libraries is the second step in data preprocessing in machine learning. The three core Python libraries used for this data preprocessing in Machine Learning are

- NumPy – NumPy is the fundamental package for scientific calculation in Python. Hence, it is used for inserting any type of mathematical operation in the code. Using NumPy, you can also add large multidimensional arrays and matrices in your code.
- Pandas – Pandas is an excellent open-source Python library for data manipulation and analysis. It is extensively used for importing and managing the datasets. It packs in high-performance, easy-to-use data structures and data analysis tools for Python.
- Matplotlib – Matplotlib is a Python 2D plotting library that is used to plot any type of charts in Python. It can deliver publication-quality figures in numerous hard copy formats and interactive environments across platforms (IPython shells, Jupyter notebook, web application servers, etc.).

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

3. Import Dataset:

In this step, you need to import the datasets that you have gathered for the ML project at hand. Importing the dataset is one of the important steps in data preprocessing in machine learning. The most common format for machine learning data is CSV files. There are a number of ways to load a CSV file in Python.

Load CSV with Python Standard Library:

The Python API provides the module CSV and the function reader() that can be used to load CSV files. Once loaded, you convert the CSV data to a NumPy array and use it for machine learning. All fields are numeric and there is no header line. Running the recipe below will load the CSV file and convert it to a NumPy array.

```
# Load CSV (using python)
import csv
import numpy
filename = 'pima-indians-diabetes.data.csv'
raw_data = open(filename, 'rt')
reader = csv.reader(raw_data, delimiter=',', quoting=csv.QUOTE_NONE)
x = list(reader)
data = numpy.array(x).astype('float')
print(data.shape)
```

The example loads an object that can iterate over each row of the data and can easily be converted into a NumPy array. Running the example prints the shape of the array.

Load CSV File With NumPy:

You can load your CSV data using NumPy and the numpy.loadtxt() function. This function assumes no header row and all data has the same format.
The example below assumes that the file pima-indians-diabetes.data.csv is in your current working directory.

```
# Load CSV
import numpy
filename = 'pima-indians-diabetes.data.csv'
raw_data = open(filename, 'rt')
data = numpy.loadtxt(raw_data, delimiter=",")
print(data.shape)
```

Running the example will load the file as a numpy.ndarray and print the shape of the data.

```
(768, 9)
```

This example can be modified to load the same dataset directly from a URL as follows

Note: This example assumes you are using Python 3

```
# Load CSV from URL using NumPy
from numpy import loadtxt
from urllib.request import urlopen
url = 'https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv'
raw_data = urlopen(url)
dataset = loadtxt(raw_data, delimiter=",")
print(dataset.shape)
```

Again, running the example produces the same resulting shape of the data

```
(768, 9)
```

4. Identifying and handling the missing values:

Basically, there are two ways to handle missing data. In data preprocessing, it is pivotal to identify and correctly handle the missing values, failing to do this, you might draw inaccurate and faulty conclusions and inferences from the data. Needless to say, this will hamper your ML project.

Deleting a particular row – In this method, you remove a specific row that has a null value for a feature or a particular column where more than 75% of the values are missing. However, this method is not 100% efficient, and it is recommended that you use it only when the dataset has adequate samples. You must ensure that after deleting the data, there remains no addition of bias.

CODE:

```
df=ds.drop(['Address'],axis=1)
df.head()
```

OUTPUT:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05

Calculating the mean – This method is useful for features having numeric data like age, salary, year, etc. Here, you can calculate the mean, median, or mode of a particular feature or column or row that contains a missing value and replace the result for the missing value. This method can add variance to the dataset, and any loss of data can be efficiently negated. Hence, it yields better results compared to the first method (omission of rows/columns). Another way of approximation is through the deviation of neighbouring values. However, this works best for linear data.

CODE:

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))

print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

OUTPUT:

Mean Absolute Error: 97816.09821376632

Mean Squared Error: 14827659280.278809

Root Mean Squared Error: 121768.87648442358

5. Encoding the categorical data:

Categorical data refers to the information that has specific categories within the dataset. In the dataset cited above, there are two categorical variables – country and purchased.Machine Learning models are primarily based on mathematical equations. Thus, you can intuitively understand that keeping the categorical data in the equation will cause certain issues since you would only need numbers in the equations.

CODE:

```
labelencoder_y= LabelEncoder()

y= labelencoder_y.fit_transform(y)
```

OUTPUT:

```
array([0, 1, 0, 0, 1, 1, 0, 1, 0, 1]
```

6. Splitting the dataset

Splitting the dataset is the next step in data preprocessing in machine learning. Every dataset for Machine Learning model must be split into two separate sets – training set and test set.

CODE:

```
from sklearn.model_selection import train_test_split,cross_val_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42)
```

7. Feature scaling:

Feature scaling marks the end of the data preprocessing in Machine Learning. It is a method to standardize the independent variables of a dataset within a specific range. In other words, feature scaling limits the range of variables so that you can compare them

on common grounds.In the dataset, you can notice that the age and salary columns do not have the same scale.

In such a scenario, if you compute any two values from the age and salary columns, the salary values will dominate the age values and deliver incorrect results. Thus, you must remove this issue by performing feature scaling for Machine Learning.

Most ML models are based on Euclidean Distance, which is represented as:

$$d(A,B) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

CODE:

```
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Feature Extration Technique:-

When predicting house prices using machine learning, feature extraction is a crucial step to select and transform relevant information from your dataset. Here are some feature extraction techniques you can consider:

1. Feature Selection: Choose a subset of the most relevant features. You can use techniques like correlation analysis, mutual information, or feature importance from tree-based models to select the most important features.

2. Feature Engineering:

- Numerical Features:
  Create new features like total square footage, price per square foot, or age of the property.
  Handle missing values by imputing with means, medians, or zeros.

Scale numerical features using techniques like Min-Max scaling or Standardization.

- Categorical Features:
  One-hot encoding or label encoding for categorical variables.
  Create binary features like "has_pool," "is_garage_attached," etc.

- Temporal Features:
  Extract features like year built, year renovated, or time on market from date variables.

- Geospatial Features:
  Extract location-based features like distance to the nearest school, hospital, or city center.

3. Text Features (if you have property descriptions):
Use natural language processing (NLP) techniques to extract meaningful information from textual descriptions. You can use techniques like TF-IDF, word embeddings (Word2Vec, GloVe), or pretrained language models (e.g., BERT) to convert text into numerical features.

4. Principal Component Analysis (PCA):
Reduce dimensionality while retaining important information. PCA can be useful when you have a large number of correlated features.

5. Feature Scaling:
Ensure all features are on the same scale to prevent some features from dominating the learning process. Common techniques include Min-Max scaling and Z-score normalization.

6. Domain-Specific Feature Extraction:
In real estate, there may be specific domain knowledge that can guide feature extraction. For example, you might extract features related to neighborhood characteristics, school ratings, crime rates, or proximity to public transportation.

7.Recursive Feature Elimination (RFE):
Use this technique in combination with a model (e.g., linear regression) to iteratively eliminate the least significant features until you reach a desired number of features.

8. Feature Aggregation:

Combine features in meaningful ways. For example, you can create an "overall quality" feature by aggregating features related to the quality of different components of the house.

9. Cross-Validation:

Use cross-validation techniques to validate the effectiveness of your feature extraction methods and model performance.

Remember that the choice of feature extraction techniques should be guided by your dataset and problem.

Experiment with different methods to find the combination that results in the best predictive performance for house price prediction.

Machine Learning Algorithm :-

Our choice of machine learning algorithm is Random forest

The Random Forest was found to consistently perform better than the k- NN algorithm in terms of smaller errors and be better suited as a prediction model for the house price problem.

Random Forest:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset."

Model Training:-

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list
files in the input directory
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import explained_variance_score
from sklearn.metrics import confusion_matrix
import os
print(os.listdir("../input"))
import warnings
warnings.filterwarnings('ignore')


# Any results you write to the current directory are saved as output.
['kc_house_data.csv']



# X(Independent variables) and y(target variables)
X = dataset.iloc[:,1:].values
y = dataset.iloc[:,0].values
```

In [30]:

```
#Splitting the data into train,test data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0
)
```

Multiple Linear Regression:

```
mlr = LinearRegression()
mlr.fit(X_train,y_train)
mlr_score = mlr.score(X_test,y_test)
pred_mlr = mlr.predict(X_test)
expl_mlr = explained_variance_score(pred_mlr,y_test)
```

Decision Tree

```
tr_regressor = DecisionTreeRegressor(random_state=0)
tr_regressor.fit(X_train,y_train)
tr_regressor.score(X_test,y_test)
pred_tr = tr_regressor.predict(X_test)
decision_score=tr_regressor.score(X_test,y_test)
expl_tr = explained_variance_score(pred_tr,y_test)
```

Random Forest Regression Model

```
rf_regressor = RandomForestRegressor(n_estimators=28,random_state=0)
rf_regressor.fit(X_train,y_train)
rf_regressor.score(X_test,y_test)
rf_pred =rf_regressor.predict(X_test)
rf_score=rf_regressor.score(X_test,y_test)
expl_rf = explained_variance_score(rf_pred,y_test)
```

Calculate Model Score

Let's calculate the model score to understand how our model performed along with the explained variance score.

```
print("Multiple Linear Regression Model Score is ",round(mlr.score(X_test,y_test)*100))
print("Decision tree  Regression Model Score is ",round(tr_regressor.score(X_test,y_test)*100))

print("Random Forest Regression Model Score is ",round(rf_regressor.score(X_test,y_test)*100))

#Let's have a tabular pandas data frame, for a clear comparison

models_score =pd.DataFrame({'Model':['Multiple Linear Regression','Decision Tree','Random forest Regression'],'Score':[mlr_score,decision_score,rf_score], ' Explained Variance Score':[expl_mlr,expl_tr,expl_rf]})
models_score.sort_values(by='Score',ascending=False)
```

Output:

```
Multiple Linear Regression Model Score is  69.0
Decision tree  Regression Model Score is  75.0
Random Forest Regression Model Score is  88.0
```

	Model	Score	Explained Variance Score
2	Random forest Regression	0.88011	0.846248
1	Decision Tree	0.74962	0.730713
0	Multiple Linear Regression	0.68779	0.527528

Random forest:

```
y = data['Price']
X = data.drop(['Price', 'Address'], axis = 1)
# Random Forest
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
train_X, val_X, train_y, val_y = train_test_split(X, y, random_state = 42)
model = RandomForestRegressor(random_state = 1)
model.fit(train_X, train_y)
preds = model.predict(val_X)
print("MAE: ", mean_absolute_error(preds, val_y))
print("RMSE: ", np.sqrt(mean_squared_error(preds, val_y)))
```

output:

MAE: 93812.37073246129
RMSE: 118380.48325186648

Evaluation :-

```
#using RandomForestRegressor
```

- ```
  from sklearn.ensemble import RandomForestRegressor
  rf_reg=RandomForestRegressor(n_estimators=1000)
  rf_reg.fit(x_train,y_train)

  test_pred=rf_reg.predict(x_test)
  train_pred=rf_reg.predict(x_train)

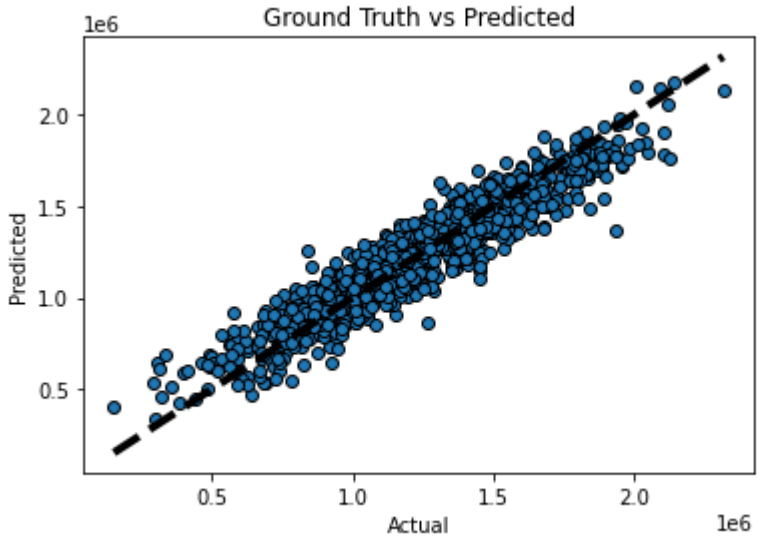
  print('Test set evaluation:\n_____')
  print_evaluate(y_test,test_pred)
  ```
  ```

```
print('Train set evaluation:\n_____')
print_evaluate(y_train,train_pred)
```

Output:

```
Test set evaluation:
_____
MAE:  94027.38848972939
MSE:  14117343785.892136
RMSE:  118816.42893931856
R2 Square 0.8803719599235804
_____
Train set evaluation:
_____
MAE:  35362.54528067692
MSE:  1989422953.623089
RMSE:  44602.94781315568
R2 Square 0.9843930758497742
_____
```

```
#visualizing the predicted value
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, edgecolors=(0, 0, 0))
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Ground Truth vs Predicted")
plt.show()
```



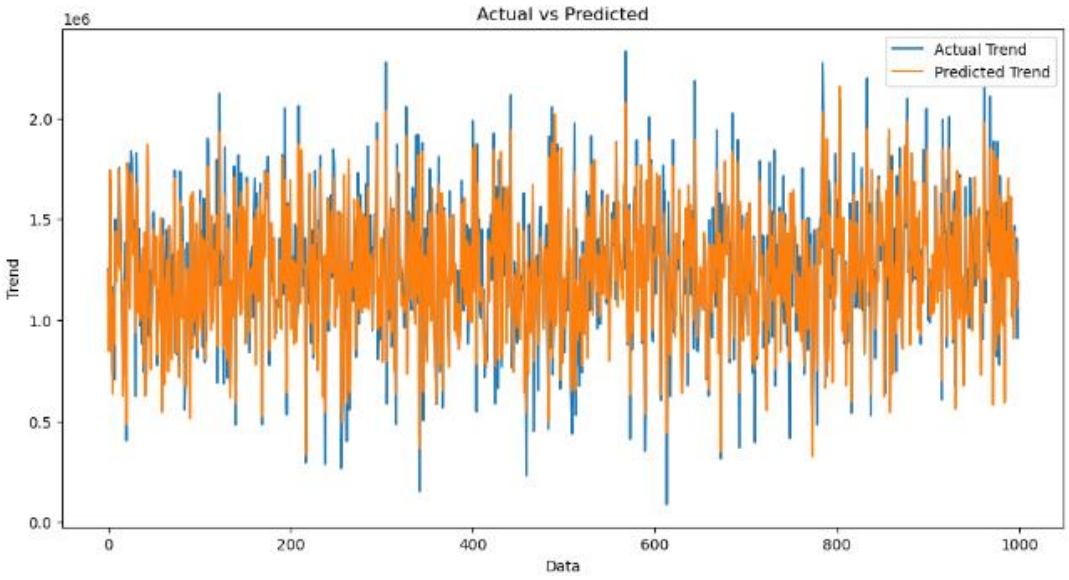

```
model_rf = RandomForestRegressor(n_estimators=50)
model_rf.fit(X_train_scal, Y_train)
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

Predicting Prices:

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

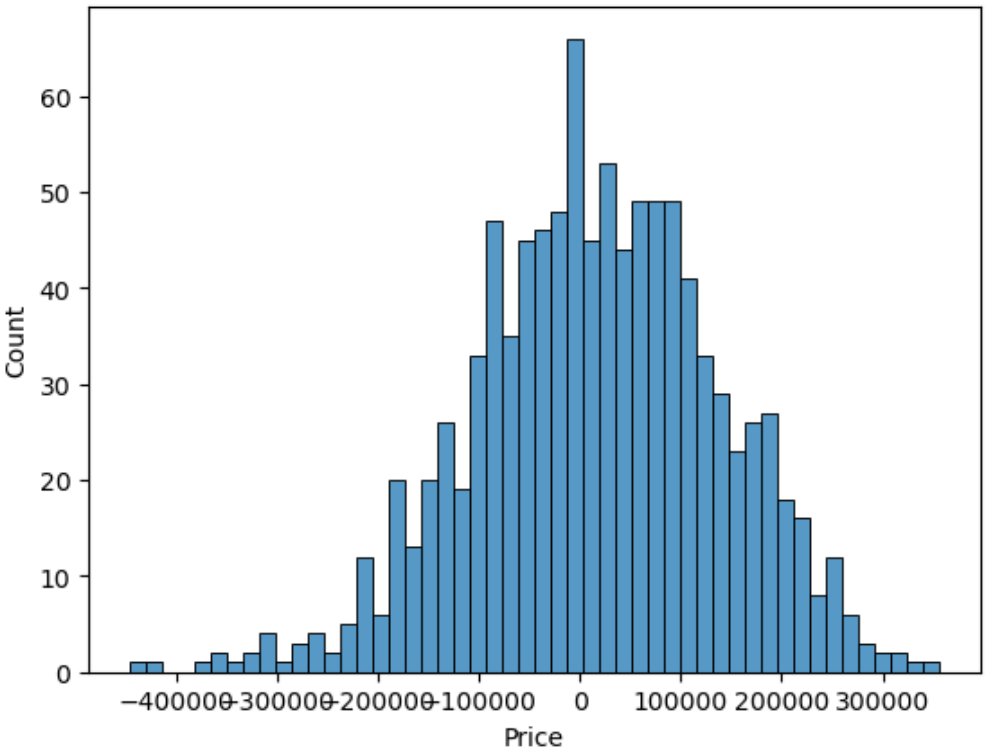
Evaluation of Predicted Data :

```
    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))
```

output:

-0.0006222175925689744

286137.81086908665

128209033251.4034

Innovation :

Certainly Predicting house prices using machine learning can be a valuableapplication. Here are some innovative ideas and approaches:

- Time Series Analysis: Incorporate time series data, such as historicalhouse price trends, economic indicators, and seasonality, to improvepredictions.

- Natural Language Processing (NLP): Analyse real estate listings,descriptions, and reviews to extract sentiment and neighbourhoodfeatures that can impact prices.

- Image Analysis: Use computer vision to analyse property images to extract features like curb appeal, interior design or outdoor amenities thatinfluence prices.

- Geospatial Data: Utilize geographic data like proximity to schools, parks,public transportation, and crime rates to enhance predictions.

- DeepLearning: Implement deep learning models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for more accurate predictions based on complex data.

- Ensemble Methods: Combine multiple machine learning models, such as Random Forests, Gradient Boosting, and Neural Networks to improve prediction accuracy.

- Feature Engineering: Create new features like walkability scores, commute times, or neighbourhood safety indices to capture important aspects of location.

- Transfer Learning: Apply pre-trained models (e.g., BERT for NLP tasks)to extract features from textual data related to properties.

- Anomaly Detection: Identify unusual patterns or outliers in house price data to detect potential investment opportunities or areas for further investigation.

- Interactive Web Applications: Develop user-friendly web apps or mobile apps that allow users to input property characteristics and receive instantprice predictions.

- Blockchain and Smart Contracts: Explore the use of blockchain technology for transparent and secure property transactions, which couldimpact pricing models.

- Predicting Future Value: Rather than just predicting current prices,

createmodels that forecast how property values might change over time,
helping investors make long-term decision.

- Data Fusion: Combine various data sources, including social media sentiment, local news articles, and economic forecasts, to build a comprehensive prediction model.

- Explainable AI: Ensure your model provides interpretable explanationsfor its predictions, which can be crucial for real estate professionals.

- Market Segmentation: Develop models that segment the housing marketinto different buyers categories (e.g., luxury, affordable, starter homes) for more targeted predictions.

- Collaborative Filtering: Apply collaborative filtering techniques,similar to recommendation systems, to predict house prices based on user preferences and Behaviour.

- Energy Efficiency: Incorporate energy efficiency ratings and potentialcost savings into price predictions, as green features become more important to buyers.

- Legal and Regulatory Analysis: Consider how changes in local zoninglaws, tax policies, or regulations might impact house prices.

- Crowdsourced Data: Use data from platforms like Zillow, Redfin, or Airbnb to supplement your dataset and improve prediction accuracy.

Remember that innovation in this field often requires a deep understanding of both machine learning techniques and the real estate market, as well as access to diverse and high-quality data source