HOUSE PRICE PREDICTION

1. Data Collection:

- Gather a dataset with features related to houses (e.g., square footage, number of bedrooms, location, etc.) and their corresponding prices.
- Ensure your dataset is diverse and representative of the kind of data the model will encounter in the real world.

2. Data Pre-processing:

- Handle missing values: Decide on a strategy to deal with missing data (e.g., removing rows, filling with mean/median, or using more advanced imputation methods).
- Encode categorical variables: Convert categorical variables into numerical representations using techniques like one-hot encoding.
- Scale numerical features: Standardize or normalize numerical features to bring them to a similar scale.

3. Feature Engineering:

- Create new features that might have a significant impact on house prices (e.g., price per square foot, age of the house, etc.).
- Remove irrelevant features that might not contribute much to the prediction.

4. Split Data:

- Split your dataset into training and testing sets to evaluate the model's performance on unseen data.

5. Choose a Model:

- Select a regression model suitable for predicting continuous values. Common choices include Linear Regression, Decision Trees, Random Forests, and Gradient Boosting.

6. Train the Model:

- Train your chosen model on the training dataset.

7. Hyperparameter Tuning:

- Fine-tune the hyperparameters of your model to optimize its performance.

8. Evaluate the Model:

- Use the testing dataset to evaluate how well your model is performing. Common metrics include Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

9. Adjust and Improve:

- Based on the evaluation results, make adjustments to your model. This could involve tweaking hyperparameters, adding more features, or trying a different algorithm.

10. Deployment:

- Once satisfied with the model's performance, deploy it to a production environment where it can make predictions on new data.

11. Monitor and Update:

- Regularly monitor the model's performance in a real-world setting. Update the model as needed with new data to ensure it stays accurate over time.

12. Consider Advanced Techniques:

- Explore more advanced techniques like ensemble methods, neural networks, or other regression algorithms for potentially improved performance.

```
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

%matplotlib inline import
warnings
warnings.filterwarnings("ignore")
```

```
dataset = pd.read_csv('/kaggle/input/usa-housing/USA_Housing.csv')
```

Data Exploration

```
dataset
```

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address	
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...

2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address	
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386
...	...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 429913352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316

4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

```
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()
```

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05

50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

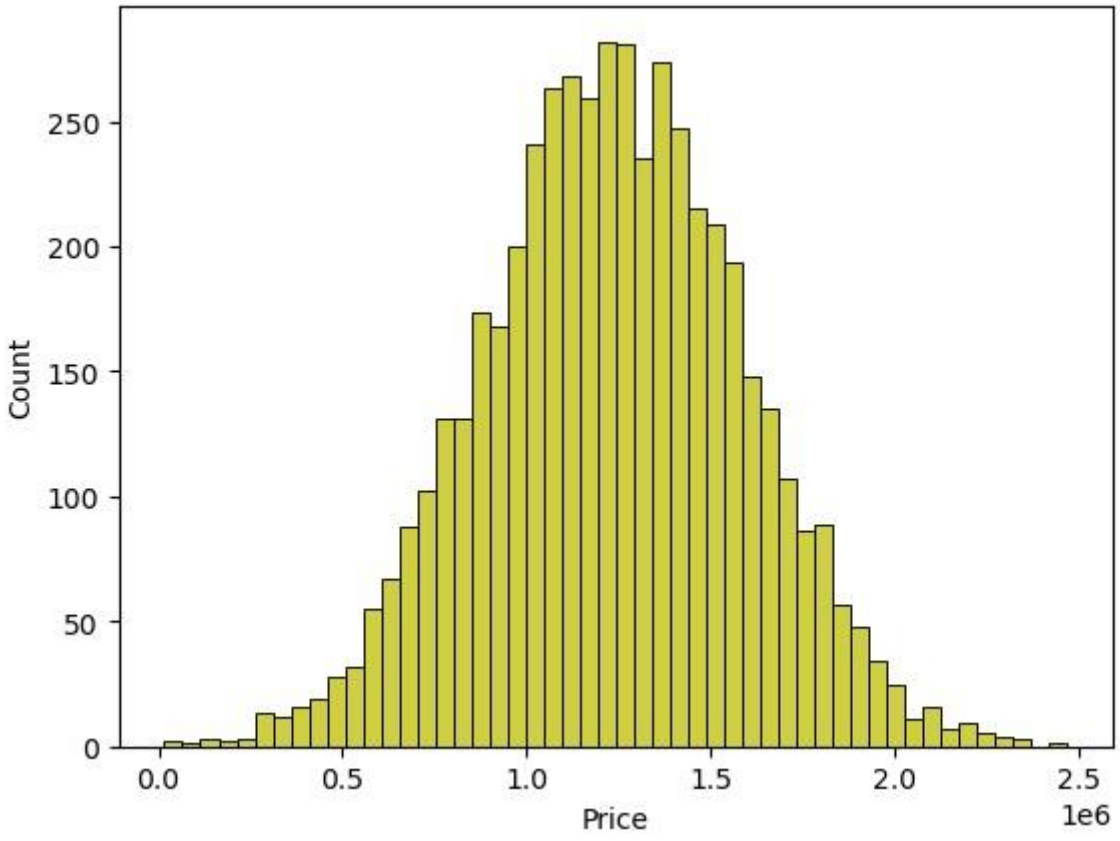
```
dataset.columns
```

```
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

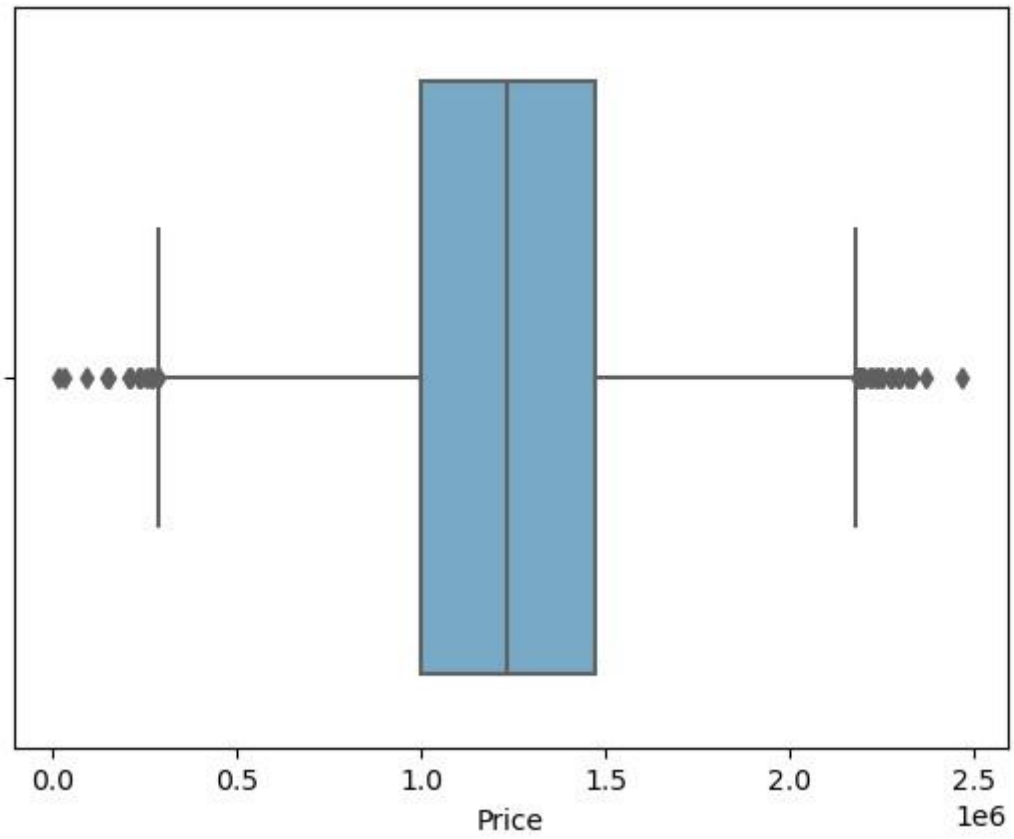
```
sns.histplot(dataset, x='Price', bins=50, color='y')
```

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

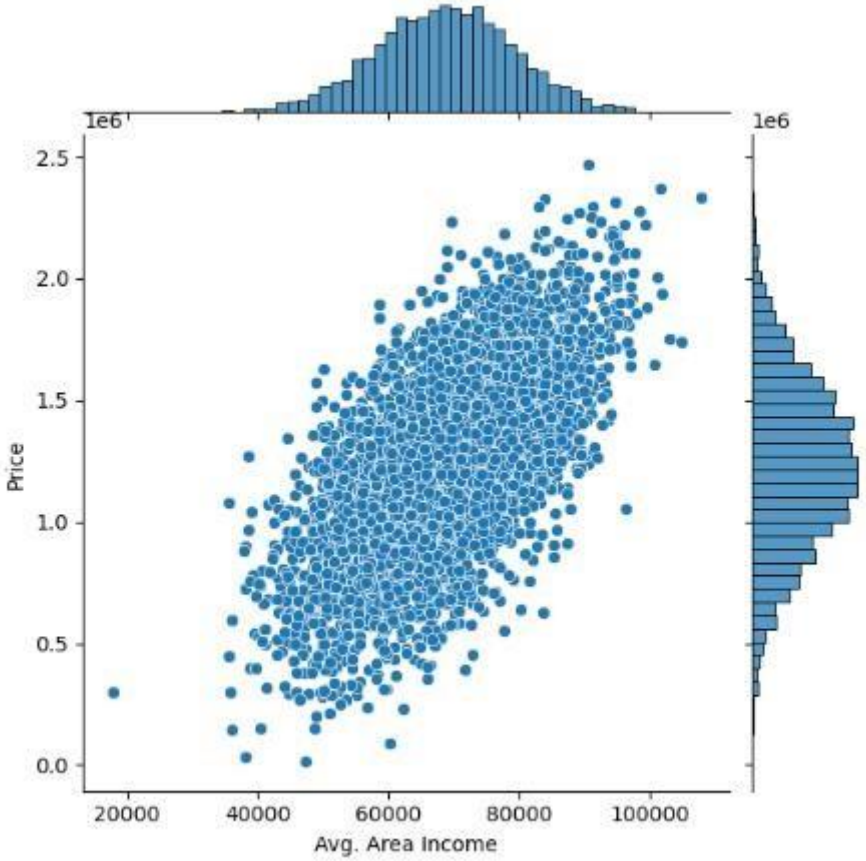



```
sns.boxplot(dataset, x='Price',  palette='Blues')
```

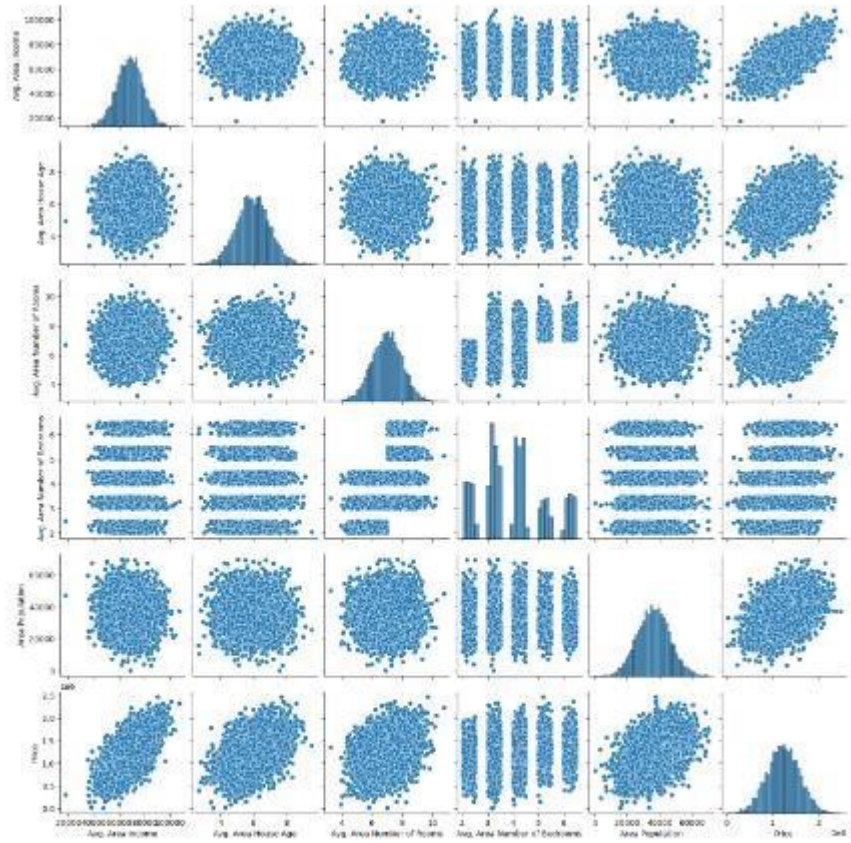
```
<Axes: xlabel='Price'>
```




```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex')
<seaborn.axisgrid.JointGrid at 0x7f2b65fe5780>
```

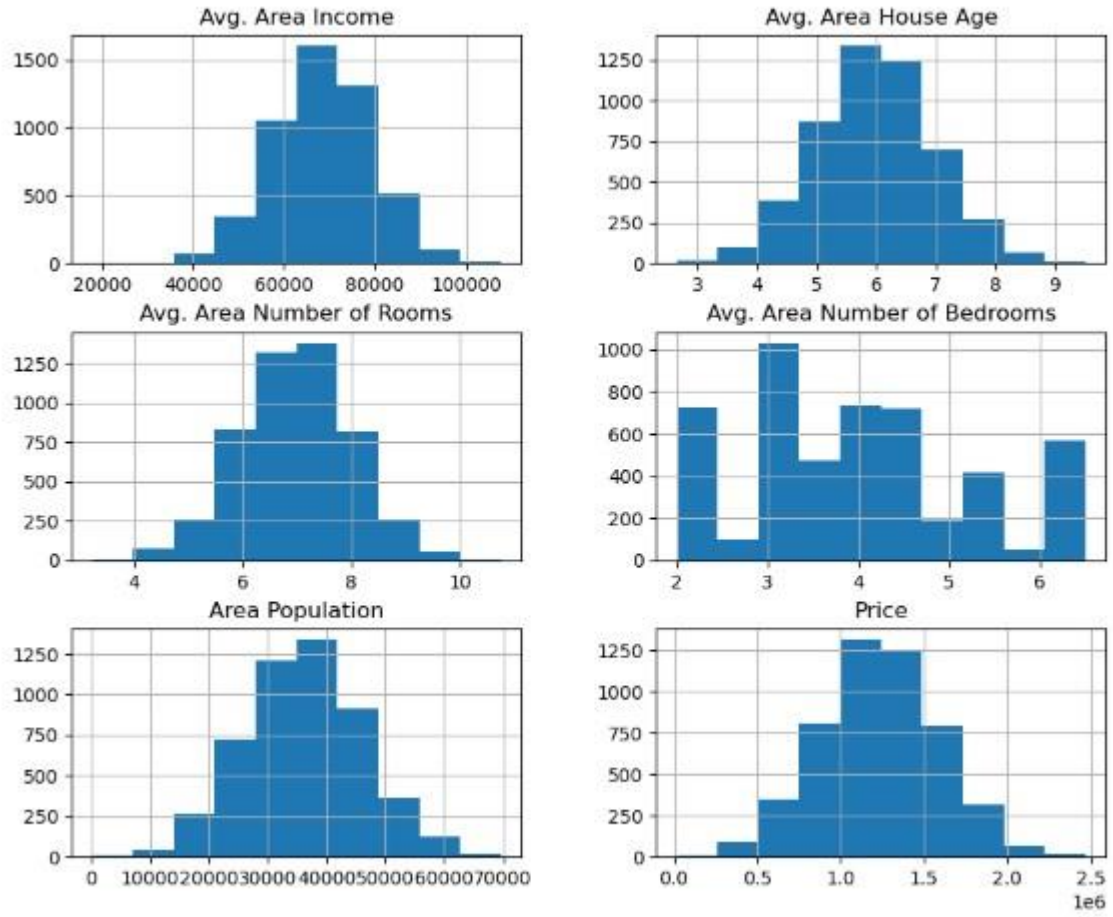



```
plt.figure(figsize=(12,8)) sns.pairplot(dataset)
<seaborn.axisgrid.PairGrid at 0x7f2b52c24430>
<Figure size 1200x800 with 0 Axes>
```



```
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

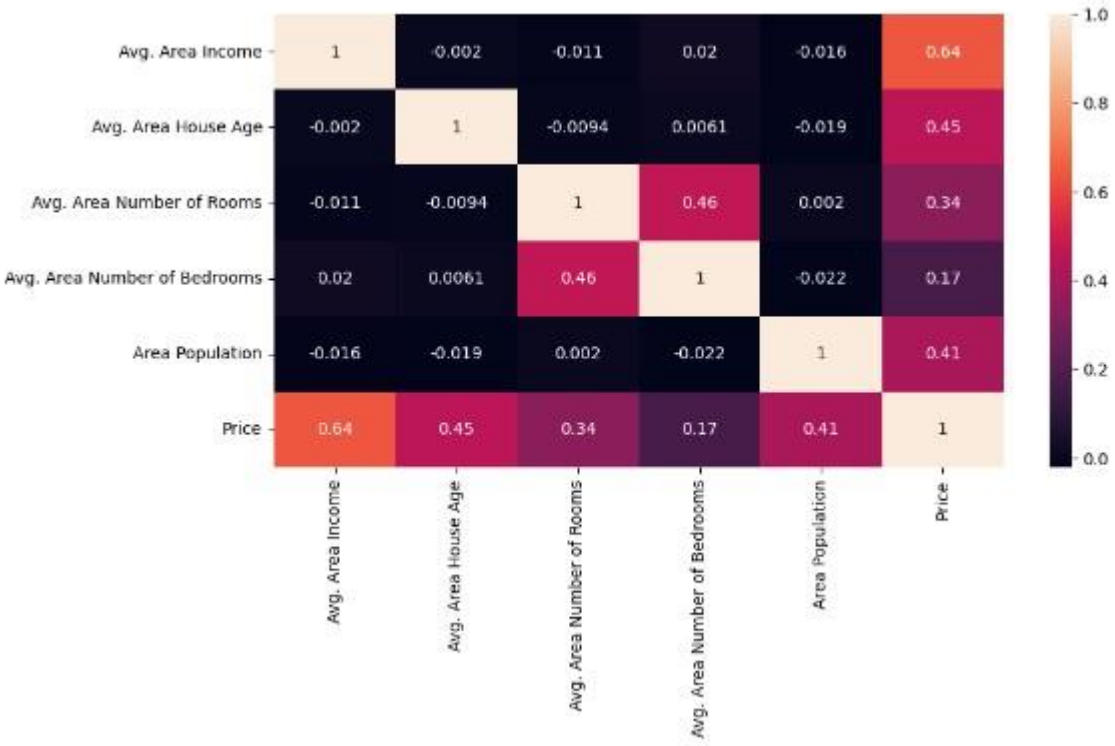
```
dataset.corr(numeric_only=True)
```

Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	0.171071

Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

```
plt.figure(figsize=(10,5))
sns.heatmap(dataset.corr(numeric_only = True), annot=True)
```

```
<Axes: >
```



Dividing Dataset in to features and target variable

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

Using Train Test Spli

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=101)
Y_train.head()
```

```
3413    1.305210e+06
1610    1.400961e+06
3459    1.048640e+06
4293    1.231157e+06
1039    1.391233e+06
Name: Price, dtype: float64
```

```
Y_train.shape
```

```
(4000,)
```

```
Y_test.head()
```

```
1718    1.251689e+06
2511    8.730483e+05
345     1.696978e+06
2521    1.063964e+06
54      9.487883e+05
Name: Price, dtype: float64
```

```
Y_test.shape
```

```
(1000,)
```

Standardizing the data

```
sc = StandardScaler()
X_train_scal = sc.fit_transform(X_train)
X_test_scal = sc.fit_transform(X_test)
```