HOUSE PRICE PREDICTION

```
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

dataset=pd.read_csv("USA_Housing.csv")
print(dataset)
```

```
      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

3         63345.240046             7.188236                   5.586729

4         59982.197226             5.040555                   7.839388

...                ...                  ...                        ...

4995      60567.944140             7.830362                   6.137356

4996      78491.275435             6.999135                   6.576763

4997      63390.686886             7.250591                   4.805081

4998      68001.331235             5.534388                   7.130144

4999      65510.581804             5.992305                   6.792336


      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
3                             3.26     34310.242831  1.260617e+06
4                             4.23     26354.109472  6.309435e+05
...                            ...              ...           ...
4995                          3.46     22837.361035  1.060194e+06
4996                          4.02     25616.115489  1.482618e+06
4997                          2.13     33266.145490  1.030730e+06
```

```
4998                          5.44     42625.620156  1.198657e+06
4999                          4.07     46501.283803  1.298950e+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...
3                           USS Barnett\nFPO AP 44820
4                          USNS Raymond\nFPO AE 09386
...                                                ...
4995                 USNS Williams\nFPO AP 30153-7653
4996              PSC 9258, Box 8489\nAPO AA 42991-3352
4997  4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998                          USS Wallace\nFPO AE 73316
4999  37778 George Ridges Apt. 509\nEast Holly, NV 2...

[5000 rows x 7 columns]

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  \
count       5000.000000          5000.000000
5000.000000
mean       68583.108984             5.977222
6.987792
std        10657.991214             0.991456
1.005833
min        17796.631190             2.644304
3.236194
25%        61480.562388             5.322283
6.299250
50%        68804.286404             5.970429
```

```
7.002902
75%        75783.338666             6.650808
7.665871
max       107701.748378             9.519088
10.759588
```

```
       Avg. Area Number of Bedrooms  Area Population         Price
count                   5000.000000      5000.000000  5.000000e+03
mean                       3.981330     36163.516039  1.232073e+06
std                        1.234137      9925.650114  3.531176e+05
min                        2.000000       172.610686  1.593866e+04
25%                        3.140000     29403.928702  9.975771e+05
50%                        4.050000     36199.406689  1.232669e+06
75%                        4.490000     42861.290769  1.471210e+06
max                        6.500000     69621.713378  2.469066e+06
```

```
dataset.head()
```

```
   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
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388
```

```
   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
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05
```

```
                                             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
```

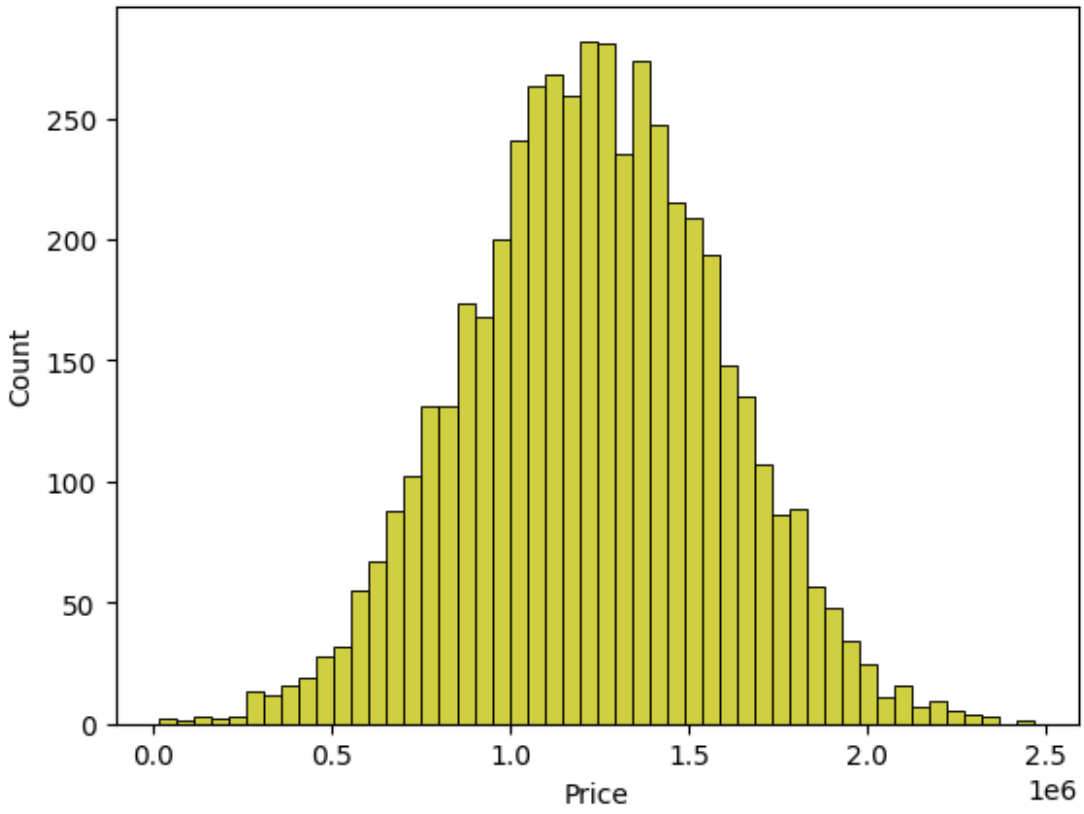
```
dataset.tail()
```

```
      Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms
\
4995      60567.944140             7.830362                   6.137356

4996      78491.275435             6.999135                   6.576763

4997      63390.686886             7.250591                   4.805081

4998      68001.331235             5.534388                   7.130144
```

```
4999      65510.581804              5.992305                     6.792336


      Avg. Area Number of Bedrooms  Area Population         Price  \
4995                          3.46     22837.361035  1.060194e+06
4996                          4.02     25616.115489  1.482618e+06
4997                          2.13     33266.145490  1.030730e+06
4998                          5.44     42625.620156  1.198657e+06
4999                          4.07     46501.283803  1.298950e+06

                                                  Address
4995                  USNS Williams\nFPO AP 30153-7653
4996              PSC 9258, Box 8489\nAPO AA 42991-3352
4997  4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998                          USS Wallace\nFPO AE 73316
4999  37778 George Ridges Apt. 509\nEast Holly, NV 2...
```

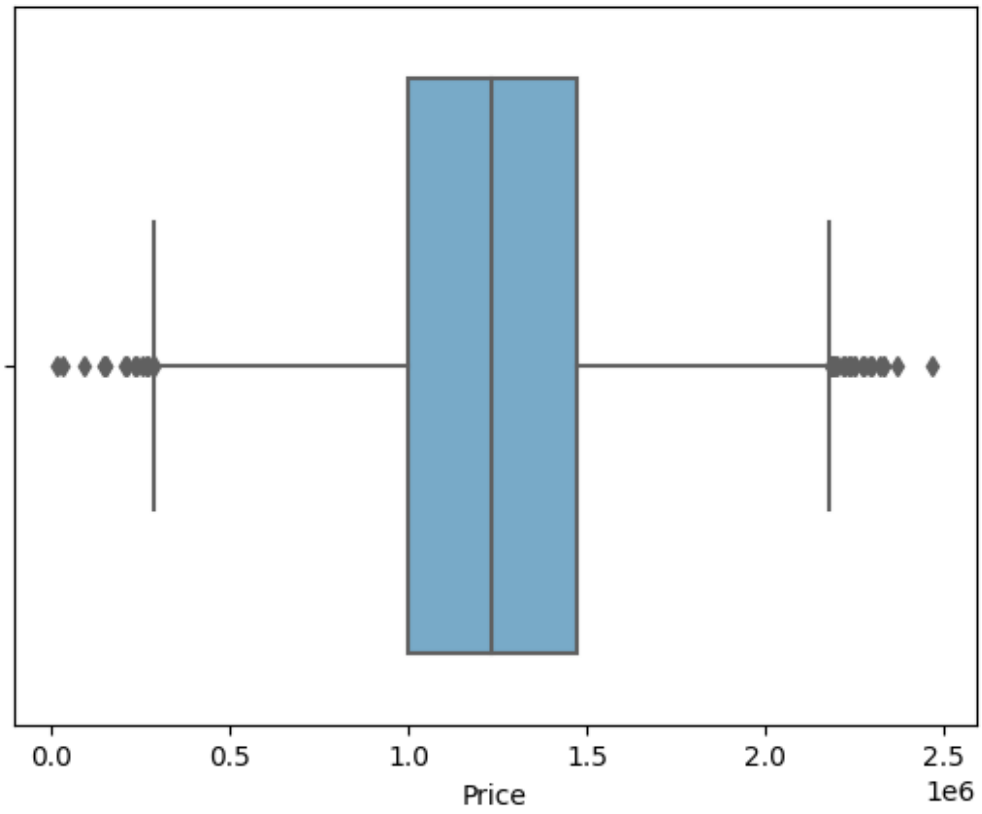
```
sns.histplot(dataset,x='Price',bins=50,color='y')
```

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



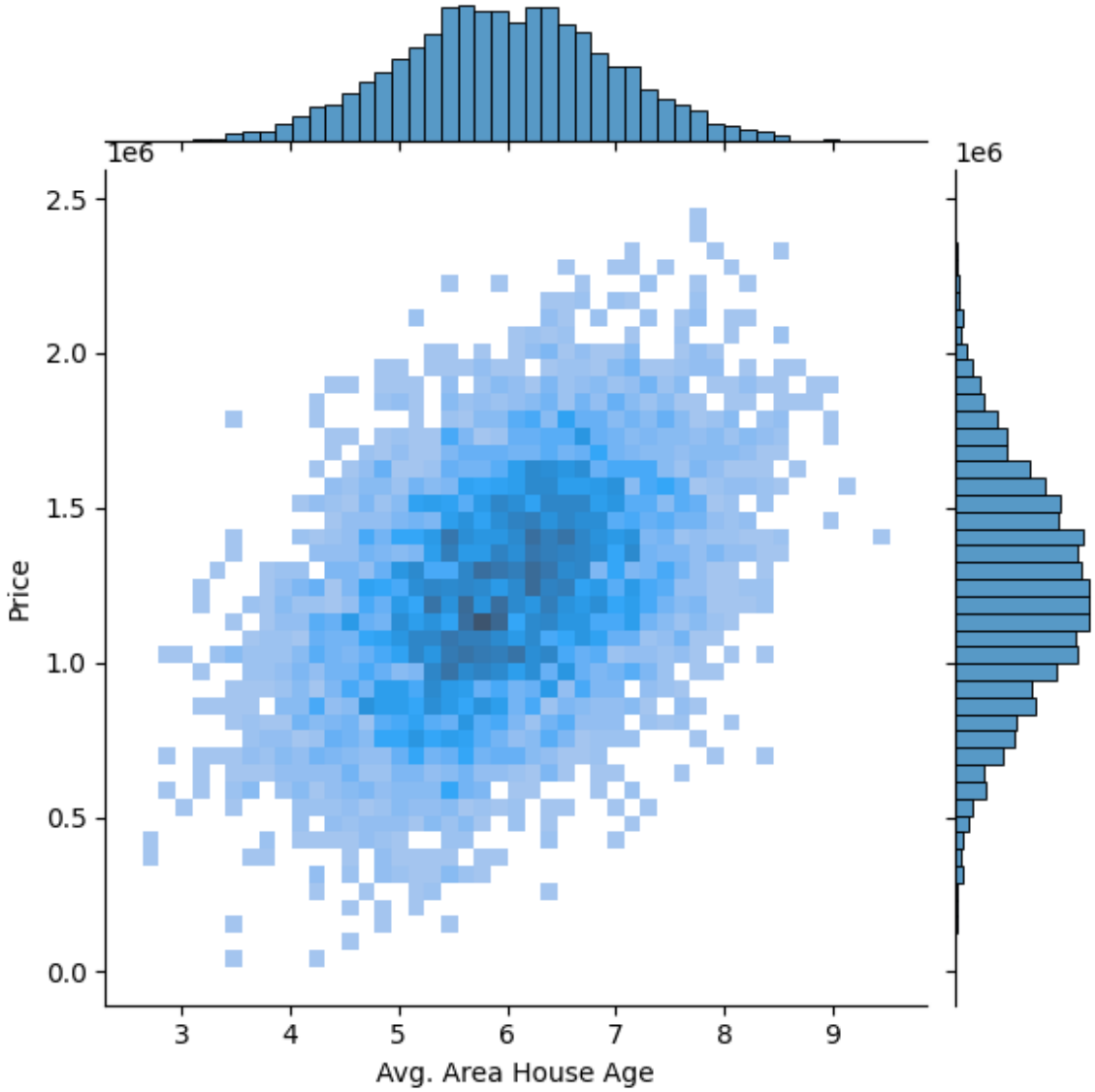

```
sns.boxplot(dataset, x='Price', palette='Blues')
```

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



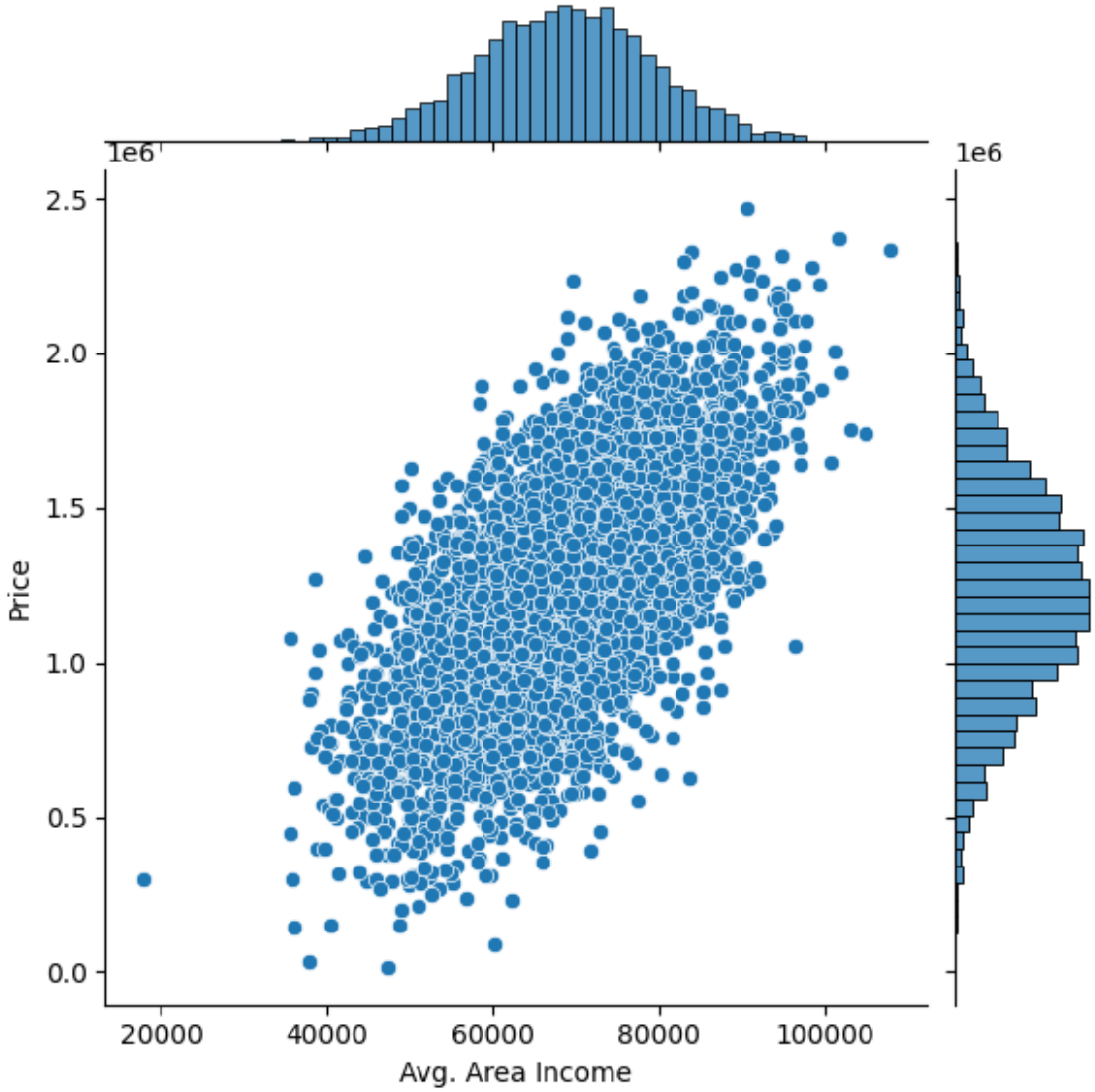

```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price',
kind='hist')
```

```
<seaborn.axisgrid.JointGrid at 0x79e7dca76320>
```




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

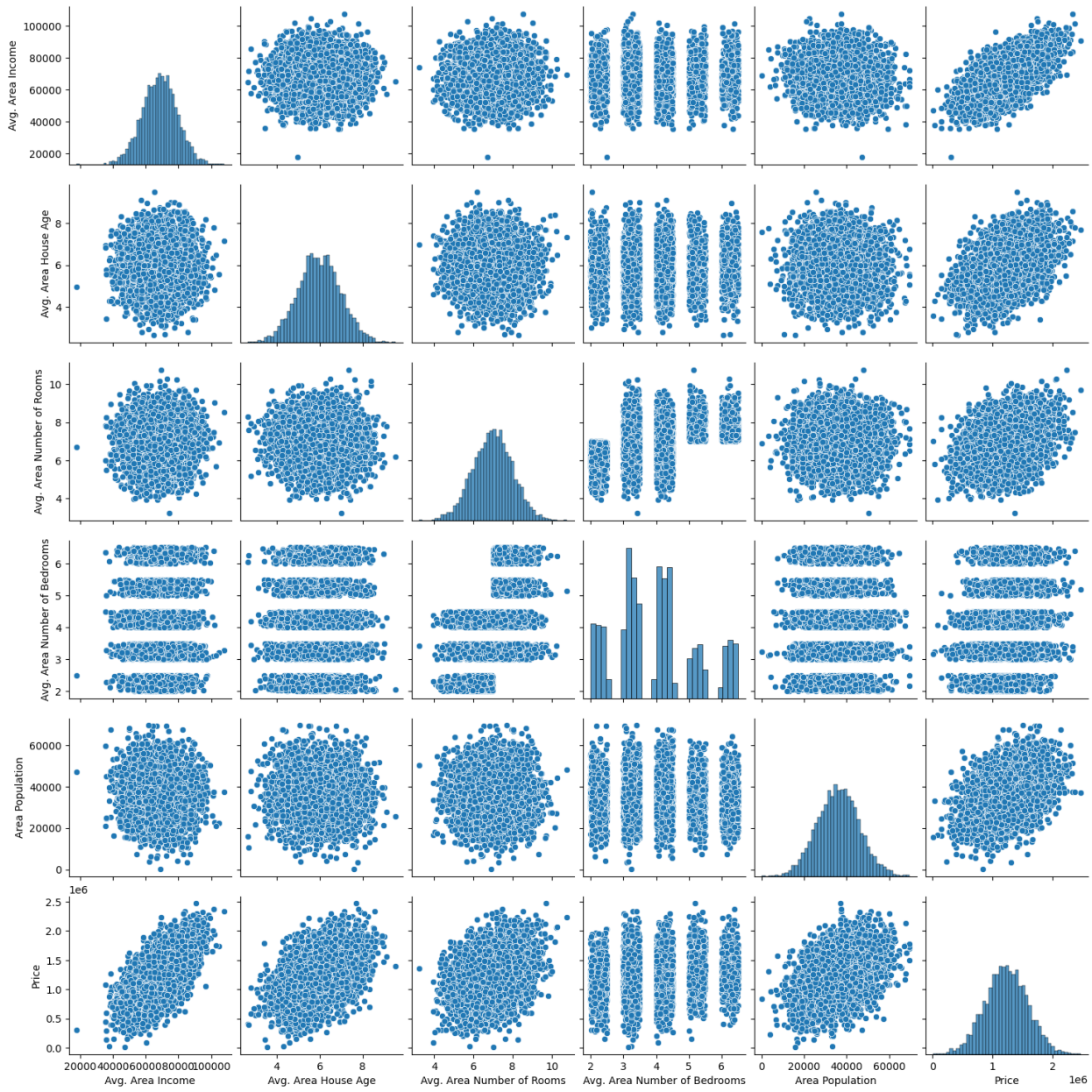
```
<seaborn.axisgrid.JointGrid at 0x79e812768520>
```




```
plt.figure(figsize=(12,8))
sns.pairplot(dataset)
```

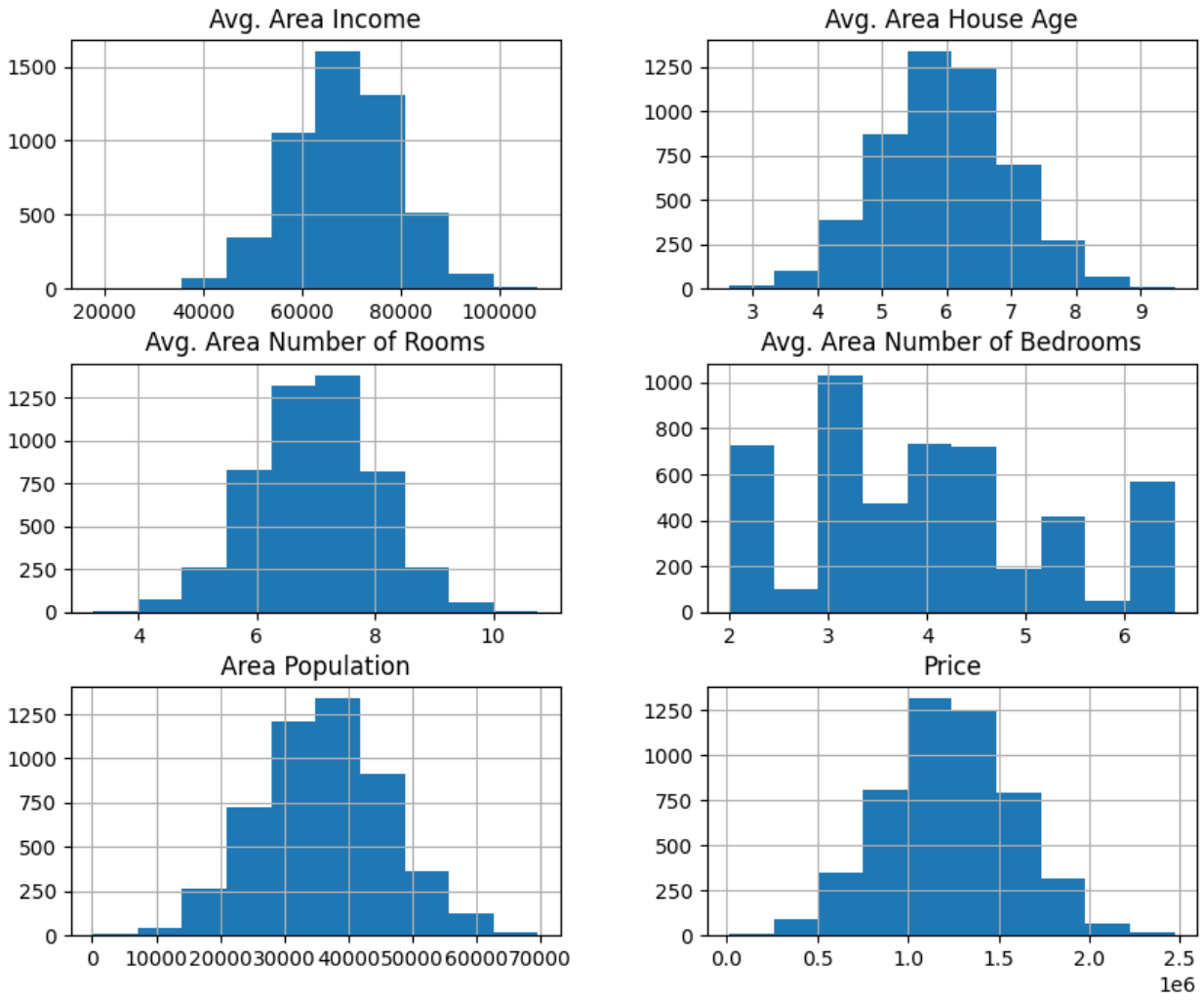
```
<seaborn.axisgrid.PairGrid at 0x79e7da199750>
```

```
<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)
```




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

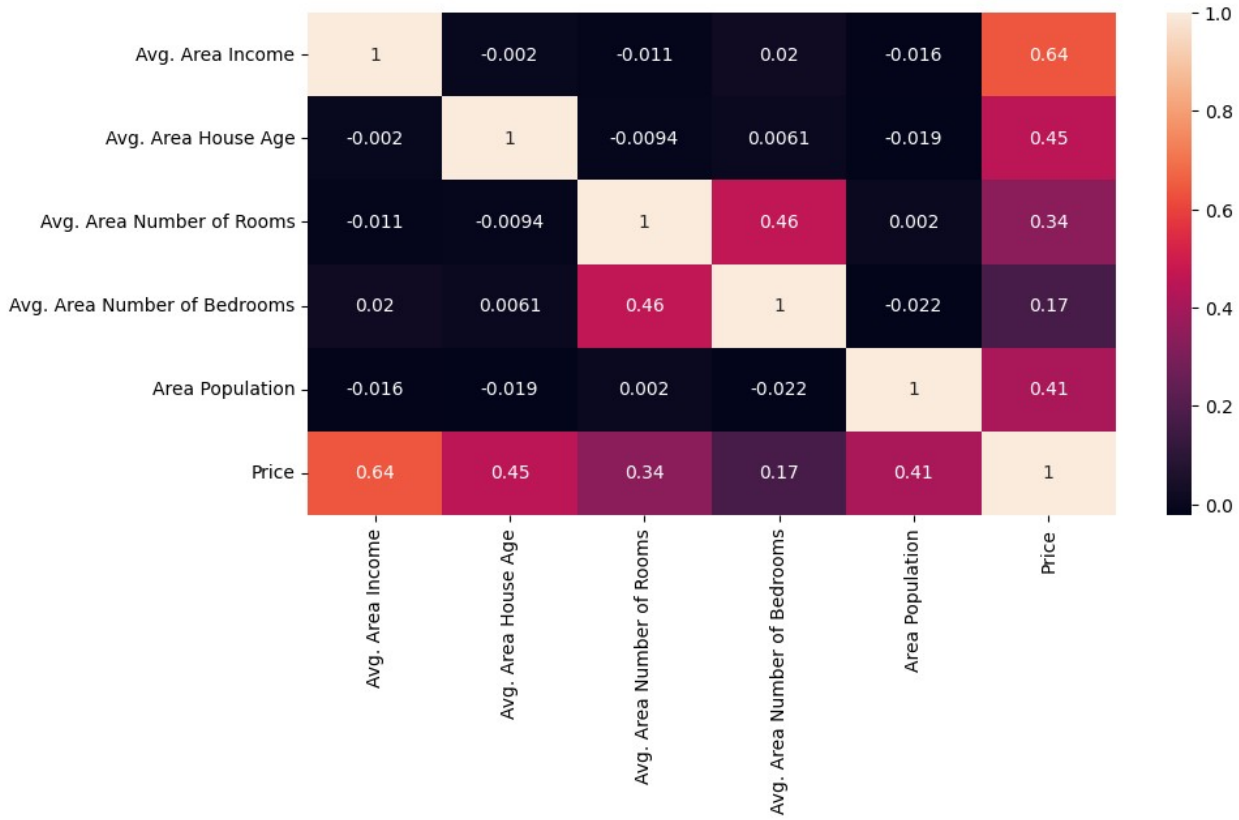
	Avg. Area Income	Avg. Area House Age
Avg. Area Income	1.000000	-0.002007
Avg. Area House Age	-0.002007	1.000000
Avg. Area Number of Rooms	-0.011032	-0.009428
Avg. Area Number of Bedrooms	0.019788	0.006149
Area Population	-0.016234	-0.018743
Price	0.639734	0.452543

	Avg. Area Number of Rooms
Avg. Area Income	-0.011032
Avg. Area House Age	-0.009428
Avg. Area Number of Rooms	1.000000
Avg. Area Number of Bedrooms	0.462695
Area Population	0.002040
Price	0.335664

```
                               Avg. Area Number of Bedrooms  Area
Population  \
Avg. Area Income                                0.019788        -
0.016234
Avg. Area House Age                             0.006149        -
0.018743
Avg. Area Number of Rooms                       0.462695
0.002040
Avg. Area Number of Bedrooms                    1.000000        -
0.022168
Area Population                                -0.022168
1.000000
Price                                           0.171071
0.408556

                                  Price
Avg. Area Income               0.639734
Avg. Area House Age            0.452543
Avg. Area Number of Rooms      0.335664
Avg. Area Number of Bedrooms   0.171071
Area Population                0.408556
Price                          1.000000
```

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

```
<Axes: >
```



FEATURE SELECTION

```
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 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    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)
```

Model Building and Evaluation

```
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
```

Model 1 - Linear Regression

```
model_lr=LinearRegression()
```

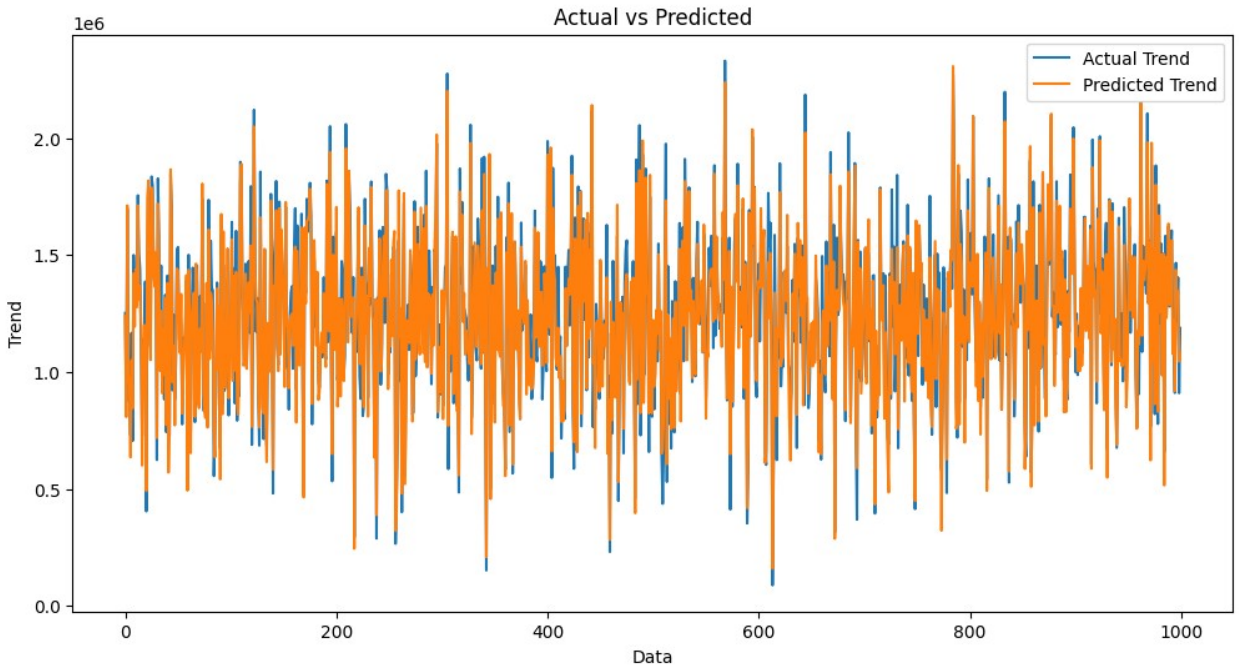
```
model_lr.fit(X_train_scal, Y_train)
```

```
LinearRegression()
```

```
Prediction1 = model_lr.predict(X_test_scal)
```

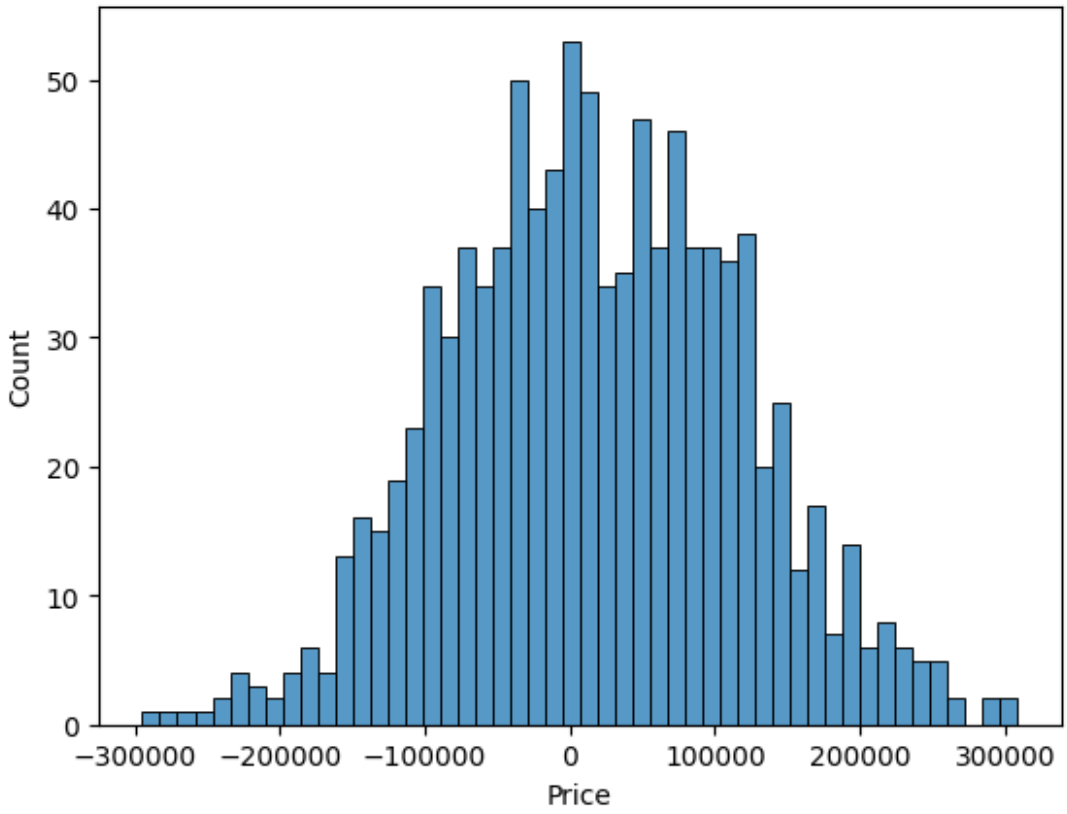
```
    plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
```

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




```
sns.histplot((Y_test-Prediction1), bins=50)
```

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




```
print(r2_score(Y_test, Prediction1))
print(mean_absolute_error(Y_test, Prediction1))
print(mean_squared_error(Y_test, Prediction1))
```

```
0.9182928179392918
82295.49779231755
10469084772.975954
```

Model 2 - Support Vector Regressor

```
model_svr = SVR()

model_svr.fit(X_train_scal, Y_train)
```

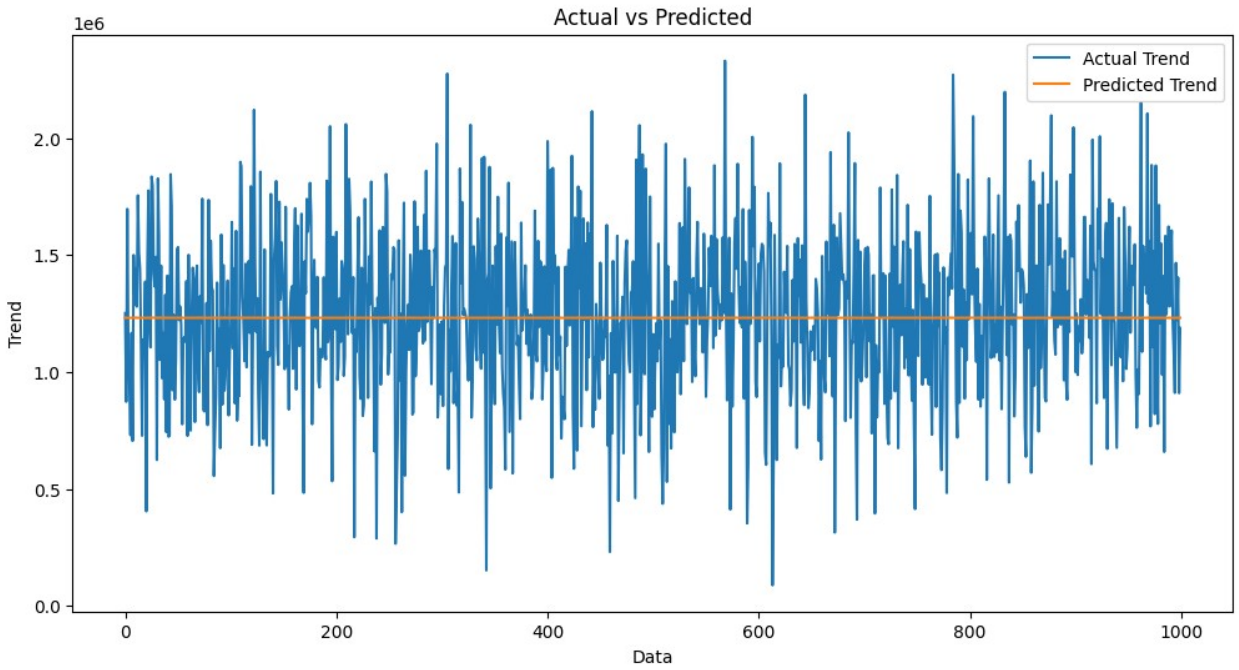
```
SVR()
```

```
Prediction2 = model_svr.predict(X_test_scal)
```

```
    plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
```

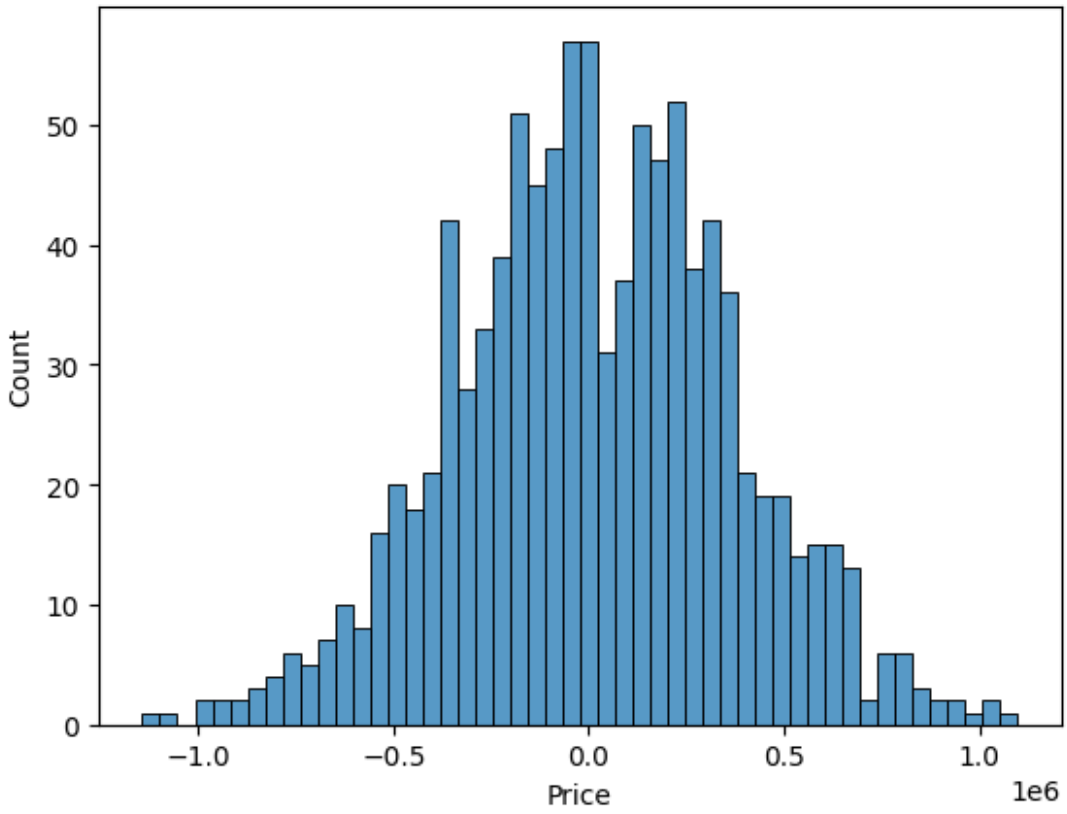
```
    plt.legend()
    plt.title('Actual vs Predicted')
```

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




```
sns.histplot((Y_test-Prediction2), 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))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 3 - Lasso Regression

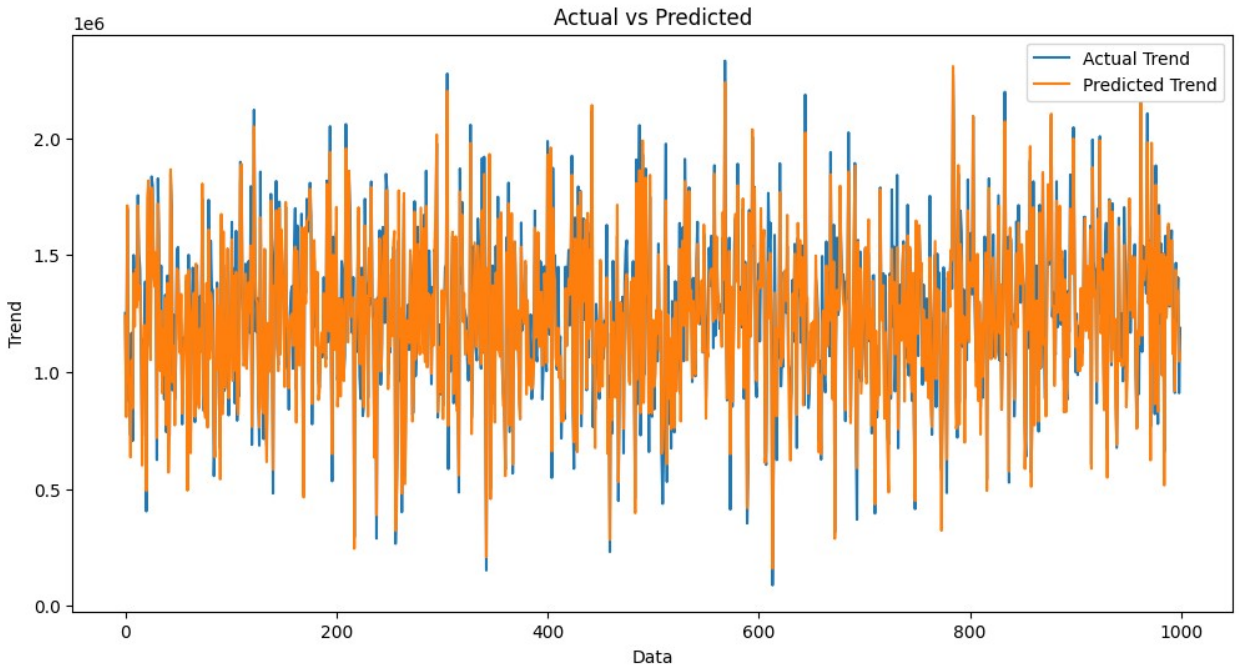
```
model_lar = Lasso(alpha=1)

model_lar.fit(X_train_scal,Y_train)
```

```
Lasso(alpha=1)
```

```
Prediction3 = model_lar.predict(X_test_scal)

    plt.figure(figsize=(12,6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Y_test)), Prediction3, label='Predicted
Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
```

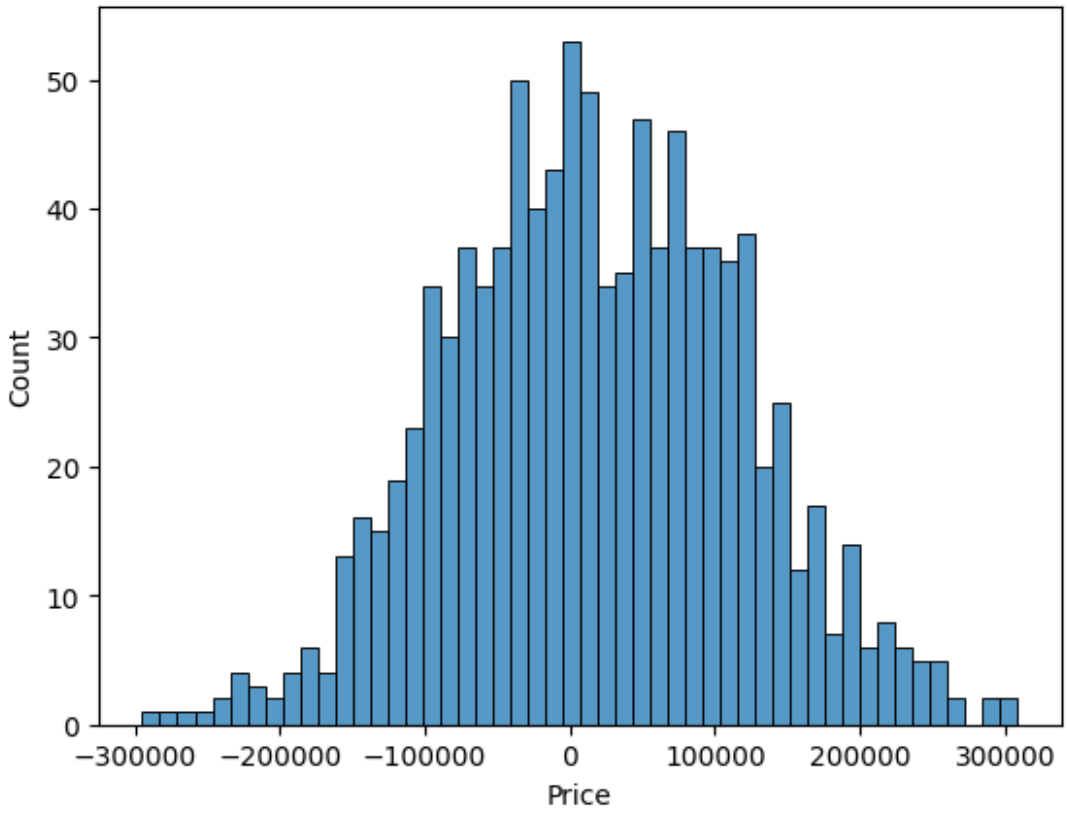
```
    plt.legend()
    plt.title('Actual vs Predicted')
```

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




```
sns.histplot((Y_test-Prediction3), 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))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 4 - Random Forest Regressor

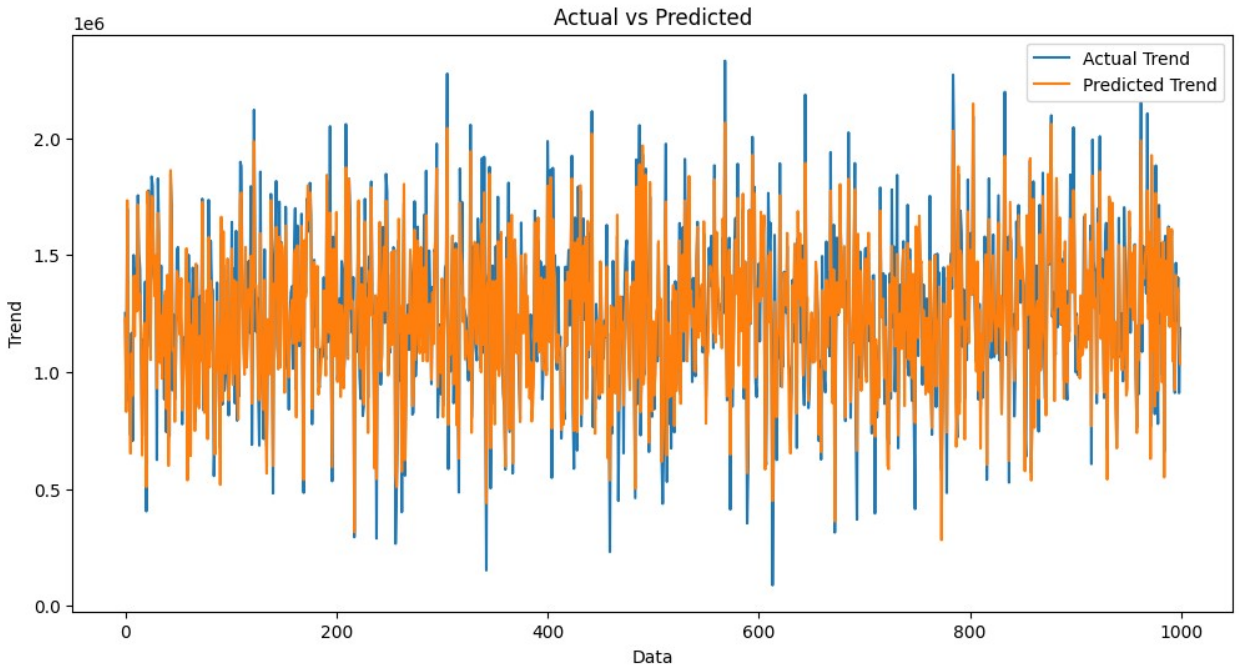
```
model_rf = RandomForestRegressor(n_estimators=50)

model_rf.fit(X_train_scal, Y_train)
```

```
RandomForestRegressor(n_estimators=50)
```

```
Prediction4 = model_rf.predict(X_test_scal)

    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')
```

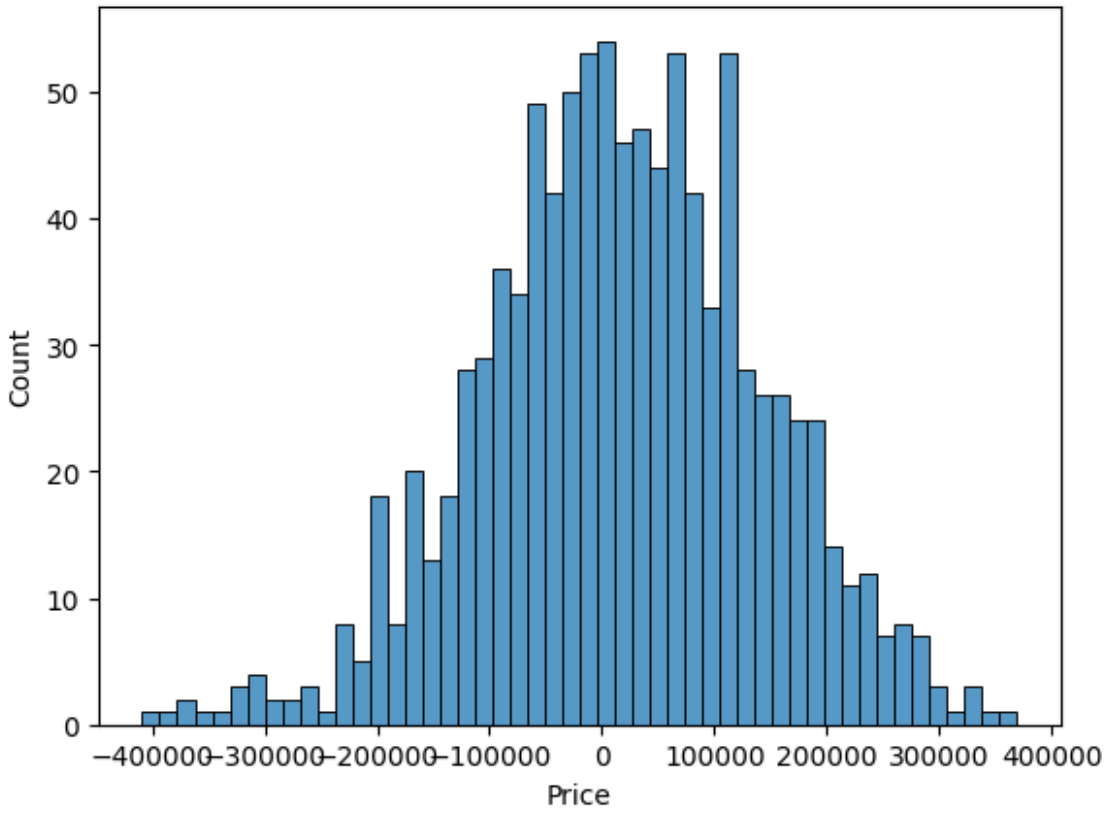
```
    plt.legend()
    plt.title('Actual vs Predicted')
```

```
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))
```

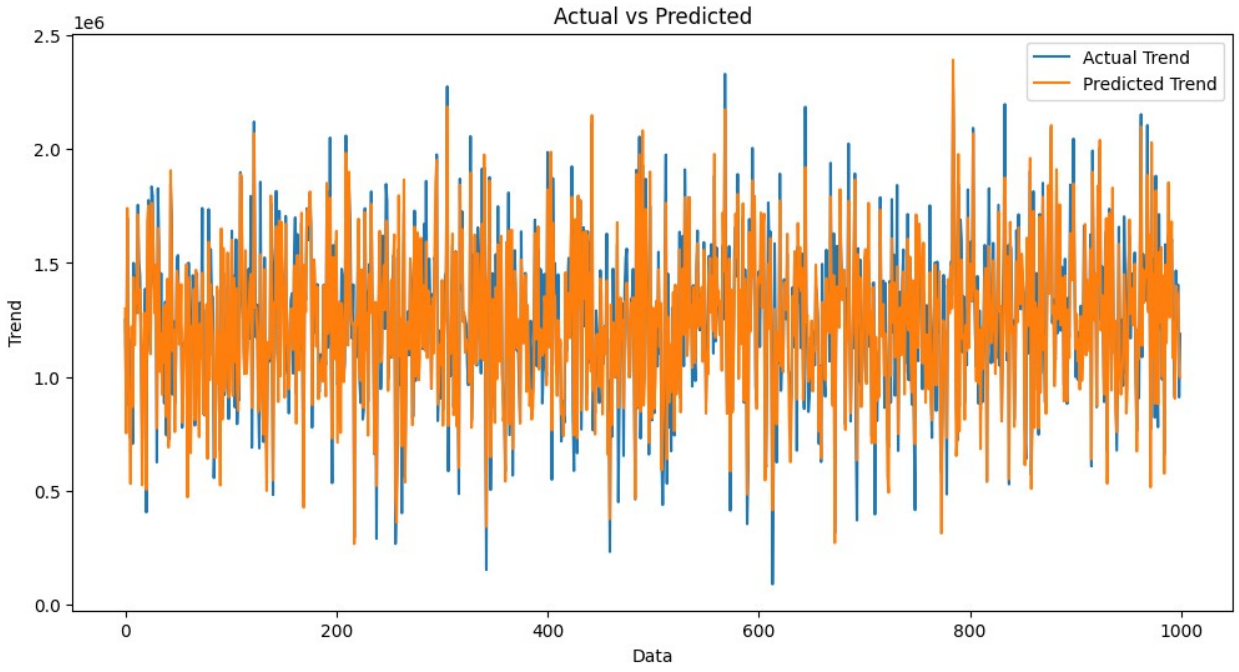
```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Model 5 - XGboost Regressor

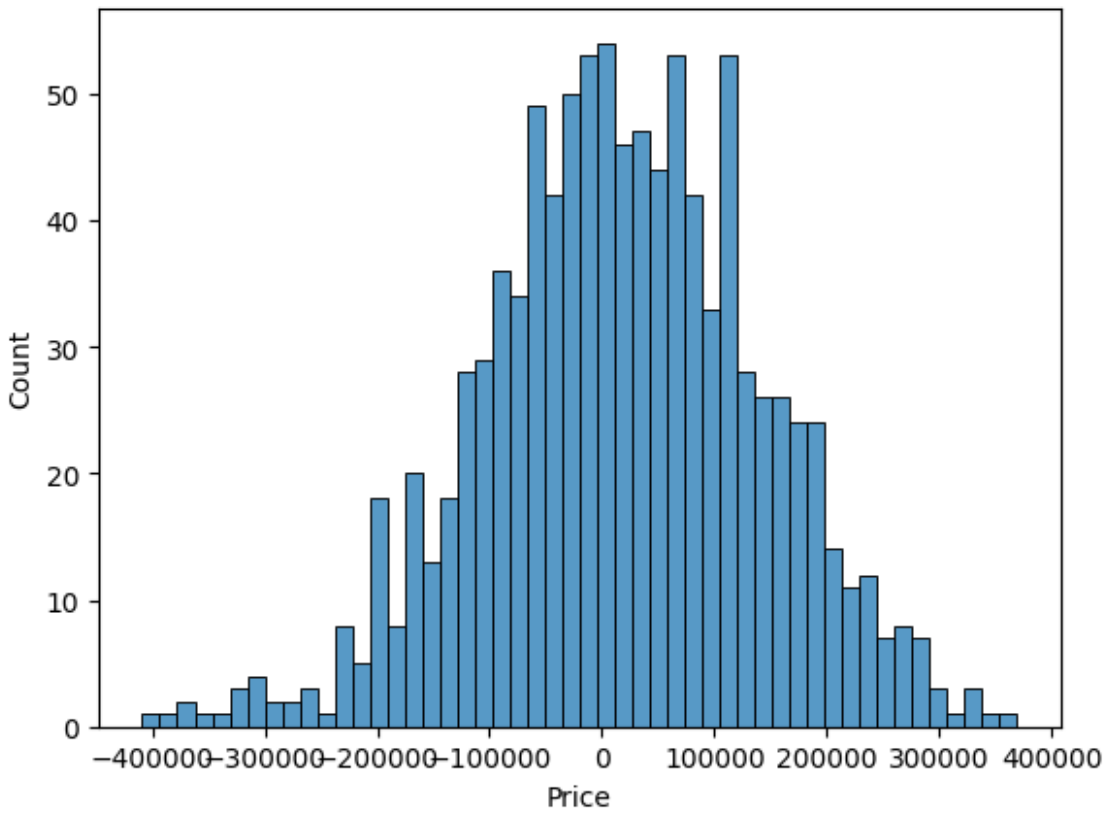
```
model_xg = xg.XGBRegressor()

model_xg.fit(X_train_scal, Y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None,
early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None,
feature_types=None,
             gamma=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,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)

Prediction5 = model_xg.predict(X_test_scal)

    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))
```

```
-0.0006222175925689744
286137.81086908665
128209033251.4034
```

Linear Regression is giving us best Accuracy