PREDICTING HOUSE PRICES USING MACHINE LEARNING

Introduction:

To build a house price prediction project using machine learning . The objective is to develop a model that accurately predicts the prices of houses based on a set of features such as price,address,number of bedrooms, and other relevant factors.

Problem Statement:

The problem is to predict house prices using machine learning techniques. Using machine learning, we can easily find the house which is to be prefect for us and to predict the prices accurately. This project involves data pre-processing, feature engineering, model selection, training, and evaluation.

Design Thinking Process:

Data Source: Data processing techniques and processes are numerous. We choose a dataset containing information about houses, including features like location, square footage, bedrooms, bathrooms, and price.

Data Preprocessing: Data preprocessing is the process of cleaning our data set. Cleaning and preprocessing the data, handle missing values, and convert categorical features into numerical representations.

Feature Selection: Selecting the most relevant features for predicting house prices.

Model Selection: Choosing a suitable regression algorithm (e.g., Linear Regression, Random Forest Regressor) for predicting house prices.

Model Training: Training the selected model using the preprocessed data. Since the data is broken down into two modules: a Training set and Test set, we must initially train the model. The training set includes the target variable.

Evaluation: Evaluating the model's performance using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared.

Phases of development:

Phase 1: Problem definition and Design thinking

In this phase, the problem statement is clearly explained. The various stages of design thinking like data source, data preprocessing, feature selection , model selection, model training and evaluation are neatly explained.

Phase 2: Innovation

In this phase, the complete steps taken to put the design that we have considered in the previous phase has to be made into transformation. We have to put our design into innovation. We consider exploring advanced regression techniques like XGBoost to improve prediction accuracy.

Phase 3: Development Part 1

We start building the house price prediction model by loading and preprocessing the data. We perform different analysis as needed.

Phase 4: Development Part 2

We continue building the house price prediction model by feature selection, model training and evaluation.

Phase 5: Project documentation and Submission

We document the house price prediction project and prepare it for submission.

Dataset and its description:

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

This USA_Housing.csv dataset has been provided by Kaggle. Kaggle is an online community of data scientists and machine learners, owned by Google LLC. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science.

The dataset contains the following columns:

- 'Avg. Area Income': Avg. Income of residents of the city house is located in.
- 'Avg. Area House Age': Avg Age of Houses in same city
- 'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city
- 'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city
- 'Area Population': Population of city house is located in
- 'Price': Price that the house sold at
- 'Address': Address for the house

Steps involved in making our project:

1.Data Source:

In order to proceed further into this project, we should collect the data, then we should pre-process the data and then we perform exploratory data analysis on the given data set.

2. Data Preprocessing:

Data preprocessing is the process of cleaning our data set. Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations and outlier removal.

- Dealing with Missing Values:

  Dealing with the problem of missing values because some machine learning models don't accept data with missing values. For that we can see the number of missing values in our dataset.

- Converting categorical features into numerical representations:

We will encode our categorical features using one-hot encoding technique which transforms the categorical variable into a number of binary variables based on the number of unique categories in the categorical variable.

3.Feature Selection:

Feature selection is a common machine learning technique used to build a simplified model for understanding and to enhance generalization by removing irrelevant or redundant information. Machine learning models accept only numbers as input, and since our dataset contains categorical features, we encode them in order for our dataset to be suitable for modeling. In order to make all algorithms work properly with our data, we need to scale the features in our dataset. For that, we will use a helpful function named StandardScaler() from the popular Scikit-Learn Python package.

4. Model Selection:

The important step is choosing a machine learning algorithm. In this project , XGBoost algorithm is used for predicting the price of the house using a machine learning approach. XGBoost stands for extreme gradient boosting, where gradient boosting is implemented with several additional features focusing on performance and speed. With careful parameter tuning, it is capable of training highly accurate models.

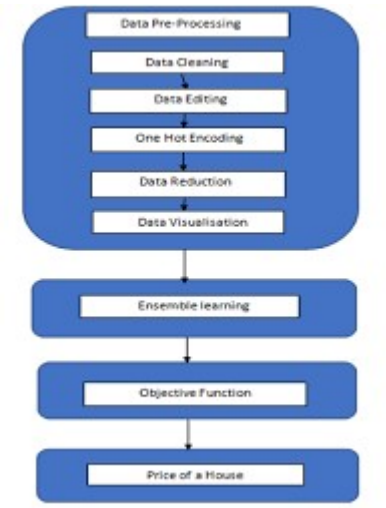
5. Model training:

We train our model for house price prediction using XGBoost algorithm. The data is split down into two modules: a Training set and Test set. The training set includes the target variable. To make all algorithms work properly with our data, we need to scale the features in our dataset. Then we start building our project. We build our XGBoost model with the best parameters found. We must initially train the model. Then, test the model on the test dataset and get the results.

6.Evaluation:

Evaluate the model's performance using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R-squared. Here, we evaluate the model performance by comparing its predictions with the actual true values of tested data using the MAE metric. We also used visualisation tool for evaluating the predicted data.

Architectural diagram:



The data preprocessing steps and model training process are clearly explained in the code given below:

CODE:

Predicting house prices using machine learning

Loading the dataset:

The first step is reading the dataset from the csv file we downloaded.

```
import pandas as pd
import numpy as np
```

```
dataset = pd.read_csv("/content/USA_Housing.csv")
```

```
pd.options.display.float_format = '{:20.2f}'.format
dataset.head(n=5)
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.46	5.68	7.01	4.09	23086.80	1059033.56	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64	6.00	6.73	3.09	40173.07	1505890.92	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.07	5.87	8.51	5.13	36882.16	1058987.99	9127 Elizabeth Stravenue\nDanieltown, WI 06482...

let's get statistical information about the numeric columns in our dataset.

```
dataset.describe(include=[np.number], percentiles=[.5]).transpose().drop("count", axis=1)
```

	mean	std	min	50%	max
Avg. Area Income	68583.11	10657.99	17796.63	68804.29	107701.75
Avg. Area House Age	5.98	0.99	2.64	5.97	9.52
Avg. Area Number of Rooms	6.99	1.01	3.24	7.00	10.76
Avg. Area Number of Bedrooms	3.98	1.23	2.00	4.05	6.50
Area Population	36163.52	9925.65	172.61	36199.41	69621.71
Price	1232072.65	353117.63	15938.66	1232669.38	2469065.59

Then, we move to see statistical information about the non-numerical columns in our dataset:

Data Preprocessing:

Data cleaning:

Dealing with Missing Values: We should deal with the problem of missing values because some machine learning models don't accept data with missing values. Firstly, let's see the number of missing values in our dataset. We want to see the number and the percentage of missing values for each column that actually contains missing values.

```
num_missing = dataset.isna().sum()

num_missing = num_missing[num_missing > 0]

percent_missing = num_missing * 100 / dataset.shape[0]

pd.concat([num_missing, percent_missing], axis=1,
          keys=['Missing Values', 'Percentage']).\
          sort_values(by="Missing Values", ascending=False)
```

	Missing Values	Percentage

```
dataset["Price"].value_counts()
```

```
1059033.56    1
1521141.34    1
```

```
1148372.40    1
2065710.16    1
1749820.01    1
             ..
1444701.33    1
 788427.84    1
 875904.53    1
 984421.23    1
1298950.48    1
Name: Price, Length: 5000, dtype: int64
```

```
dataset['Price'].fillna('No feature', inplace=True)
```

let's check if there is any remaining missing value in our dataset:

```
dataset.isna().values.sum()
```

```
0
```

Converting categorical features into numerical representations:

We will encode our categorical features using one-hot encoding technique which transforms the categorical variable into a number of binary variables based on the number of unique categories in the categorical variable.

```
dataset[['Address']].head()
```

	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

```
dataset = pd.get_dummies(dataset)
```

```
address_oneHot = [c for c in dataset.columns if c.startswith("Address")]
dataset[address_oneHot].head()
```

	Address_000 Adkins Crescent\nSouth Teresa, AS 49642-1348	Address_000 Todd Pines\nAshleyberg, KY 90207-1179	Address_001 Steve Plaza\nJessicastad, UT 25190	Address_0010 Gregory Loaf\nSouth Ericfort, VA 34651-0718	Address_00149 Raymond Knolls\nNew Jason, UT 75026	Address_002 Katherine Flat\nHartmanland, AZ 37973-3049	Address_0022 Young Rest\nLake Kevin, CA 25438-1821	Address_ Melinda 591\r Gregorysl HI (
0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	

5 rows × 5000 columns

Feature Selection:

Selecting the most relevant features for predicting house prices.

```
X = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Number of Bedrooms', 'Area Population']]
Y = dataset['Price']
```

Model Building and Evaluation:

Splitting the data:

we need a training dataset to train our model and a test dataset to evaluate the model. So we will split our dataset randomly into two parts, one for training and the other for testing.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=101)
```

```
Y_train.head()
```

```
3413             1305210.26
1610             1400961.28
3459             1048639.79
4293             1231157.25
1039             1391232.53
Name: Price, dtype: float64
```

```
Y_train.shape
```

```
(4000,)
```

Standardizing the data:

To make all algorithms work properly with our data, we need to scale the features in our dataset.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_scal = sc.fit_transform(X_train)
X_test_scal = sc.fit_transform(X_test)
```

Modeling:

We have used xgboost regression algorithm.

```
from xgboost import XGBRegressor
from sklearn.model_selection import RandomizedSearchCV

parameter_space = \
    {
        "max_depth": [4, 5, 6],
        "learning_rate": [0.005, 0.009, 0.01],
        "n_estimators": [700, 1000, 2500],
        "booster": ["gbtree",],
        "gamma": [7, 25, 100],
        "subsample": [0.3, 0.6],
        "colsample_bytree": [0.5, 0.7],
        "colsample_bylevel": [0.5, 0.7,],
        "reg_alpha": [1, 10, 33],
        "reg_lambda": [1, 3, 10],
    }

clf = RandomizedSearchCV(XGBRegressor(random_state=3),  parameter_space, cv=3, n_jobs=4,scoring="neg_mean_absolute_error" ,random_state=3
clf.fit(X_train, Y_train)
print("Best parameters:")
print(clf.best_params_)
```

```
Best parameters:
{'subsample': 0.6, 'reg_lambda': 10, 'reg_alpha': 10, 'n_estimators': 2500, 'max_depth': 4, 'learning_rate': 0.009, 'gamma': 7, 'co
```

we build our XGBoost model with the best parameters found:

```
xgb_model = XGBRegressor(**clf.best_params_)
```

we train our model using our training set:

```
xgb_model.fit(X_train_scal, Y_train);
```

Predicting Prices:

```
y_pred = xgb_model.predict(X_test_scal)
print(y_pred)
```

```
[1245976.5   782677.56 1693791.1   965170.56  979755.2   646750.8
 1057040.    830071.8  1431099.9  1197648.1  1397959.1  1316015.9
 1695341.4  1276931.8  1377154.1  1182152.1   576097.4   954917.
 1199935.4  1202382.5   516015.78 1721938.8  1761200.2  1185399.9
 1075547.1  1781688.4  1726844.8  1442473.4  1334613.2  1507267.1
  771825.25 1715043.9  1415200.6   994509.44 1226292.9   898196.56
 1211128.    995412.94 1306071.1   766165.4  1392926.2   710576.3
  816781.25 1829430.1  1610588.    964923.4  1100228.1   834148.5
 1152134.9  1455172.5  1426714.6  1155611.6  1084399.   1355186.5
  811177.06  981478.9  1127577.1  1262368.2  1412904.    553008.
 1401322.1  1119077.1   666898.2  1205247.2  1317447.4  1312847.
  770164.1  1462357.   1378863.1   903097.25  832729.1  1193881.4
 1065133.6  1805220.5   938752.75  815205.75  824481.6  1387455.2
  716566.75 1629987.9  1048518.44 1337776.   1272283.8  1161088.6
  780454.2   653229.56 1037503.75 1359368.6  1030076.   1259515.5
  530128.7  1665477.9   839129.25 1585533.2  1003546.7   914340.94
 1177607.9  1497553.9   904518.1   963354.94 1293866.   1594332.2
  977592.6  1425910.6  1020598.1  1429731.2   875411.    811756.7
  983460.06 1781253.6  1850731.2  1374917.   1020346.6  1054523.2
 1443067.6   989831.4  1138557.2  1353490.4  1251818.5  1522629.
  761960.2  1375488.9  2003338.9  1408435.8  1309945.9  1260702.1
  901227.1   787137.94 1658782.9  1156781.4   804902.9   797000.75
 1500750.2   807861.06  612415.8  1214568.2  1015149.2  1127418.2
 1743913.8  1146609.9   587511.44 1366554.2  1340100.8  1682448.6
 1146391.8   940850.2  1714339.6  1071124.9  1603927.9  1404990.2
 1355839.6   936140.9  1743655.9  1220214.4  1189614.1   954079.9
 1215688.5  1232872.4  1235361.4  1130275.8  1501174.   1518622.1
  754458.94 1507055.8  1249687.4  1043025.4  1285001.2  1590073.4
 1125203.1   452490.9  1585616.8  1305087.5  1691244.6  1657147.4
 1643898.4  1789545.5  1540740.2   863019.1  1398801.9  1553175.9
 1375250.6  1098951.8  1423329.4   862656.25  933576.4  1135039.2
 1161830.4  1075511.2  1315606.1  1121894.5  1005345.   1860049.6
 1268599.9  1158853.6  1864908.6  1138624.2   638460.7  1494228.8
 1192738.2  1098275.8  1691183.1   839590.2   959550.44 1312380.8
  887882.3  1292466.4  1218735.8   963816.1  1199474.   1955396.6
 1469954.2  1115057.   1897259.   1531206.4  1229241.8  1255883.4
 1278953.5   290238.72 1097409.2  1261194.4  1360012.6  1709601.5
 1037033.   1021330.06 1419971.5   901380.3   849442.9  1650343.
 1152343.5  1360353.6   805469.75 1499488.2  1294056.4  1788028.1
 1224186.   1209327.5   617314.94 1339496.4   447641.34 1234403.1
 1363001.2  1555654.1  1010476.8  1095950.8  1422622.1  1549270.2
 1318741.5  1767452.8  1668699.1   931584.44 1236547.1  1097623.
 1467367.   1147653.6  1541595.4  1592310.4   432503.12  443719.06
  830936.75 1797482.6  1160272.6  1266255.1   577449.2  1355219.5
 1763136.2   590837.    878033.   1233764.8  1208419.2  1146874.1
 1538212.9  1324988.2   784592.9   891839.06 1703797.9  1019139.5
 1285343.9  1538439.8  1104124.   1068997.2  1706038.9  1395827.1
 1073209.2  1588123.9  1069633.8  1678661.1  1001016.56 1307823.4
 1289877.2  1298618.6  1016938.3  1320756.2  1123396.6  1507990.1
 1721597.5  1959184.9   821205.5   996996.2  1097749.    953779.6
 1372524.6   795225.3  1204508.9  1292104.5  1471064.2  2180610.
  773127.94 1164970.8  1058797.   1250029.8  1599913.    918090.75
  982963.   1549428.6   779682.9   845983.4   580195.56 1767558.6
 1489818.1  1580570.2  1221700.5  1297775.2  1049536.   1167059.9
 1105472.1  1100124.4  1008109.2  1927116.    753471.5  1098812.9
 1307260.1  1546845.4  1085846.   1117448.1  1506715.6  1423899.8
 1132030.4  1006307.1  1661975.8  1089029.4  1856908.    952819.1
  346407.78 1408254.9  1320984.1  1912898.    489201.56 1063913.4
```

Evaluation of Predicted Data:

Then we evaluate the model performance by comparing its predictions with the actual true values in Y_test using the MAE metric :

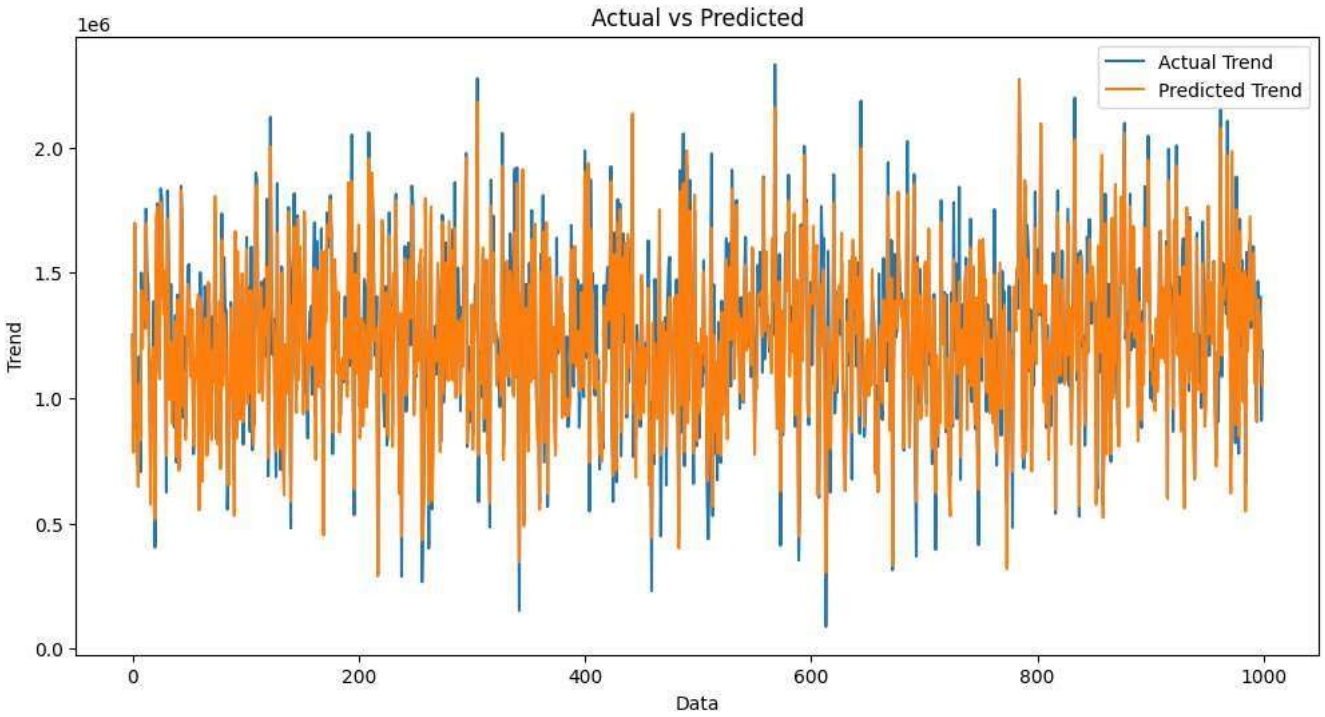
```
from sklearn.metrics import mean_absolute_error
xgb_mae = mean_absolute_error(Y_test, y_pred)
print("XGBoost MAE =", xgb_mae)
```

```
XGBoost MAE = 88382.87552199
```

Evaluation of predicted data using visualisation tool:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), y_pred, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

```
Text(0.5, 1.0, 'Actual vs Predicted')
```



Choice of XGBoost regression algorithm:

- As it has low mean absolute error, the XGBoost algorithm is expected to perform superior to the other models to predict the housing price.
- Has an advantage in situations with many features to consider.
- The speed of execution and the performance of the model is too good.
- High accuracy

Evaluation metrics:

We evaluate the model performance by comparing its predictions with the actual true values of tested data using the MAE(Mean Absolute Error) metric. We also used visualisation tool for evaluating the predicted data. We have compared the actual and predicted trend in data using visualisation plot. In this case , matplotlib library could be used.

Conclusion:

Thus, the machine learning model to predict the house price based on given dataset is executed successfully XGBoost. It helps people looking to sell a house at best time for greater profit. With appropriate dataset, house prices could be predicted in any location.