HOUSE PRICE PREDICTION USING MACHINE LEARNING

TEAM MEMBER

961321104025 : SUMAN.G

Phase-1 : Problem Definition and Design Thinking

Project Title: House Price Predictor

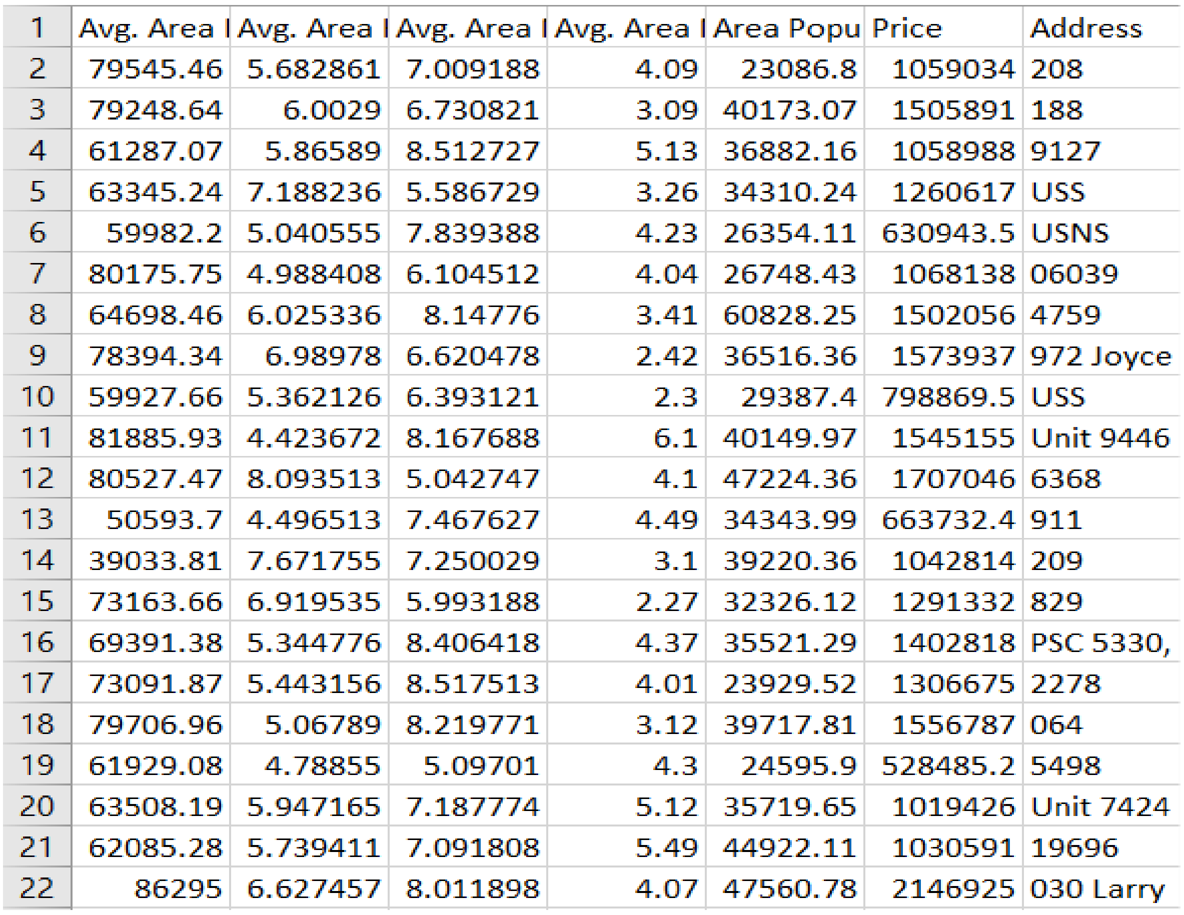
OBJECTIVE:

The objective of this project is to develop a machine learning model that accurately predicts the price of house based on set of features such as location, square footage, number of bedrooms and bathrooms, and other relevant factors.

1. Data Source:

The data source for this project is collected from www.Kaggle.com, which containing information about houses, including features like, location, square footage, rooms, bedrooms, and price.

DATASET LINK: https://www.kaggle.com/datasets/vedavyasv/usa-housing



1	Avg. Area I	Avg. Area I	Avg. Area I	Avg. Area I	Area Popu	Price	Address
2	79545.46	5.682861	7.009188	4.09	23086.8	1059034	208
3	79248.64	6.0029	6.730821	3.09	40173.07	1505891	188
4	61287.07	5.86589	8.512727	5.13	36882.16	1058988	9127
5	63345.24	7.188236	5.586729	3.26	34310.24	1260617	USS
6	59982.2	5.040555	7.839388	4.23	26354.11	630943.5	USNS
7	80175.75	4.988408	6.104512	4.04	26748.43	1068138	06039
8	64698.46	6.025336	8.14776	3.41	60828.25	1502056	4759
9	78394.34	6.98978	6.620478	2.42	36516.36	1573937	972 Joyce
10	59927.66	5.362126	6.393121	2.3	29387.4	798869.5	USS
11	81885.93	4.423672	8.167688	6.1	40149.97	1545155	Unit 9446
12	80527.47	8.093513	5.042747	4.1	47224.36	1707046	6368
13	50593.7	4.496513	7.467627	4.49	34343.99	663732.4	911
14	39033.81	7.671755	7.250029	3.1	39220.36	1042814	209
15	73163.66	6.919535	5.993188	2.27	32326.12	1291332	829
16	69391.38	5.344776	8.406418	4.37	35521.29	1402818	PSC 5330,
17	73091.87	5.443156	8.517513	4.01	23929.52	1306675	2278
18	79706.96	5.06789	8.219771	3.12	39717.81	1556787	064
19	61929.08	4.78855	5.09701	4.3	24595.9	528485.2	5498
20	63508.19	5.947165	7.187774	5.12	35719.65	1019426	Unit 7424
21	62085.28	5.739411	7.091808	5.49	44922.11	1030591	19696
22	86295	6.627457	8.011898	4.07	47560.78	2146925	030 Larry

Python Program:

```
# importing necessary libraries

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

%matplotlib inline

import pandas as pd


# importing data

df = pd.read_csv('./USA_Housing.csv')


# displaying data to get an overview

print(df.head(3))


# displaying an information about data

print("information:")

print(df.info())
```

Output:

```
  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

   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

                                             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...
Information:
<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
None
```

2. DATA PREPROCESSING

Data preprocessing is the critical first step in any machine learning project. It involves cleaning the data, removing outliers, and handling missing values to prepare the dataset for model training. In the context of house price prediction project.

a) DUPLICATE REMOVAL:

Duplicate rows can introduce bias into the model. We will identify and remove duplicates, typically by sorting the dataset based on a unique identifier (eg: property ID) and then eliminating consecutive rows with the same identifier.

b) HANDLING MISSING VALUES:

Missing data is common and needs to be addressed, we will utilize suitable methods such as:

- Mean Imputation: replacing missing values with the feature of remaining rows. This is appropriate for numerical features
- Median Imputation: if data contains outliers, median imputation can be more robust as it less sensitive to extreme values

c) CATEGORICAL VARIABLE ENCODING:

Categorical variable, such as property type or location, need to be converted into numerical form so that machine learning models can process them. There are two common approach

- One-Hot Encoding: Create binary columns for each category, representing the presence or absence of that category.
- Label Encoding: Assign a unique integer to each category, preserving the ordinal relationship if applicable.

Python Program:

```
# importing necessary libraries
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer

# data preprocessing
# data cleaning

# handling missing values
numeric_cols = df.select_dtypes(include='number').columns
categorical_cols = df.select_dtypes(exclude='number').columns
```

```
print(numeric_cols)
print(categorical_cols)

imputer_numeric = SimpleImputer(strategy='mean')
imputer_categorical = SimpleImputer(strategy='most_frequent')

df[numeric_cols] = imputer_numeric.fit_transform(df[numeric_cols])
df[categorical_cols] = imputer_categorical.fit_transform(df[categorical_cols])

# Convert Categorical Features to Numerical
label_encoder = LabelEncoder()
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])

# display categorical column data after numerical conversion
print(df[categorical_cols].loc[1])
```

Output:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price'],
      dtype='object')

Index(['Address'], dtype='object')

Address    863
Name: 1, dtype: int32
```

3. FEATURE SELECTION:

Feature Selection is the process of identifying and selection the most relevant features from a dataset for a given machine learning task. The goal of feature features selection is to improve the performance of the machine learning model by reducing the number of features and eliminating irrelevant or redundant features.

Python Program:

```
# determining features & target variable
print(df.columns)

X=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
      'Avg. Area Number of Bedrooms', 'Area Population']]
# display single data from X
print('X:\n',X.loc[1])

y=df['Price']
```

```
# display single data from y
print('y:\n',y.loc[1])
```

Output:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
X:
 Avg. Area Income                79248.642455
Avg. Area House Age                 6.002900
Avg. Area Number of Rooms           6.730821
Avg. Area Number of Bedrooms        3.090000
Area Population                 40173.072174
Name: 1, dtype: float64
y:
 1505890.91484695
```

4. MODEL SELECTION:

Model selection is the process of choosing a suitable machine learning algorithm for a given machine learning task. The goal of model selection is to find an algorithm that is both accurate and efficient, for this house price prediction we use linear regression algorithm.

LINEAR REGRESSION:

Linear regression is a simple but efficient house price prediction. Linear regression models the relationship between the house price and the features using a linear function.

RANDOM FOREST REGRESSOR:

Random forest regressor is a more complex algorithm that builds a multitude of decision trees to predict the house price. Random forest regressor are typically more accurate that linear regression models, but they can be more computationally expensive to train.

GRADIENT BOOSTING REGRESSOR:

Gradient boosting regressor is another complex algorithm to builds a sequence of decision tree to predict the house price. Gradient boosting regressor are typically more accurate than random forest regressors, but they can be even more computationally expensive to train.

5. MODEL TRAINING:

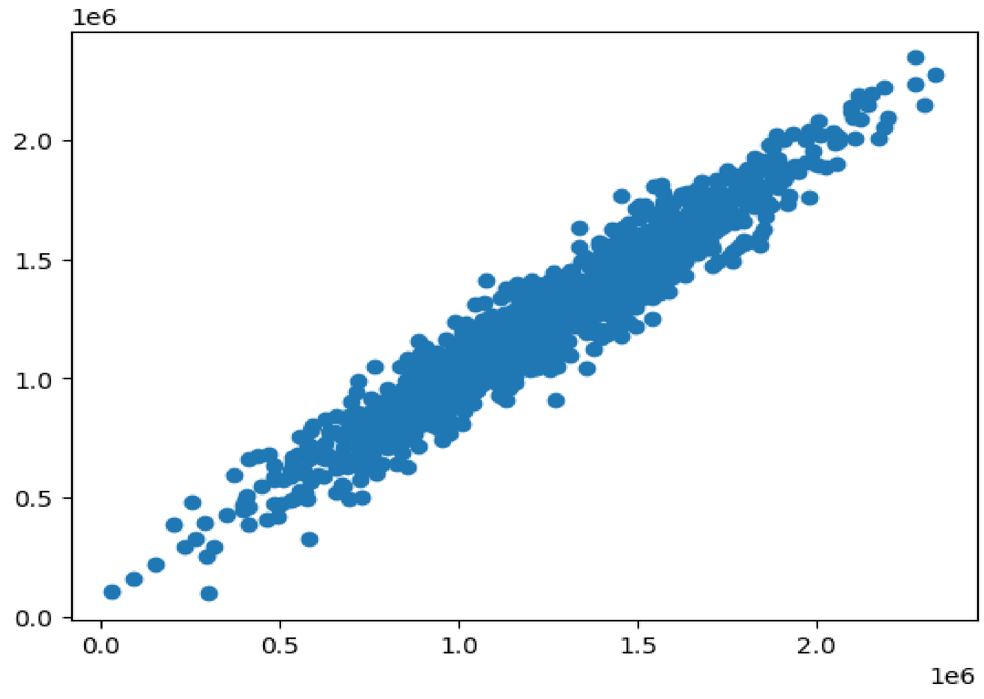
The task involves training a selected machine learning model using preprocessed data and subsequently evaluating the model’s performance using key metrics such as Mean Absolute Error, Root Mean Squared Error, and R-squared.

Python Program:

```
# importing necessary library
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# spliting the dataset to train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

# training the model
model= LinearRegression()
model.fit(X_train, y_train)
print(pd.DataFrame(model.coef_, X.columns, columns=['coef']))

# perdicting test data
predictions=model.predict(X_test)
plt.scatter(y_test, predictions)
```

Output:

```
                                        coef
Avg. Area Income                  21.617635
Avg. Area House Age          165221.119872
Avg. Area Number of Rooms      121405.376596
Avg. Area Number of Bedrooms     1318.718783
Area Population                   15.225196

<matplotlib.collections.PathCollection at 0x241e4676d10>
```



6. EVALUATION:

Mean Absolute Error (MAE):

MAE measures the average absolute difference between the predicted values and the actual target values. It provides insight into the average magnitude of error made by the model.

Root Mean Squared Error (RMSE):

RMSE is another common metric that calculate the square root of the average of squared difference between predicted and actual values. It provides information about the typical magnitude of errors and gives higher penalties to larger errors.

R-Squared:

R-Squared quantifies the proportion of the variance in the target variable that is explained by the model. It ranges from 0 – 1, where a higher value indicates a better fit. It is often used to assess how well the model captures the variation in the data.

Python Program:

```
# importing necessary libraries
from sklearn import metrics

# evaluating the model
print('MAE:', metrics.mean_absolute_error(y_test,predictions))
print('R-Squared:', metrics.r2_score(y_test,predictions))
print('RSME:', np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

Output:

```
MAE: 81257.55795855941
R-Squared: 0.9185060945363622
RSME: 100842.08231635048
```

CONCLUSION:

In phase 1, we have established a clear understanding of goal: to predict house price using machine learning. We outlined a structured approach that includes data source selection, data preprocessing, features selection, model selection, model training, and evaluation. This sets the stage for our project’s successful execution in subsequent phases.