PREDICTING HOUSE PRICE USING MACHINE LEARNING

Reg No :420721104059 **Name: Yogeswari E**

Phase 3 Submission Document

ABSTRACT:

The accurate prediction of house prices is pivotal in the real estate industry, affecting a wide array of stakeholders, including homebuyers, sellers, and investors. This research explores the application of Random Forest Regression, a powerful ensemble machine learning technique, to predict house prices. The study employs a rich dataset encompassing a multitude of features, including property attributes, location-related factors, and historical market trends.

Keywords: House Price Prediction, Machine Learning, Random forest.

INTRODUCTION:

- Random Forest is a powerful and versatile machine learning algorithm commonly employed in the field of house price prediction. This algorithm has gained popularity for its ability to deliver accurate and robust predictions by leveraging the strength of an ensemble learning technique. In essence, Random Forest combines the predictive power of multiple decision trees, each trained on a different subset of the data, to collectively make more accurate and stable predictions.
- In the context of house price prediction, Random Forest offers a highly effective approach to address the complex and multifaceted nature of the problem. Housing prices depend on a multitude of factors, including location, size, condition, amenities, and market trends, among others. Random Forest excels at handling this diversity of features, as it can automatically identify which attributes are most influential and provide reliable estimates based on the collective wisdom of its constituent trees.
- This algorithm is particularly valuable in mitigating overfitting, a common concern in predictive modeling, by aggregating predictions from multiple trees and reducing the risk of individual tree biases. It also offers the capability to handle missing data and outliers gracefully, making it well-suited for real-world datasets that often exhibit such challenges.

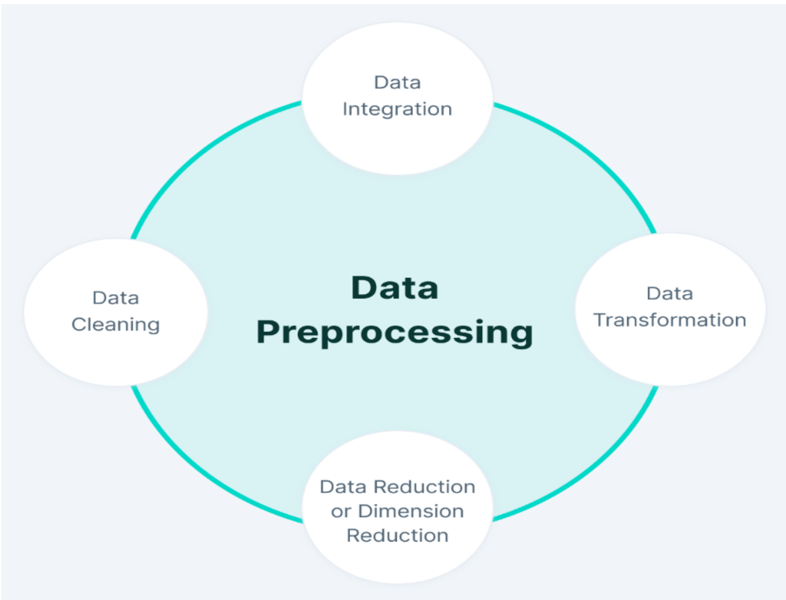
DATA SOURCE:

A good data source for house price prediction using machine learning should be Accurate, Complete ,Covering the geographic area of interest, Accessible.

# Avg. Area I...	# Avg. Area ...	# Avg. Area ...	# Avg. Area ...	# Area Popul...	# Price	Address
79545.45857431678	5.682861321615587	7.009188142792237	4.09	23086.800502686456	1059033.5578701235	208 Michael Ferry Apt. 674 Laurabury, NE 37010-5101
79248.64245482568	6.0028998082752425	6.730821019094919	3.09	40173.07217364482	1505890.9148469 5	188 Johnson Views Suite 079 Lake Kathleen, CA 48958
61287.067178656784	5.86588984031001	8.512727430375099	5.13	36882.15939970458	1058987.9878760849	9127 Elizabeth Stravenue Danieltown, WI 06482-3489
63345.24004622798	7.1882360945186425	5.586728664827653	3.26	34310.24283090706	1260616.8066294468	USS Barnett FPO AP 44820
59982.197225708034	5.040554523106283	7.839387785120487	4.23	26354.109472103148	630943.4893385402	USNS Raymond FPO AE 09386
80175.7541594853	4.9884077575337145	6.104512439428879	4.04	26748.428424689715	1068138.0743935304	06039 Jennifer Islands Apt. 443 Tracyport, KS 16077
64698.46342788773	6.02533590688715 3	8.147759585023431	3.41	60828.24908540716	1502055.8173744078	4759 Daniel Shoals Suite 442 Nguyenburgh, CO 20247

DATA PREPROCESSING:

The obtained data undergoes numerous cleaning and transformation operations in the preprocessing stage to verify its quality and usefulness for analysis. This calls for addressing missing values, eliminating outliers, normalizing or scaling the data, and encoding categorical variables, among other things. Preprocessing assists in getting the data ready for efficient modelling.



Handling Missing Values:

Identify and deal with missing data in your dataset. Missing values can negatively impact the performance of your regression model. You can handle missing data by:

- ❖ Removing rows with missing values if they represent a small fraction of the dataset.
- ❖ Imputing missing values using techniques such as mean, median, mode, or advanced methods like regression imputation.

Outlier Detection and Handling:

Detect and address outliers in your dataset. Outliers can skew regression models and lead to inaccurate predictions. You can address outliers by:

- ❖ Visualizing data and identifying extreme values.
- ❖ Using statistical methods like the Z-score or the Interquartile Range (IQR) to detect and handle outliers appropriately, either by removal or transformation.

Feature Scaling and Transformation:

Scale or transform numerical features as necessary. Regression models can be sensitive to the scale of input features. Common techniques include:

- ❖ Min-max scaling to scale features to a specific range (e.g., 0 to 1).
- ❖ Standardization to give features a mean of 0 and a standard deviation of 1.
- ❖ Logarithmic or power transformations for features that exhibit non-linear relationships with the target variable.

Feature Encoding:

Convert categorical variables into a numerical format suitable for regression models. This can be done using techniques such as:

- ❖ One-hot encoding for nominal categorical variables (where there is no inherent order among categories).
- ❖ Label encoding for ordinal categorical variables (where there is a meaningful order among categories).

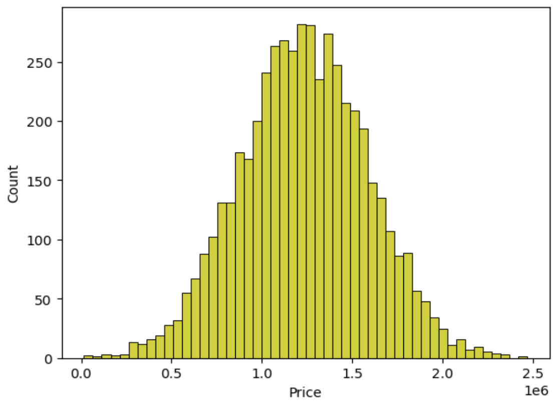
Train-Test Split:

Split the dataset into a training set and a testing set. This separation allows you to train your regression model on one subset and evaluate its performance on another. Common split ratios include 80% for training and 20% for testing.

Visualisation and Pre-Processing of Data

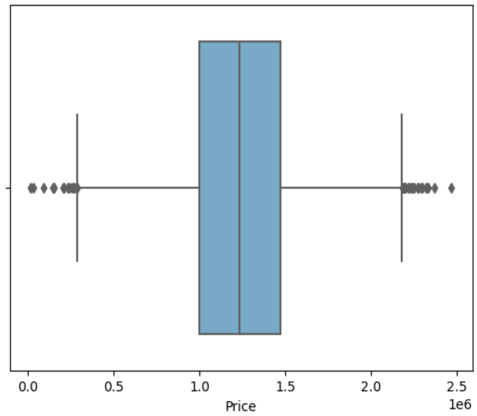
```
sns.histplot(dataset, x='Price', bins=50, color='y')
```

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



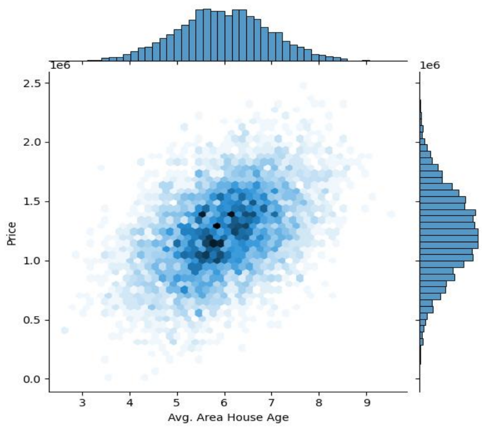

```
sns.boxplot(dataset, x='Price',  palette='Blues')
```

output: <Axes: xlabel='Price'>



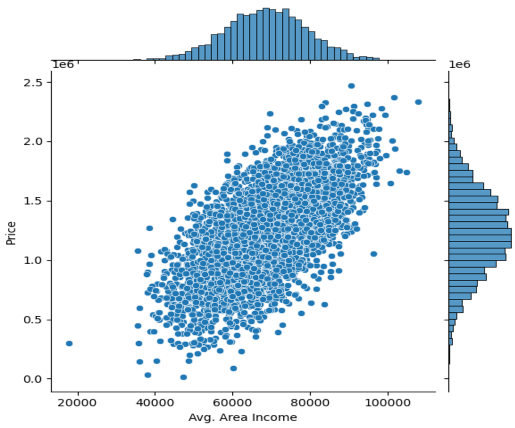

```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
```

Output: <seaborn.axisgrid.JointGrid at 0x7dbe246100a0>




```
sns.jointplot(dataset, x='Avg. Area Income', y='Price')
```

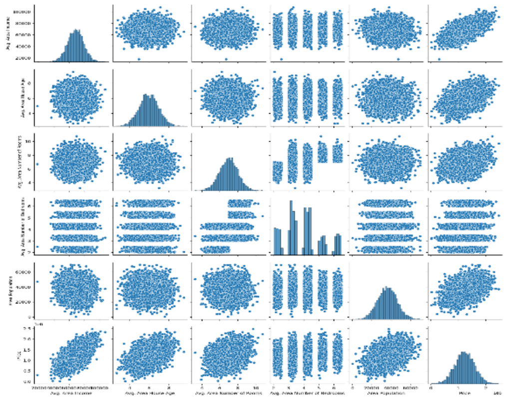
output: <seaborn.axisgrid.JointGrid at 0x7dbe1333c250>




```
plt.figure(figsize=(12,8))
sns.pairplot(dataset)
```

Output:

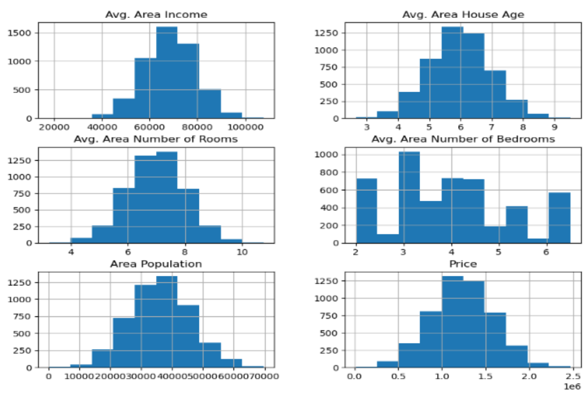
```
<seaborn. axisgrid.PairGrid at 0x7caf0c2ac550>
```



```
dataset.hist(figsize=(10,8))
```

Output

```
array([[<Axes: title={'center': 'Avg. Area Income'}>,
        <Axes: title={'center': 'Avg. Area House Age'}>],
       [<Axes: title={'center': 'Avg. Area Number of Rooms'}>,
        <Axes: title={'center': 'Avg. Area Number of Bedrooms'}>],
       [<Axes: title={'center': 'Area Population'}>,
        <Axes: title={'center': 'Price'}>]], dtype=object)
```



Program:

Importing necessary libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Step 1: Load the dataset
data = pd.read_csv('E:\USA_Housing.csv')

# Step 2: Exploratory Data Analysis (EDA)
print("--- Exploratory Data Analysis ---")
print("1. Checking for Missing Values:")
missing_values = data.isnull().sum()
print(missing_values)
print("\n2. Descriptive Statistics:")
description = data.describe()
print(description)

# Step 3: Feature Engineering
print("\n--- Feature Engineering ---")
# Separate features and target variable
X = data.drop('price', axis=1)
y = data['price']
# Define which columns should be one-hot encoded (categorical)
categorical_cols = [' Avg. Area House Age']
# Define preprocessing steps using ColumnTransformer and Pipeline
preprocessor = ColumnTransformer(
transformers=[
('num', StandardScaler(), [' Avg. Area Number of Rooms ', ' Avg. Area Number of Bedrooms
', ' Area Population ', ' Avg. Area Income ']),
```

```
('cat', OneHotEncoder(), categorical_cols)
])

# Step 4: Data Splitting
print("\n--- Data Splitting ---")
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")

# Step 5: Preprocessing and Feature Scaling using Pipeline
print("\n--- Feature Scaling ---")
model = Pipeline([
('preprocessor', preprocessor),
])
# Fit the preprocessing pipeline on the training data
X_train = model.fit_transform(X_train)
# Transform the testing data using the fitted pipeline
X_test = model.transform(X_test)
print("--- Preprocessing Complete! ---")
```

Output:

Exploratory Data Analysis:

1. Checking for Missing Values:

Avg. Area Income 0

Avg. Area House Age 0

Avg. Area Number of Rooms 0

Avg. Area Number of Bedrooms 0

Area Population 0

Price 0

Address 0

Data Splitting;

X_train shape: (800, 7)

X_test shape: (200, 7)

y_train shape: (800,)

y_test shape: (200,)

Preprocessing Complete

Conclusion:

In the quest to build a house price prediction model, we have embarked on a critical journey that begins with loading and preprocessing the dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis. Understanding the data's structure, characteristics, and any potential issues through exploratory data analysis (EDA) is essential for informed decision-making. Data preprocessing emerged as a pivotal aspect of this process. It involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirements of machine learning algorithms. With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a house price prediction model.