**Predicting House Prices using MachineLearning**

**Import dependencies:**

import numpy as np

import pandas as pd

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

for filename in filenames:

print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt

import seaborn as sb

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

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

from sklearn import metrics

import matplotlib.pyplot as plt

**Loading Dataset:**

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

**Data Exploration:**

**dataset**

Avg. Area Income Avg. Area House Age Avg. Area Number of Room 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...

3 63345.240046 7.188236 5.586729 3.26 34310.242831 1.260617e+06

USS Barnett\nFPO AP 44820

4 59982.197226 5.0405557.839388 4.22 26354.10947 6.309435e+05

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 42991-3352

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...

data.head()

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...

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

**data.shape**

(5000, 7)

**data.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 Room 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

**data.isna().sum()**

Avg. Area Income 0

Avg. Area House Age 0

Avg. Area Number of Rooms 0

Avg. Area Number of Bedrooms 0

Area Population 0

Price 0

Address 0

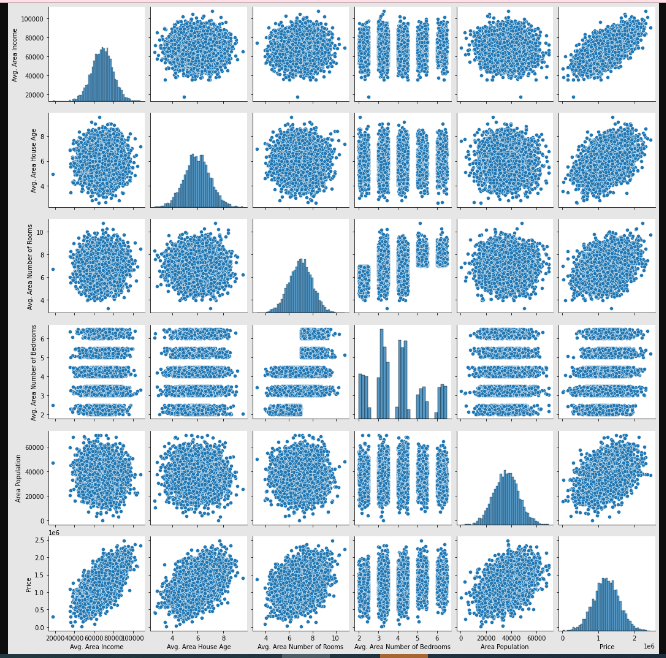
dtype: int64

**data.duplicated().sum()**

0

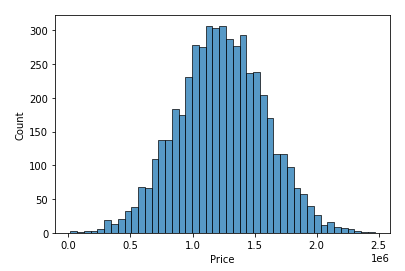
**sb.pairplot(data = data)**

/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118: self.\_figure.tight\_layout(\*args,\*\*kwarg)

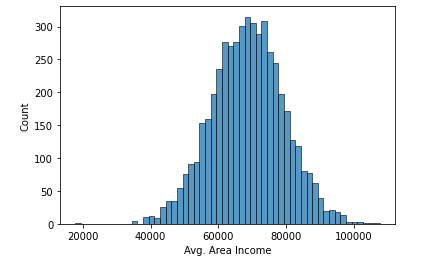
<seaborn.axisgrid.PairGrid at 0x79d70cabb5b0>

**Visualisation & Pre processor:**

**sb.histplot(x = data['Price']);**

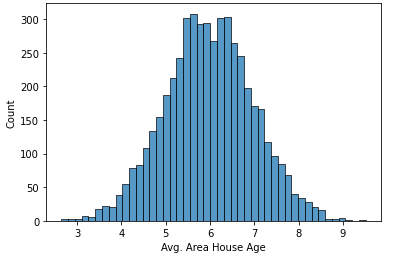


**sb.histplot(x = data['Avg. Area Income']);**

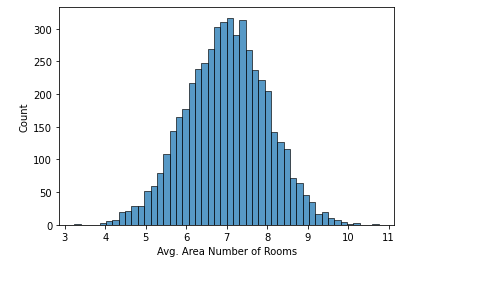


**sb.histplot(x = data['Avg. Area House Age'])**

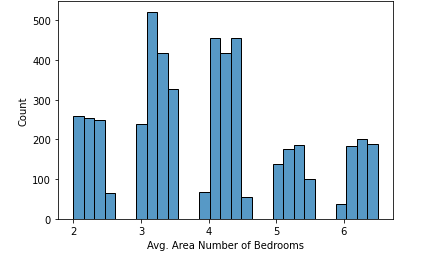
**<AxesSubplot:xlabel='Avg. Area House Age', ylabel='Count'>**



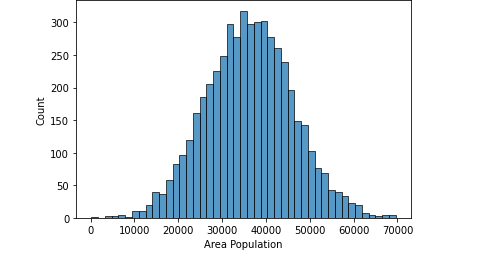
**sb.histplot(x = data['Avg. Area Number of Rooms']);**



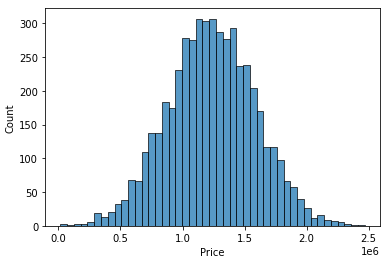
**sb.histplot(x = data['Avg. Area Number of Bedrooms']);**



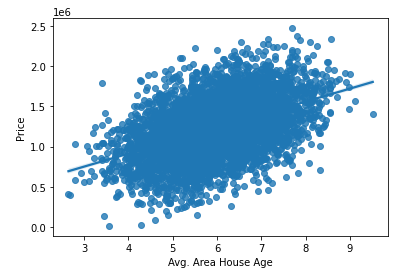
**sb.histplot(x = data['Area Population']);**



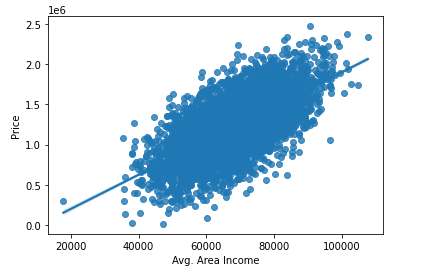
**sb.histplot(x = data['Price']);**



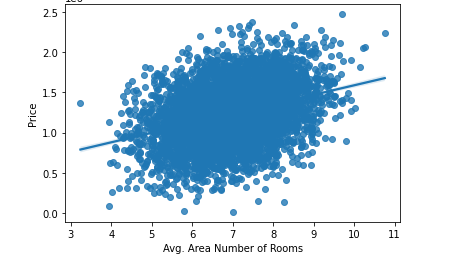
**sb.regplot(x = data['Avg. Area House Age'], y = data['Price']);**



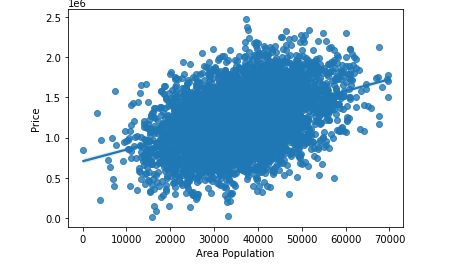
**sb.regplot(x = data['Avg. Area Income'], y = data['Price']);**



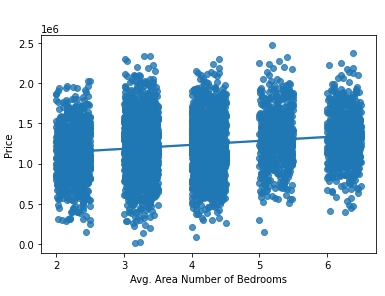
**sb.regplot(x = data['Avg. Area Number of Rooms'], y = data['Price']);**



**sb.regplot(x = data['Area Population'], y = data['Price']);**



**sb.regplot(x = data['Avg. Area Number of Bedrooms'], y = data['Price']);**



**Analysis:**

Price increases with all the variables

Price increases sharply with increase in Average Area Income

plt.figure(figsize = (15, 10))

sb.heatmap(data.corr(), annot = True, cmap = 'mako')

<AxesSubplot:>



**Random forest:**

y = data['Price']

X = data.drop(['Price', 'Address'], axis = 1)

# Random Forest

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

from sklearn.metrics import mean\_absolute\_error

from sklearn.metrics import mean\_squared\_error

from sklearn.ensemble import RandomForestRegressor

train\_X, val\_X, train\_y, val\_y = train\_test\_split(X, y, random\_state = 42)

model = RandomForestRegressor(random\_state = 1)

model.fit(train\_X, train\_y)

preds = model.predict(val\_X)

print("MAE: ", mean\_absolute\_error(preds, val\_y))

print("RMSE: ", np.sqrt(mean\_squared\_error(preds, val\_y)))

**output:**

MAE: 93812.37073246129

RMSE: 118380.48325186648

y=dataset.Price

features=['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Area Population']

X=dataset[features]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

regressor = RandomForestRegressor(n\_estimators=500, random\_state=0)

regressor.fit(X\_train, y\_train)

y\_pred = regressor.predict(X\_test)

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))

**output:**

Mean Absolute Error: 97816.09821376632

Mean Squared Error: 14827659280.278809

Root Mean Squared Error: 121768.87648442358

#visualizing the predicted value

fig, ax = plt.subplots()

ax.scatter(y\_test, y\_pred, edgecolors=(0, 0, 0))

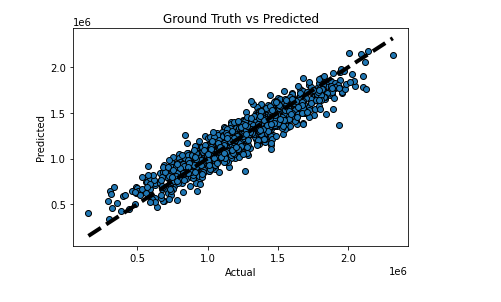
ax.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'k--', lw=4)

ax.set\_xlabel('Actual')

ax.set\_ylabel('Predicted')

ax.set\_title("Ground Truth vs Predicted")

plt.show()



model\_rf = RandomForestRegressor(n\_estimators=50)

model\_rf.fit(X\_train\_scal, Y\_train)

RandomForestRegressor

RandomForestRegressor(n\_estimators=50)

**Predicting Prices**:

Prediction4 = model\_rf.predict(X\_test\_scal)

**Evaluation of Predicted Data :**

plt.figure(figsize=(12,6))

plt.plot(np.arange(len(Y\_test)), Y\_test, label='Actual Trend')

plt.plot(np.arange(len(Y\_test)), Prediction4, label='Predicted Trend')

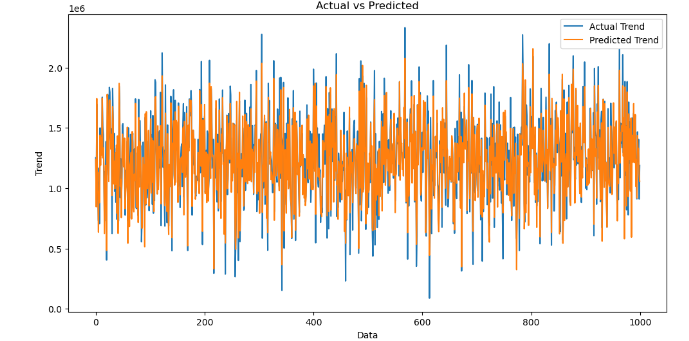
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

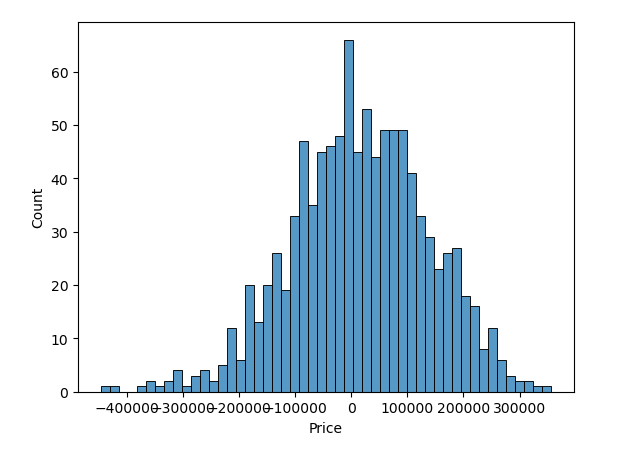
plt.title('Actual vs Predicted')

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



sns.histplot((Y\_test-Prediction4), bins=50)

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



print(r2\_score(Y\_test, Prediction2))

print(mean\_absolute\_error(Y\_test, Prediction2))

print(mean\_squared\_error(Y\_test, Prediction2))

**output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034

**XGboost Regressor:**

model\_xg = xg.XGBRegressor()

model\_xg.fit(X\_train\_scal, Y\_train)

**XGBRegressor**

XGBRegressor(base\_score=None, booster=None, callbacks=None,

colsample\_bylevel=None, colsample\_bynode=None,

colsample\_bytree=None, early\_stopping\_rounds=None,

enable\_categorical=False, eval\_metric=None, feature\_types=None,

gamma=None, gpu\_id=None, grow\_policy=None, importance\_type=None,

interaction\_constraints=None, learning\_rate=None, max\_bin=None,

max\_cat\_threshold=None, max\_cat\_to\_onehot=None,

max\_delta\_step=None, max\_depth=None, max\_leaves=None,

min\_child\_weight=None, missing=nan, monotone\_constraints=None,

n\_estimators=100, n\_jobs=None, num\_parallel\_tree=None,

predictor=None, random\_state=None, ...)

**Predicting Prices**

Prediction5 = model\_xg.predict(X\_test\_scal)

**Evaluation of Predicted Data:**

plt.figure(figsize=(12,6))

plt.plot(np.arange(len(Y\_test)), Y\_test, label='Actual Trend')

plt.plot(np.arange(len(Y\_test)), Prediction5, label='Predicted Trend')

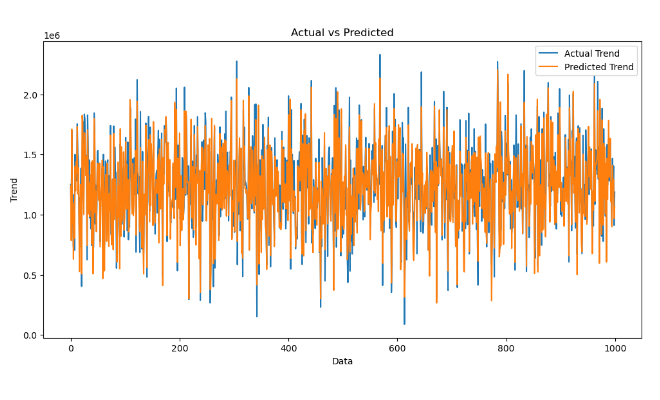
plt.xlabel('Data')

plt.ylabel('Trend')

plt.legend()

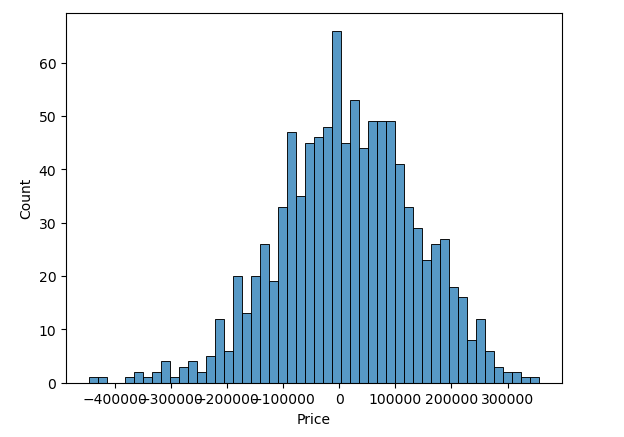
plt.title('Actual vs Predicted')

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



sns.histplot((Y\_test-Prediction4), bins=50)

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



print(r2\_score(Y\_test, Prediction2))

print(mean\_absolute\_error(Y\_test, Prediction2))

print(mean\_squared\_error(Y\_test, Prediction2))

**output:**

-0.0006222175925689744

286137.81086908665

128209033251.4034