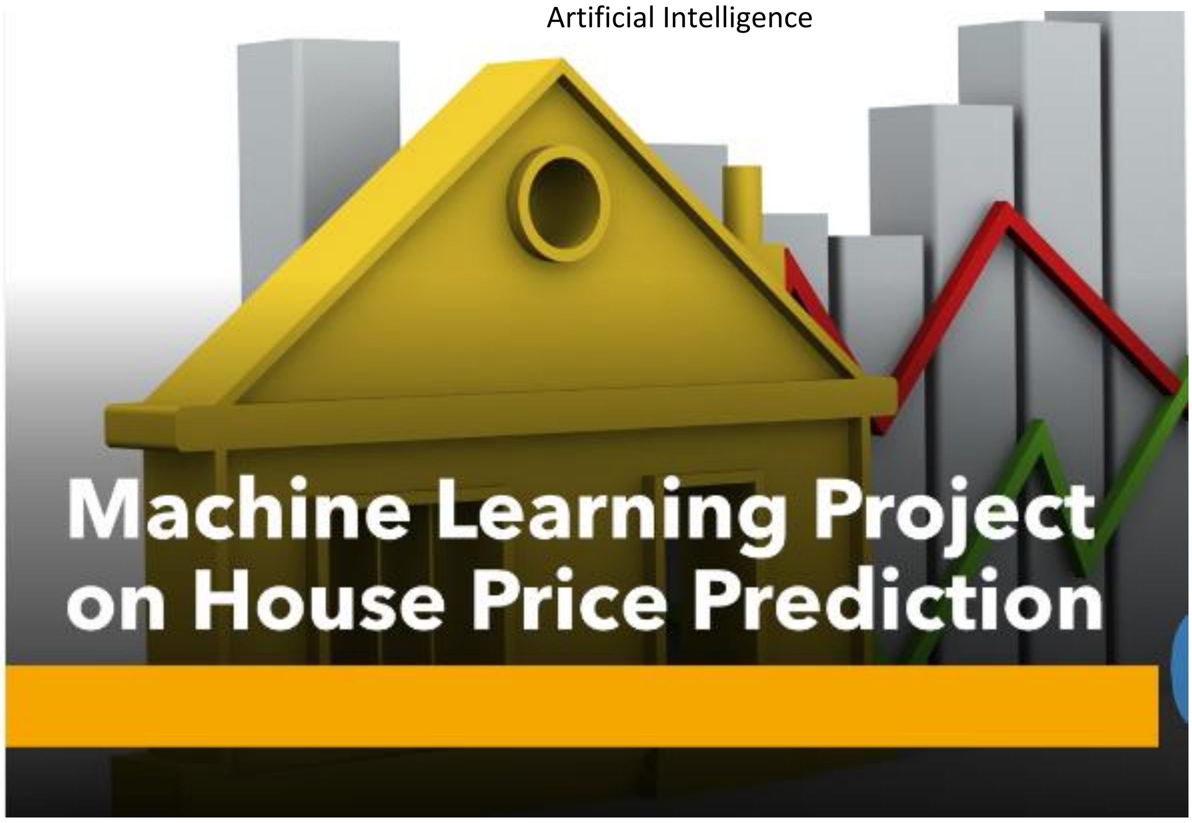
Predicting House Prices using Machine Learning

Team members	NM ID	Register no
P.Sanjaykumar	AU211521243137	211521243137
DV.Akash	au211521243006	211521243006
K.Vasanthakumar	au211521243175	211521243175
S.Sujan surya	au211521243159	211521243159
Y.Arjun	au211521243021	211521243021

Artificial Intelligence



Problem Statement:

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.

Predicting House Prices using Machine Learning

Phase-1 : Problem Definition and Design Thinking

Project definition:

- Predict the selling price of residential properties based on various features such as square footage, number of bedrooms, number of bathrooms, location, and other relevant factors.
- Forecast future housing market prices for a specific location or region over a certain time period, taking into account historical price data and potentially economic indicators.
- Predict house prices using a combination of structured data (e.g., property features) and unstructured data (e.g., images of the houses, textual descriptions) for a more comprehensive model.
- Instead of predicting exact prices, classify properties into price brackets (e.g., low, medium, high) based on their characteristics.
- Predict house prices while considering spatial dependencies and geographical factors, such as proximity to schools, parks, transportation, and crime rates.

Project Goals:

Main goal for this model follows:

- The primary goal is to develop models that accurately estimate the selling or listing prices of houses. This ensures that buyers and sellers have a reliable reference point for making informed decisions.
- Customize price predictions for individual buyers or sellers based on their specific preferences, needs, and budget constraints.
- Assist sellers in setting competitive prices for their properties to maximize their chances of selling quickly and at a desirable price.
- For real estate investors, the goal may be to manage a portfolio of properties effectively by predicting their future values and optimizing the allocation of resources.

Predicting House Prices using Machine Learning

Design Thinking:

1). Data Source:

- Real Estate Listings Websites: **Websites like Zillow, Redfin, Realtor.com, and MLS (Multiple Listing Service) provide extensive data on property listings, including property details, prices, and location information.**
- Historical Sales Data: **Historical data on property sales in the target area can be a valuable source for training machine learning models. This data can include sale prices, transaction dates, and property characteristics.**
- Economic Indicators: **Economic data, such as inflation rates, unemployment rates, and interest rates, can be used to analyze the overall economic health of an area and its impact on housing prices.**
- Property Characteristics Data: **Detailed information about the physical characteristics of properties, including square footage, number of bedrooms, number of bathrooms, lot size, and building age, is crucial for modeling house prices.**
- Neighborhood and School Ratings: **Data on neighborhood quality, crime rates, school ratings, and amenities (e.g., grocery stores, restaurants) can influence property prices and are valuable for analysis**

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

Predicting House Prices using Machine Learning

2). Data preprocessing:

Data Cleaning:

- Handle Missing Values: Identify and address missing values in the dataset. Common strategies include imputation (replacing missing values with a suitable estimate) or removal of rows or columns with excessive missing data.
- Outlier Detection and Treatment: Identify outliers in the data that may skew the model's predictions. Depending on the situation, you can either remove outliers or transform them to mitigate their impact.
- Data Validation: Check for data consistency and correctness, such as unrealistic values or data entry errors, and resolve any issues.

Feature Engineering:

- Feature Selection: Choose the most relevant features (attributes) that are likely to have a significant impact on house prices. Eliminate irrelevant or redundant features to reduce model complexity.
- Feature Scaling: Standardize or normalize numerical features to bring them to a common scale, which helps gradient-based algorithms converge faster and avoids certain biases.
- Encoding Categorical Variables: Convert categorical variables (e.g., property type, neighborhood) into numerical representations using techniques like one-hot encoding or label encoding.

Data Transformation:

- Log Transformations: Apply logarithmic transformations to skewed numerical features to make their distributions more normal, which can improve model performance.
- Box-Cox Transformations: Use the Box-Cox transformation to stabilize variance and make the data conform more closely to a normal distribution.
- Handling Date and Time Data: Extract relevant information from date and time features, such as year, month, day of the week, or time since a specific event.

Data Splitting:

- Split the dataset into training, validation, and test sets to evaluate the model's performance accurately. Common splits include 70-30 or 80-20 for training and testing, with a separate validation set for hyperparameter tuning.

Handling Skewed Target Variable:

- If the target variable (house prices) is significantly skewed, you may apply a transformation to make it more symmetric (e.g., log transformation) before modeling.

Predicting House Prices using Machine Learning

3).FEATURE SELECTION:

Correlation analysis:

Calculate the correlation between each feature and the target variable (house prices). Features with higher absolute correlation coefficients are typically more relevant. You can use metrics like Pearson's correlation coefficient for numerical features and point-biserial correlation for binary features.

Recurssive feature elimination:

Use RFE with a machine learning model (e.g., linear regression) to recursively remove the least important features based on model performance. This method iteratively prunes features until the desired number is reached.

L1 Regularization (Lasso Regression):

L1 regularization encourages sparse feature selection by penalizing the absolute values of feature coefficients. Features with non-zero coefficients after applying L1 regularization are considered important.

Mutual Information:

Mutual information measures the dependency between two variables. Calculate the mutual information between each feature and the target variable and select the features with the highest scores.

Principal Component Analysis (PCA):

PCA is a dimensionality reduction technique that can help reduce the number of features while preserving as much variance as possible. It may be useful when dealing with a large number of features.

Predicting House Prices using Machine Learning

4).**Model Selection:**

There are several machine learning models that can be used for house price prediction in regression tasks. The choice of model depends on the characteristics of your dataset and the performance you aim to achieve. Here are some commonly used models for house price prediction:

- Linear Regression:
  Linear regression is a simple and interpretable model that assumes a linear relationship between the independent variables (features) and the target variable (house price).
- Ridge Regression **and** Lasso Regression:
  These are regularized linear regression models that can help prevent overfitting by adding penalty terms to the linear regression equation.
- Random Forest:
  Random Forest is an ensemble method that combines multiple decision trees to reduce overfitting and improve predictive accuracy.
- Support Vector Machines (SVM):
  SVM regression aims to find a hyperplane that best fits the data, and it can be effective for house price prediction when you have a relatively small dataset.
- Neural Networks:
  Deep learning models, such as feedforward neural networks and convolutional neural networks (CNNs), can be applied to house price prediction tasks for capturing complex patterns in the data. However, they may require large amounts of data and computational resources.
- Time Series Models:
  If your dataset includes temporal information, you may consider time series models like ARIMA (AutoRegressive Integrated Moving Average) or LSTM (Long Short-Term Memory) networks for capturing time-dependent patterns in house prices.

Predicting House Prices using Machine Learning

5).Model Training:

- Data Splitting:

Split your preprocessed dataset into two or three subsets: a training set, a validation set (optional), and a test set. The training set is used to train the model, the validation set is used for hyperparameter tuning (if necessary), and the test set is used for evaluating the final model's performance.

- Model Selection:
  Choose the machine learning model you want to use for house price prediction based on your analysis and requirements. For example, you might choose Linear Regression, Random Forest, or a Gradient Boosting algorithm.
- Model Training:
  Train the selected model using the training dataset. This involves fitting the model to the preprocessed feature data (independent variables) and the corresponding target variable (house prices). The specific steps for training depend on the chosen model. Typically, you'll use a function or method provided by your machine learning library (e.g., scikit-learn in Python) to train the model.
- Model Evaluation:
  After training the model, evaluate its performance using the test dataset. Common regression metrics for house price prediction include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (R2) score. The choice of metrics depends on your specific goals and the nature of the problem.
- Documentation:
  Document the entire process, including data preprocessing steps, model selection, training, hyperparameter tuning, and evaluation. This documentation is essential for reproducibility and future reference.

Predicting House Prices using Machine Learning

6).Evaluation:

Mean Absolute Error (MAE):

- **MAE measures the average absolute difference between the predicted house prices and the actual prices. It provides a straightforward measure of prediction accuracy without considering the direction of errors.**
- **Formula: MAE = (1 / n) ∑ |actual - predicted|.**
- **Lower MAE values indicate better model performance.**

Mean Squared Error (MSE):

- **MSE measures the average squared difference between the predicted house prices and the actual prices. It penalizes larger errors more heavily than MAE, making it more sensitive to outliers.**
- **Formula: MSE = (1 / n) ∑ (actual - predicted)^2.**
- **Lower MSE values indicate better model performance.**

Root Mean Squared Error (RMSE):

- **RMSE is the square root of MSE and provides an interpretation of the average prediction error in the same unit as the target variable.**
- **Formula: RMSE = √MSE**
- **Lower RMSE values indicate better model performance.**

R-squared (R2) Score:

- **R2 measures the proportion of the variance in the target variable (house prices) that is explained by the model. It ranges from 0 to 1, where 1 indicates a perfect fit, and 0 indicates that the model does not explain any variance.**
- **Formula: R2 = 1 - (SSE / SST), where SSE is the sum of squared errors, and SST is the total sum of squares.**
- **Higher R2 values indicate better model performance, with values close to 1 suggesting a good fit.**

Predicting House Prices using Machine Learning

Conclusion:

House price prediction in machine learning is a valuable application with significant real-world implications. It involves using various predictive models and regression techniques to estimate the prices of residential properties based on relevant features and historical data.

PREDICTING HOUSE PRICES USING MACHINE LEARNING

Problem statement: Consider exploring advanced regression techniques like Gradient Boosting or XGBoost for improved prediction accuracy.

One effective machine learning method that is frequently employed in problems involving home price prediction is gradient boosting. By capturing complex correlations between multiple parameters (such as square footage, the number of bedrooms, location, etc.) and the objective variable (home prices), it can aid in enhancing forecast accuracy. Gradient boosting can be used to predict property prices in the following ways:

✞ **Data Preparation**:

- Start by collecting and preparing your dataset. Clean the data, handle missing values, and preprocess features (e.g., encoding categorical variables, scaling numeric features).

✞ **Train-Test Split**:

- Split the dataset into training and testing sets to evaluate the model's performance effectively. Common splits include 70% for training and 30% for testing.

✞ **Feature Engineering**:

- Create relevant features or transform existing ones. For house price prediction, this might involve computing features like the total square footage of the house, the age of the property, or the presence of specific amenities (e.g., pool, garage).

✞ **Gradient Boosting Model Selection**:

  - Choose a Gradient Boosting algorithm suitable for regression tasks. Common options include:
  - **Gradient Boosting Regressor**: A standard Gradient Boosting algorithm for regression tasks.

- **XGBoost (Extreme Gradient Boosting)**: An optimized and efficient version of Gradient Boosting, widely used in machine learning competitions.
- **LightGBM**: A gradient boosting framework designed for high efficiency and scalability.
- **CatBoost**: Another gradient boosting library that handles categorical features well.

Gradient boosting regressor:

The Gradient Boosting Regressor algorithm is a powerful machine learning technique used for predicting continuous numerical values, making it particularly well-suited for tasks like house price predictions. It builds an ensemble of decision trees sequentially, where each tree is trained to correct the errors made by the previous ones.

Choose the Gradient Boosting Regressor algorithm as the model for your house price prediction task. we can use libraries like Scikit-Learn for Python, which provide easy access to this algorithm.

Working of Gradient Boosting Regressor:

- The single decision tree that the Gradient Boosting Regressor starts with is frequently quite shallow (has a small depth). Because it makes unreliable predictions, this tree is referred to as a "weak learner".
- The method is concentrated on the residuals of the weak learner's mistakes. It figures out the discrepancies between the real target values and the forecasts given by the inexperienced learner.
- Progressive Boosting A new weak learner (another shallow decision tree) is then fitted by the regression to the residuals. This second tree's goal is to foretell the residuals, essentially rectifying the first tree's mistakes.
- The new weak learner's predictions are scaled by a variable known as the "learning rate" and added to those made by the preceding learner. Iteratively repeating this method until a stopping requirement is satisfied, for a predetermined number of iterations.

- Every time the algorithm adjusts the predictions and lowers the mistakes, it fits new weak learners to the residuals. The weighted average of all the learners' predictions forms the final forecast.

Accuracy in gradient boosting algorithm:

To assess the accuracy of your Gradient Boosting Regressor in house price predictions, you would typically calculate one or more of these metrics on a holdout or test dataset. Here's a general process:

- Utilise a training dataset to hone your Gradient Boosting Regressor.
- Make predictions using the trained model on a different test dataset.
- By contrasting the model's predictions with the actual home prices in the test dataset, determine the chosen evaluation metrics (for example, MAE, MSE, RMSE, and R2).

A model is more accurate and better at forecasting house values when it has a lower MAE, MSE, or RMSE and a higher R2. When analysing these metrics, it is crucial to take into account the particular context of your application and any applicable business requirements. In order to comprehend the nature of prediction errors and spot any patterns or trends in the errors, you might also wish to do residual analysis.

XGboost(Extreme Gradient Boosting):

Extreme Gradient Boosting, or XGBoost, is a well-known and effective machine learning technique that is a member of the gradient boosting method family. It is well known for its exceptional prediction performance and scalability and is commonly utilised for both regression and classification tasks. In machine learning contests, XGBoost is frequently a top contender and is also used to tackle practical issues in a variety of industries.

The gradient boosting framework is used by XGBoost. It develops an ensemble of decision trees successively, starting with an initial prediction (often the mean of the target values), to correct the mistakes caused by the earlier trees.

Accuracy of XG boosting in House Price Predictions:

1. Train the XGBoost model on a training dataset containing historical data with features (e.g., square footage, number of bedrooms, location) and corresponding actual house prices.
2. Use the trained model to make predictions on a separate test dataset containing similar features but with actual house prices withheld.
3. Calculate the chosen evaluation metrics (e.g., MAE, MSE, RMSE, R^2) by comparing the model's predictions to the actual house prices in the test dataset.
4. Interpret the evaluation metrics: A lower MAE, MSE, or RMSE and a higher R^2 indicate better accuracy, meaning that the XGBoost model's predictions are closer to the actual house prices.

When analysing these KPIs, it's crucial to take into account the particular context of your application and any applicable business requirements. Additionally, performing residual analysis will help you comprehend the types of errors made in predictions and see any patterns or trends that may help you improve your model moving forward.

Light GBM:

A robust and effective gradient boosting framework for machine learning is called LightGBM (Light Gradient Boosting Machine). LightGBM, created by Microsoft, is renowned for its strong predictive performance, speed, and ability to handle big datasets. It is frequently employed for both classification and regression applications and excels in situations where computational resources are constrained.

LightGBM follows the gradient boosting framework, which builds an ensemble of decision trees sequentially to improve predictive accuracy. It starts with an initial prediction (usually the mean of the target values) and progressively adds more trees to correct errors.

One of the distinguishing features of LightGBM is its use of histogram-based learning. Instead of using the traditional method of splitting data into continuous bins, LightGBM bins feature values into histograms, which reduces memory usage and speeds up training.

Accuracy of Light GBM in House Price Predictions:

1. Train the LightGBM model on a training dataset that includes historical data with features (e.g., square footage, number of bedrooms, location) and corresponding actual house prices.
2. Use the trained model to make predictions on a separate test dataset containing similar features but with actual house prices withheld.
3. Calculate the chosen evaluation metrics (e.g., MAE, MSE, RMSE, R^2) by comparing the model's predictions to the actual house prices in the test dataset.
4. Interpret the evaluation metrics: A lower MAE, MSE, or RMSE and a higher R^2 indicate better accuracy, meaning that the LightGBM model's predictions are closer to the actual house prices.

When analysing these data, it's crucial to take your application's unique context into account as well as any applicable business requirements.
Additionally, performing residual analysis will help you understand the types of errors that are made in predictions and spot any patterns or trends that could be used to direct future model improvements.

Cat Boost:

- CatBoost, short for Categorical Boosting, is a machine learning algorithm. It is designed to handle categorical features efficiently and is well-suited for a variety of tasks, including house price predictions.
- Prepare your dataset, including the goal variable (home pricing) and features (such as square footage, the number of bedrooms, and location). CatBoost is ideally suited for datasets with mixed data types due to the way it handles categorical characteristics.
- To evaluate your model, divide your dataset into a training set and a test set.
- Set the depth, learning rate, and regularisation parameters for the CatBoost hyperparameters.
- Utilise the training dataset to train the CatBoost model.

Accuracy of Cat Boost algorithm in House Price Predictions:

1. Train the CatBoost model on a training dataset that includes historical data th features (e.g., square footage, number of bedrooms, location) and rresponding actual house prices.
2. e the trained model to make predictions on a separate test dataset ntaining similar features but with actual house prices withheld. Calculate
3. e chosen evaluation metrics (e.g., MAE, MSE, RMSE, R^2) by comparing e model's predictions to the actual house prices in the test dataset. terpret the evaluation metrics: A lower MAE, MSE, or RMSE and a higher R^2
4. dicate better accuracy, meaning that the CatBoost model's predictions are oser to the actual house prices.

Conclusion:

An adaptable and reliable method for predicting home price trends is gradient boosting. It may produce precise and reliable predictions when used appropriately and with thorough model adjustment, making it a useful tool in the real estate and related industries. To create the most accurate predictive model for your particular use case, it's crucial to take into account aspects like data quality, feature engineering, and business objectives.

```
import pandas as pd
import numpy as np


import seaborn as sns
import matplotlib.pyplot as plt


from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean_absolute_error , mean_squared_error ,
median_absolute_error , r2_score

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression ,
SGDRegressor ,Ridge , Lasso
```

Loading Dataset

```
df=pd.read_csv(r'USA_Housing.csv')
df.head(10)
```

```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388
5      80175.754159             4.988408                   6.104512
6      64698.463428             6.025336                   8.147760
7      78394.339278             6.989780                   6.620478
8      59927.660813             5.362126                   6.393121
9      81885.927184             4.423672                   8.167688

   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09     23086.800503  1.059034e+06
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05
5                          4.04     26748.428425  1.068138e+06
6                          3.41     60828.249085  1.502056e+06
7                          2.42     36516.358972  1.573937e+06
8                          2.30     29387.396003  7.988695e+05
9                          6.10     40149.965749  1.545155e+06

                                             Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
```

```
1  188 Johnson Views Suite 079\nLake Kathleen, CA...
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3                         USS Barnett\nFPO AP 44820
4                        USNS Raymond\nFPO AE 09386
5  06039 Jennifer Islands Apt. 443\nTracyport, KS...
6  4759 Daniel Shoals Suite 442\nNguyenburgh, CO ...
7     972 Joyce Viaduct\nLake William, TN 17778-6483
8                         USS Gilbert\nFPO AA 20957
9                    Unit 9446 Box 0958\nDPO AE 97025
```

Preprocessing the dataset

```
df.shape
```

```
(5000, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

```
df.describe()
```

```
       Avg. Area Income  Avg. Area House Age  Avg. Area Number of
Rooms  \
count       5000.000000          5000.000000
5000.000000
mean       68583.108984             5.977222
6.987792
std        10657.991214             0.991456
1.005833
min        17796.631190             2.644304
3.236194
25%        61480.562388             5.322283
6.299250
50%        68804.286404             5.970429
7.002902
```

```
75%        75783.338666             6.650808
7.665871
max       107701.748378             9.519088
10.759588

       Avg. Area Number of Bedrooms  Area Population         Price
count                   5000.000000      5000.000000  5.000000e+03
mean                       3.981330     36163.516039  1.232073e+06
std                        1.234137      9925.650114  3.531176e+05
min                        2.000000       172.610686  1.593866e+04
25%                        3.140000     29403.928702  9.975771e+05
50%                        4.050000     36199.406689  1.232669e+06
75%                        4.490000     42861.290769  1.471210e+06
max                        6.500000     69621.713378  2.469066e+06
```

```
df.isnull().sum()
```

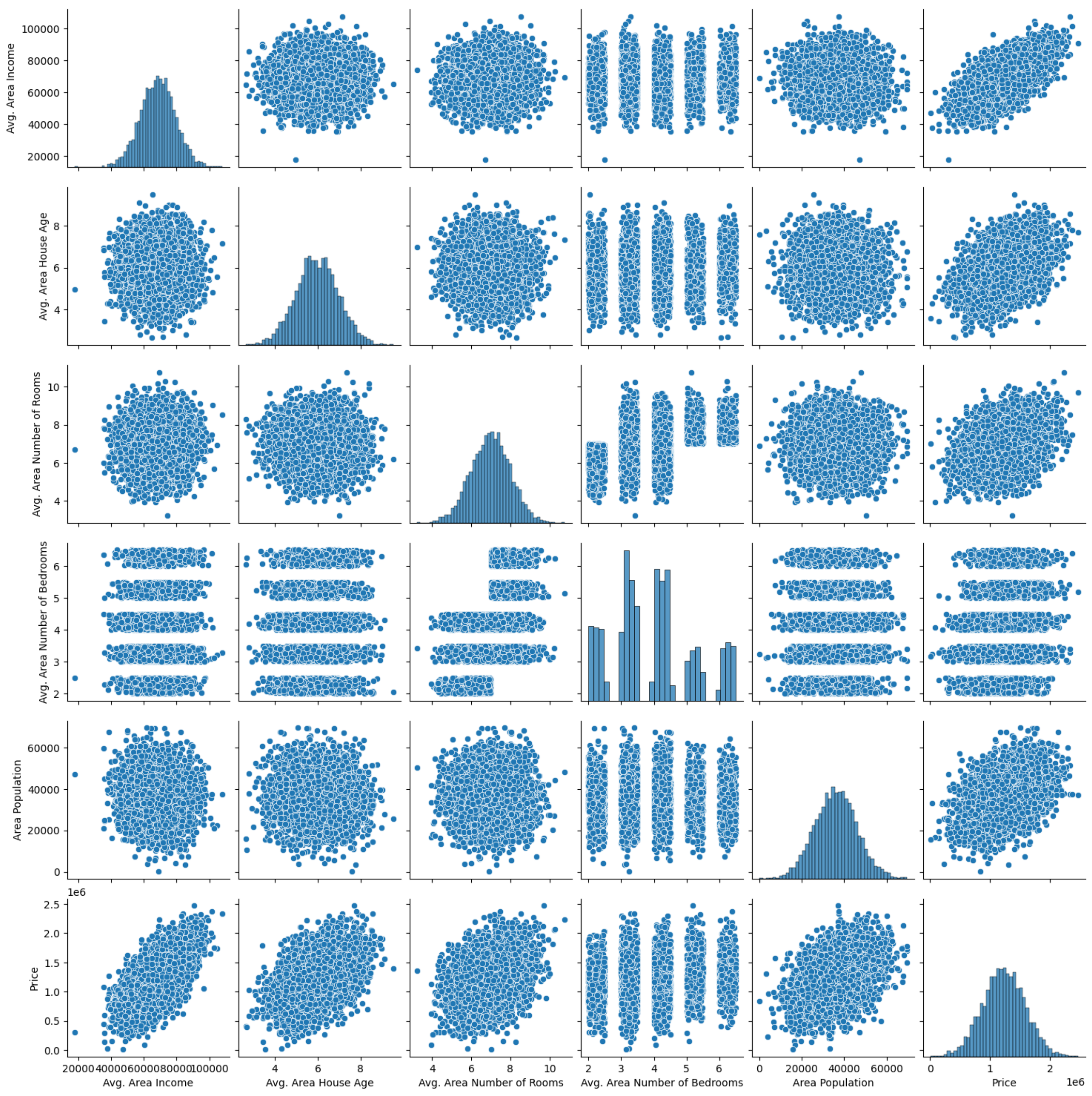
```
Avg. Area Income                0
Avg. Area House Age             0
Avg. Area Number of Rooms       0
Avg. Area Number of Bedrooms    0
Area Population                 0
Price                           0
Address                         0
dtype: int64
```

```
df.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
4995    False
4996    False
4997    False
4998    False
4999    False
Length: 5000, dtype: bool
```

```
sns.pairplot(df)
```

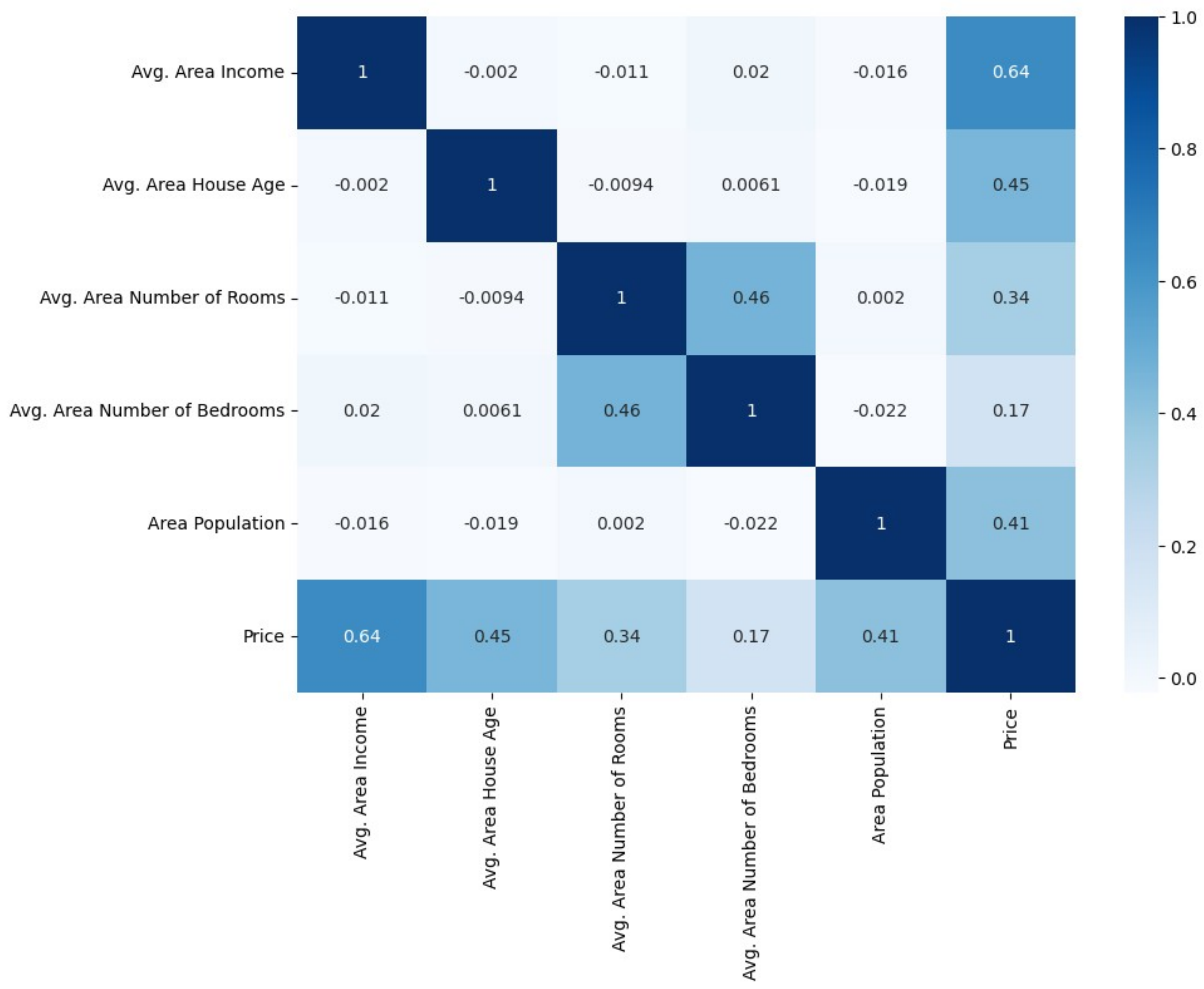
```
<seaborn.axisgrid.PairGrid at 0x7f4d11732710>
```



```
plt.figure(figsize=(10,7))
sns.heatmap(df.corr() , annot=True ,cmap='Blues')
```

```
/tmp/ipykernel_12463/850008773.py:2: FutureWarning: The default value
of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr() , annot=True ,cmap='Blues')

<Axes: >
```



```
y=df['Price'].values
X=df.iloc[:,:-2].values

X_train, X_test, y_train, y_test = train_test_split( X, y,
test_size=0.3, random_state=42)

scaler=StandardScaler()

X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(X_train , y_train)
```

```
LinearRegression()
```

```
print('Training data accurancy score :',regressor.score(X_train ,
y_train) )
print('Testing data accurancy
score :',regressor.score(X_test,y_test) )
```

```
Training data accurancy score : 0.9192986579075526
Testing data accurancy score : 0.9146818498754016
```

```
coef=pd.DataFrame(data=regressor.coef_, index=df.iloc[:,:-
2].columns,columns=['Coefficients'])
print('The Coefficient of each column :')
coef
```

```
The Coefficient of each column :
```

```
 Coefficients
Avg. Area Income 232679.724643
Avg. Area House Age 163841.046593
Avg. Area Number of Rooms 121110.555478
Avg. Area Number of Bedrooms 2892.815119
Area Population 151252.342377
```

```
print('The Intercept with y-axis :',regressor.intercept_)
```

```
The Intercept with y-axis : 1228219.1492415695
```

```
y_pred =regressor.predict(X_test)
```

```
real_data=pd.DataFrame(y_test , columns=['Y_real'])
data_fin =real_data.assign(Y_predicted =y_pred)
data_fin
```

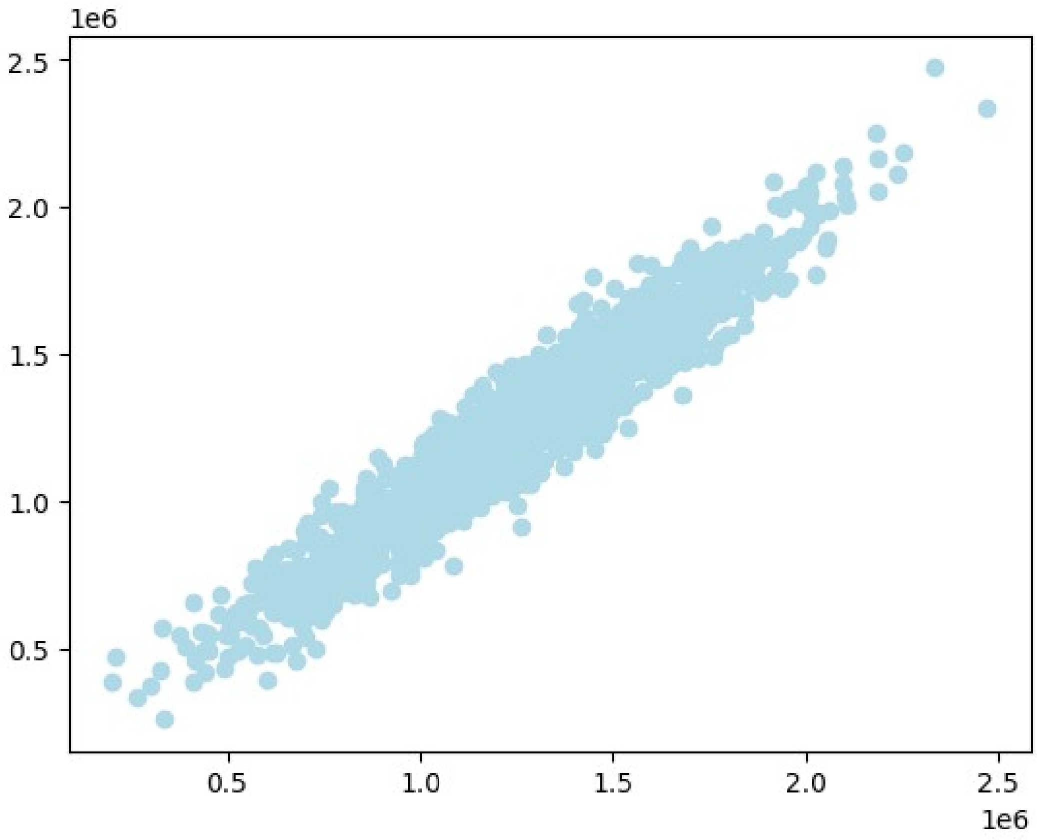
```
 Y_real Y_predicted
0 1.339096e+06 1.308536e+06
```

```
1 1.251794e+06 1.237123e+06
2 1.340095e+06 1.243836e+06
3 1.431508e+06 1.229242e+06
4 1.042374e+06 1.059353e+06
... ... ...
1495 1.348222e+06 1.437325e+06
1496 1.309937e+06 1.094962e+06
1497 1.472887e+06 1.457120e+06
1498 1.409762e+06 1.483429e+06
1499 1.009606e+06 1.047511e+06

[1500 rows x 2 columns]
```

```
plt.scatter(y_test,y_pred , color='lightblue' )
```

```
<matplotlib.collections.PathCollection at 0x7f21481702b0>
```

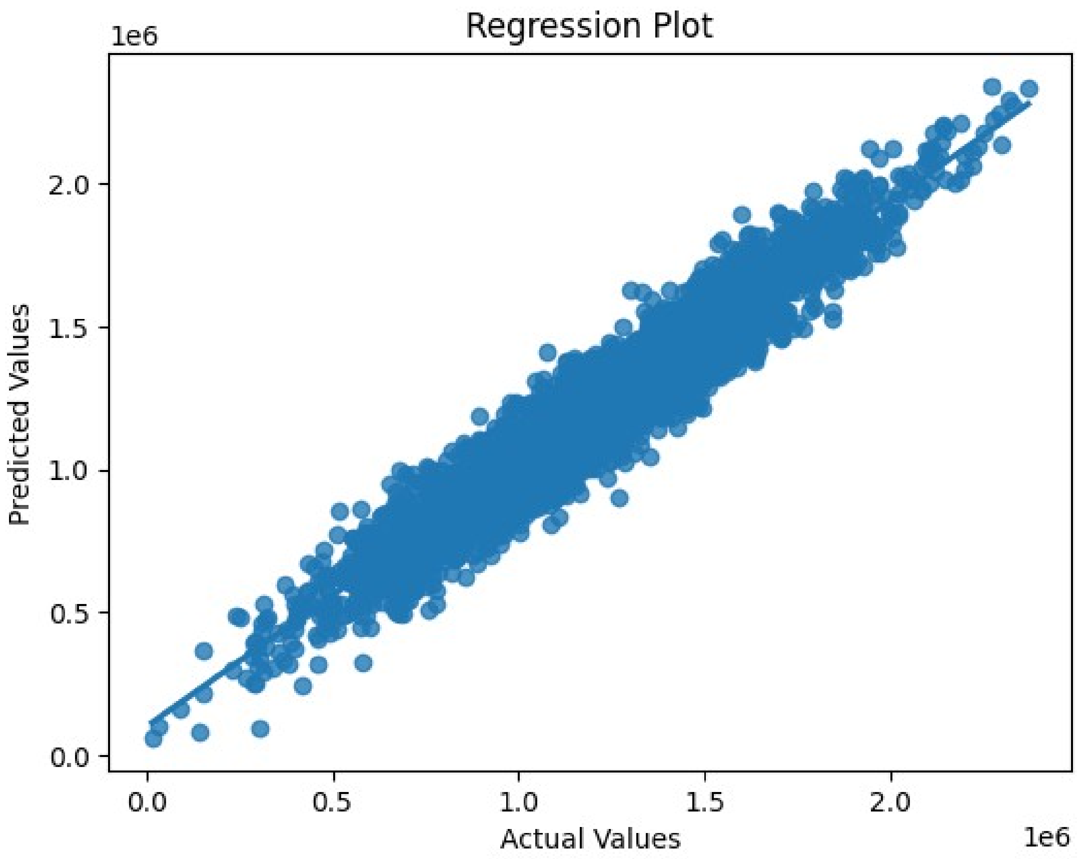


```
import seaborn as sns
import matplotlib.pyplot as plt


regressor.fit(X_train, y_train)
```

```
y_pred  =  regressor.predict(X_train)

sns.regplot(x=y_train, y=y_pred)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.title("Regression Plot")
plt.show()
```



```
print('Mean Absolute Error :',mean_absolute_error(y_test,y_pred))
print('Median Absolute Error :',median_absolute_error(y_test,y_pred))
print('Mean Squared Error :',mean_squared_error(y_test,y_pred))

Mean Absolute Error : 81135.5660933688
Median Absolute Error : 70547.8609665166
Mean Squared Error : 10068422551.401031
```

Model Accuracy:	The accuracy of the house price prediction models depends heavily on the availability quality of the data, the choice of features, and the complexity of the algorithm.
Feature Importance:	Identifying the most influential features for predicting house prices can significantly improve the accuracy of the models.
Data Quality:	The quality and cleanliness of the data play a vital role in the accuracy of the predictions.
Model Selection:	The choice of the appropriate machine learning algorithm is critical. Various algorithms, such as linear regression, decision trees, random forests, and neural networks, have their own strengths and limitations.
Temporal Considerations:	In the case of time-series data, the temporal aspects should be considered, as the housing market tends to be influenced by various seasonal and cyclical trends.
Generalization and Overfitting:	Striking the right balance between model complexity and generalization is crucial