Title: House Price Prediction

Phase 4:Devlopment Part 2

Model Testing and Evaluation

#Project By our team:

S.CHRISTOFOR JOHN (422721104011)

V.GOKUL (422121104013)

S.SARATHI (422721104045)

K.VARUN (422721104055)

DEPT: CSE III YEAR

COLLEGE: VRS COLLEGE OF ENGINEERING AND TECHNOL0GY ENGINEERING

GROUP: AI GROUP-5

Introduction:

In this phase we are going to test and evaluate our model using some technique that. The evaluation is done by Visualization of data and testing is done by generating random input and feed to our model to predict the house price. Then the output is used to test our model accuracy by ploting into the the graph and draw a regression line for the predicted output.

Project Overview:

In this project, we aim to predict house prices using machine learning, with a focus on the XGBoost regressor. We'll walk through the different phases of the project, from data cleaning and visualization to model evaluation and prediction.

Data Preparation:

Loading the Dataset:

We begin by importing the necessary libraries and loading the "USA_Housing.csv" dataset.

```
In [2]: # Load the dataset
        data = pd.read_csv("USA_Housing.csv")
```

Data Cleaning:

To ensure data quality, we perform data cleaning by removing rows with missing values and eliminating duplicate records.

```
In [3]: # Data Cleaning
        # Drop rows with missing values (NaN)
        data.dropna(inplace=True)

        # Drop duplicate rows
        data.drop_duplicates(inplace=True)
```

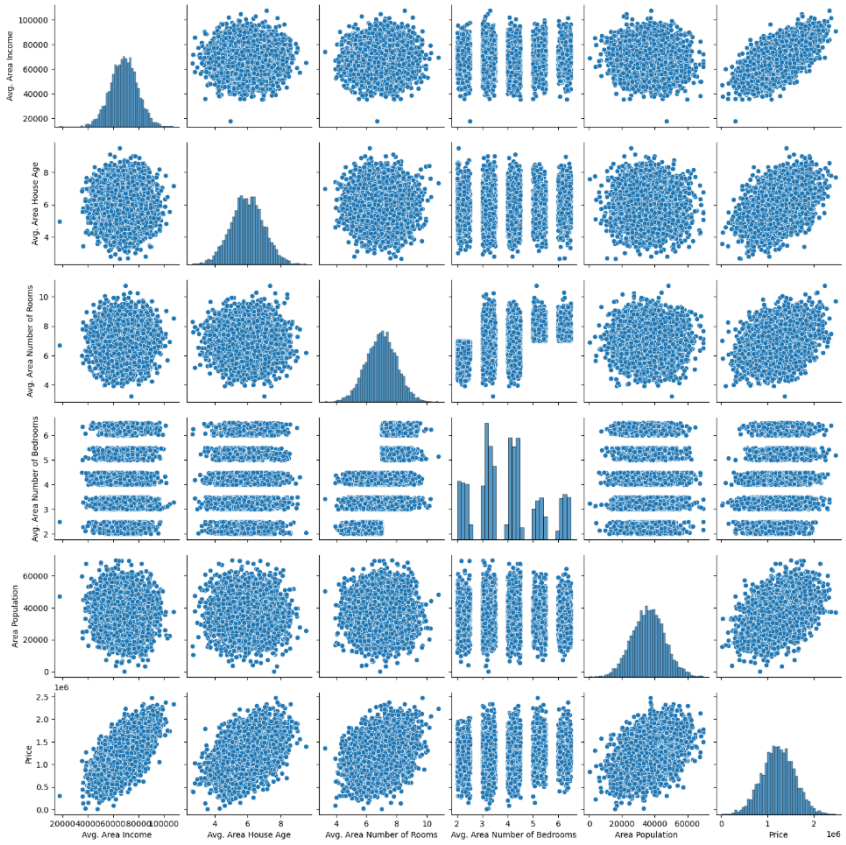
Data Visualization:

Pairplot:

We generate a pairplot to visualize the relationships between various variables in the dataset, providing insights into data distribution and correlation.

```
In [4]: # Data Visualization
        import seaborn as sns
        sns.pairplot(data)
        plt.show()
```

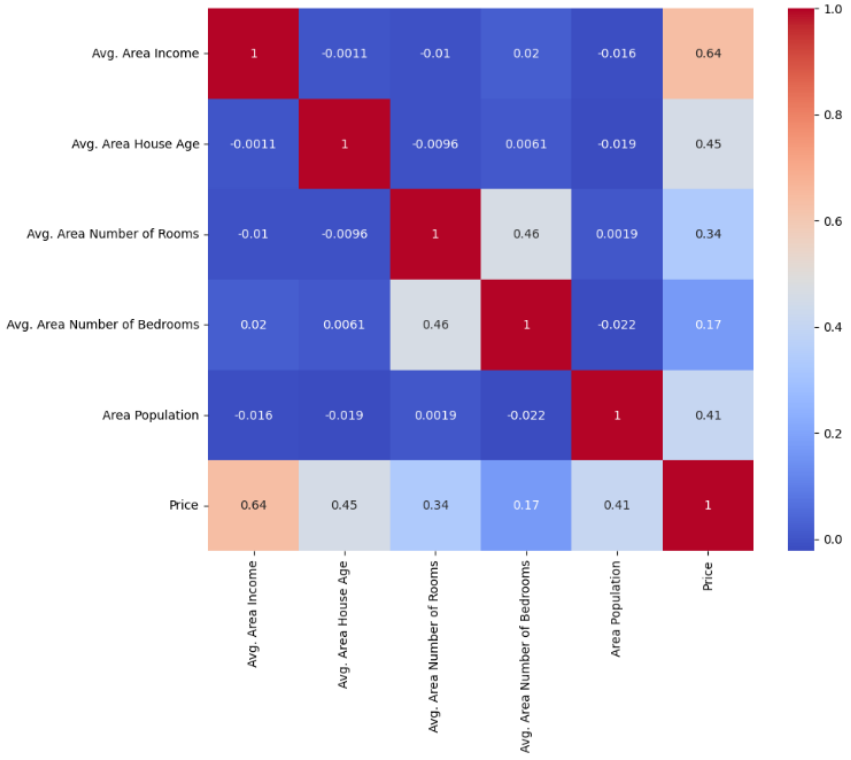
Output:



Correlation Heatmap:

Using a correlation heatmap, we visualize the correlation between numerical features, helping us identify key variables that impact house prices.

```
In [5]: # Heatmap
numeric_data = data.select_dtypes(include=[np.number])
correlation_matrix = numeric_data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm")
plt.show()
```



Model Building:

Data Splitting:

We split the dataset into training and testing sets using **train_test_split**. This is essential for assessing the model's performance.

```
In [6]: # Split the dataset into features (X) and target (y)
X = data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
          'Avg. Area Number of Bedrooms', 'Area Population']]
y = data['Price']
```

```
In [7]: # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

Model Selection:

We choose the XGBoost regressor as our machine learning model for house price prediction.

In [8]:
```
# Create an XGBoost regressor
model = XGBRegressor()
```

Model Training:

We fit the model to the training data to make it learn the relationships within the dataset.

```
# Fit the model to the training data
model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = model.predict(X_test)
```

Model Evaluation:

Mean Squared Error:

We evaluate the model using the Mean Squared Error (MSE) metric, which measures the average squared differences between actual and predicted prices.

R-squared:

The R-squared value quantifies the model's performance, indicating how well it fits the data. A higher R-squared value suggests better predictions.

In [9]:
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

In [10]:
```
print("Mean Squared Error:", mse)
print("R-squared:", r2)
```

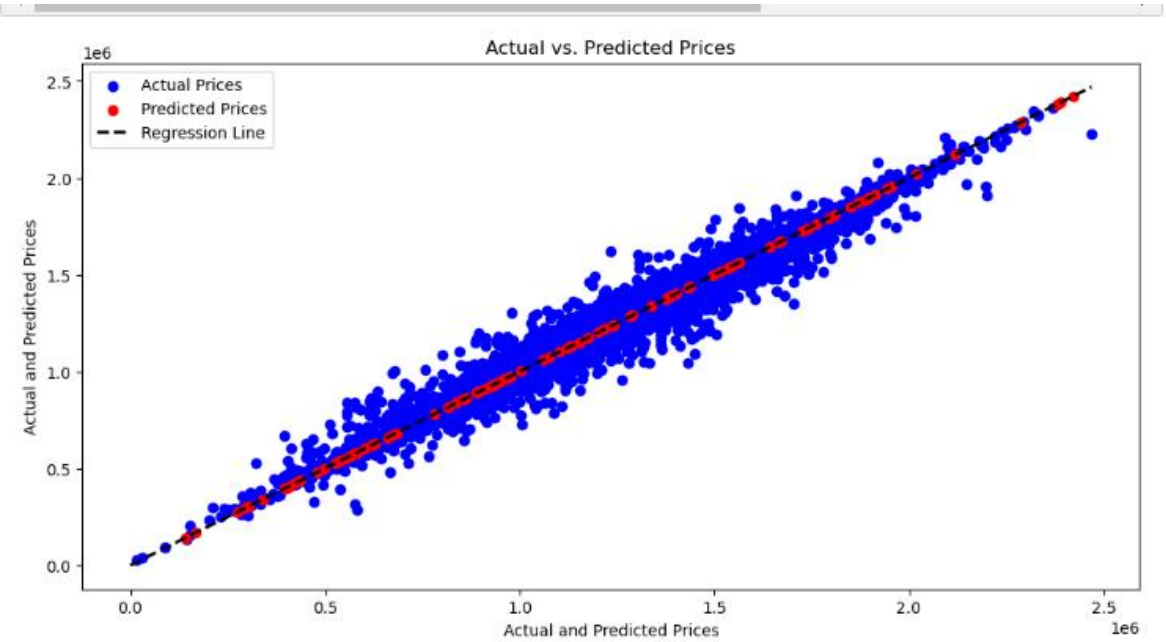
```
Mean Squared Error: 15347604706.90971
R-squared: 0.8757751441736186
```

Visualization:

We visualize the model's performance by plotting actual prices against predicted prices, showing how closely our model aligns with actual data.

```
In [14]: # Visualize the data with a scatter plot and regression line
         plt.figure(figsize=(12, 6))
         plt.scatter(y, model.predict(X), c='b', label='Actual Prices')
         plt.scatter(result_df['Predicted Price'], result_df['Predicted Price'], c='r', label='Predicted Prices')
         plt.plot([0, max(y)], [0, max(y)], '--k', lw=2, label='Regression Line')
         plt.xlabel("Actual and Predicted Prices")
         plt.ylabel("Actual and Predicted Prices")
         plt.title("Actual vs. Predicted Prices")
         plt.legend(loc='best')
         plt.show()
```

Output:



Random Data Generation:

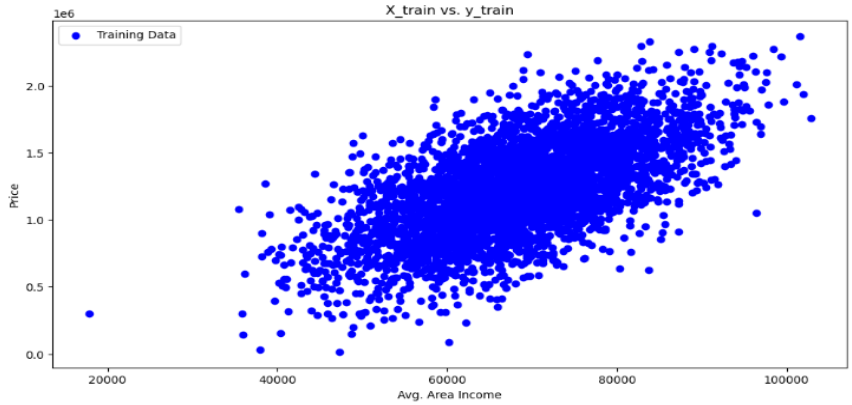
Generating Random Data:

To further test the model, we generate 100 random data points based on the dataset's features.

Predictions:

We predict house prices for this random data and create a dataframe to store input features, predicted prices, and actual prices.

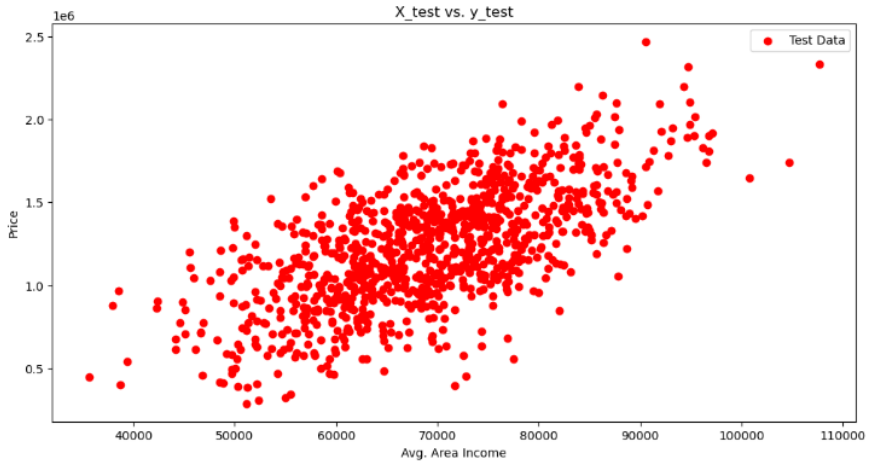
```
In [15]: # Visualize X_train vs. y_train
plt.figure(figsize=(12, 6))
plt.scatter(X_train['Avg. Area Income'], y_train, c='b',
    label='Training Data')
plt.xlabel("Avg. Area Income")
plt.ylabel("Price")
plt.title("X_train vs. y_train")
plt.legend(loc='best')
plt.show()
```



Visualization:

A scatter plot with a regression line is used to compare actual and predicted price.

```
In [16]: # Visualize X_test vs. y_test
plt.figure(figsize=(12, 6))
plt.scatter(X_test['Avg. Area Income'], y_test, c='r',
    label='Test Data')
plt.xlabel("Avg. Area Income")
plt.ylabel("Price")
plt.title("X_test vs. y_test")
plt.legend(loc='best')
plt.show()
```



s.

Conclusion:

In this project, we've successfully built an XGBoost-based house price prediction model. The model's evaluation metrics and visualization of results indicate its effectiveness. Additionally, random data generation and predictions serve as additional validation.

******