**HOUSE PRICE PREDICTION USING MACHINE LEARNING**

**BY S.KATHIR**

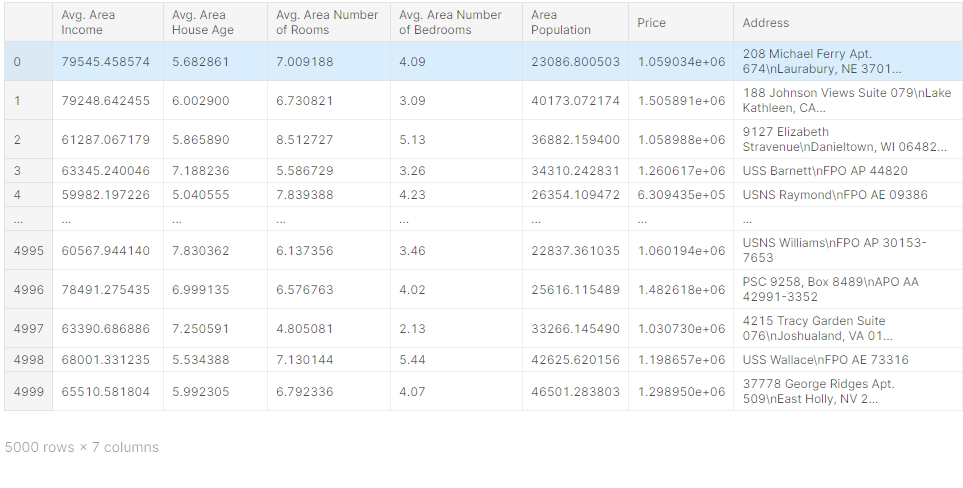
**PHASE III:DEVOLOUPMENT PART 1**

****

**INTRODUCTION**:

* House price prediction using machine learning (ML) is the use of ML algorithms to predict the future selling price of a house based on historical data on various features of the house, such as its location, size, amenities, and condition
* ML models can be trained on large datasets of house sales data to learn the relationships between these features and house prices. Once trained, the model can be used to predict the price of a new house by providing it with the relevant feature information.
* House price prediction using ML can be a valuable tool for buyers, sellers, and investors in the real estate market. Buyers can use it to get an estimate of the fair market value of a house they are interested in purchasing. Sellers can use it to set a competitive asking price for their home. And investors can use it to identify investment opportunities.

**GIVEN DATA:**

****

**Necessory steps to follow:**

Loading and preprocessing a dataset are two important steps in machine learning. Loading involves reading the data from a file or database into memory. Preprocessing involves cleaning and transforming the data so that it is ready for machine learning algorithms.

**Loading a dataset:**

The first step is to load the dataset into memory. This can be done using a variety of programming languages and libraries

**Preprocessing a dataset**

Once the dataset is loaded into memory, it is important to preprocess the data before using it for machine learning. Preprocessing involves cleaning and transforming the data so that it is ready for machine learning algorithm

**Some common preprocessing tasks include:**

* Handling missing values: Missing values are a common problem in datasets. There are a number of ways to handle missing values, such as dropping rows with missing values, imputing missing values with the mean or median of the column, or using a machine learning algorithm to predict the missing values.
* Encoding categorical data: Categorical data is data that can be divided into categories, such as gender, country, or product type. Categorical data must be encoded before it can be used by machine learning algorithms. One common way to encode categorical data is to use one-hot encoding.
* Scaling numerical data: Scaling numerical data involves transforming the data so that it has a mean of 0 and a standard deviation of 1. This is done to improve the performance of many machine learning algorithms.
* Feature engineering: Feature engineering is the process of creating new features from existing features. This can be done to improve the performance of machine learning algorithms.

**Program:**

**Importing dependences:**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score, mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

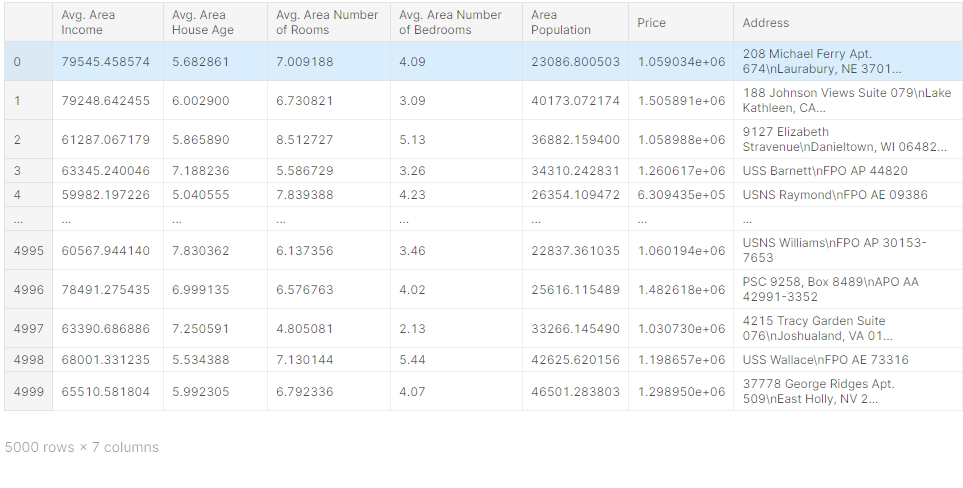
import xgboost as xg

**Loading Dataset:**

dataset = pd.read\_csv('/kaggle/input/usa-housing/USA\_Housing.csv')

**Data Exploration:**

dataset

****

dataset.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

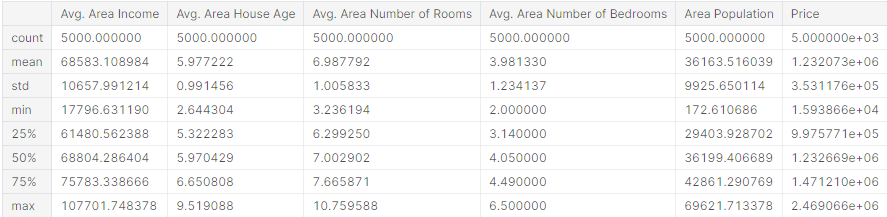
5 Price 5000 non-null float64

6 Address 5000 non-null object

dtypes: float64(6), object(1)

memory usage: 273.6+ KB

dataset.describe()

****

dataset.columns

Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',

'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],

dtype='object')

# Visualisation and Pre-Processing of Data

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

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



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

<Axes: xlabel='Price'>



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

<seaborn.axisgrid.JointGrid at 0x7caf1d571810>



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

<seaborn.axisgrid.JointGrid at 0x7caf1d8bf7f0>

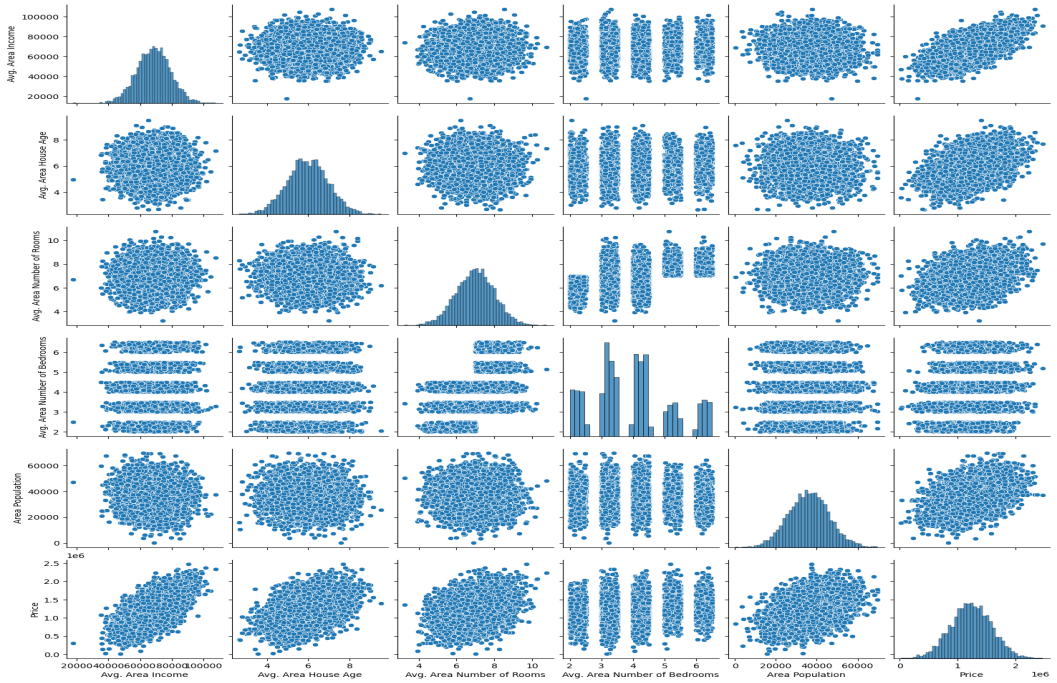


plt.figure(figsize=(12,8))

sns.pairplot(dataset)

<seaborn.axisgrid.PairGrid at 0x7caf0c2ac550>

<Figure size 1200x800 with 0 Axes>



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

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'}>],

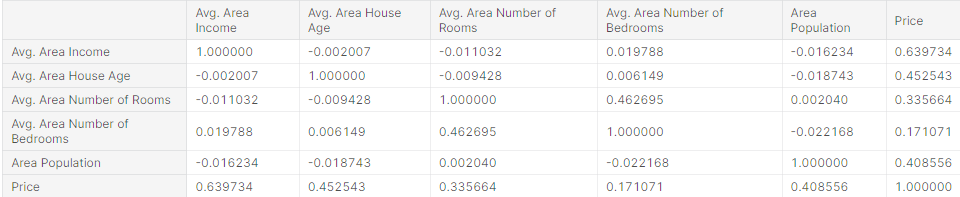
[<Axes: title={'center': 'Area Population'}>,

<Axes: title={'center': 'Price'}>]], dtype=object)

****

## Visualising Correlation

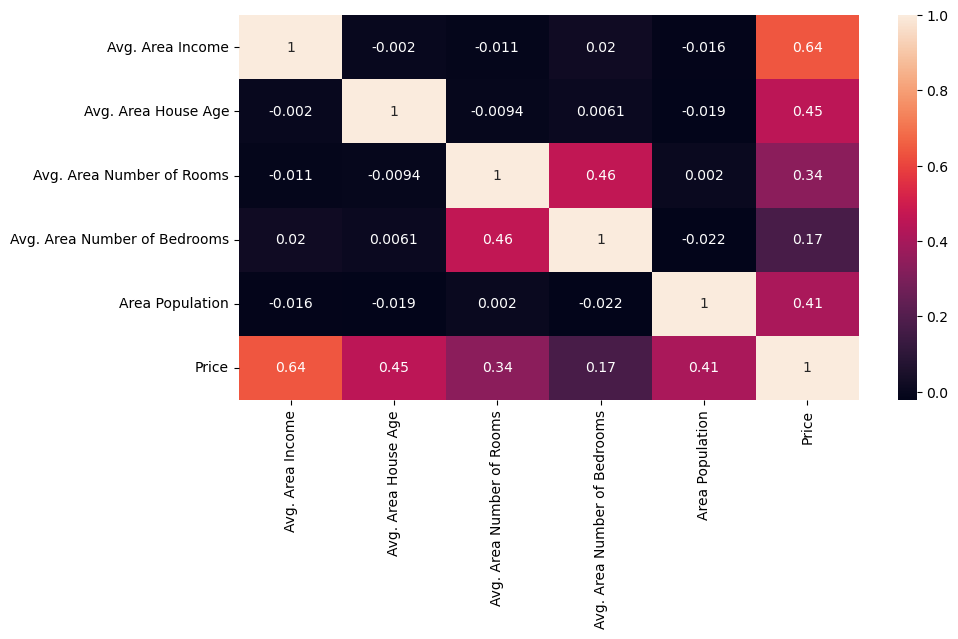
dataset.corr(numeric\_only=True)



plt.figure(figsize=(10,5))

sns.heatmap(dataset.corr(numeric\_only = True), annot=True)

<Axes: >



**Thank you**