# DATE - 28/10/2023

# PHASE - IV

# TEAM ID - 696

# PROJECT TITLE - House Pricing forecasting using ML

## Importing Dependencies

In [2]:
```
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 matplotlib.pyplot as plt
import tkinter as tk
import random
import requests
import scipy
```

## Loading Dataset

In [3]:
```
dataset = pd.read_csv("USA_Housing.csv")
```

# Data importing

In [4]:
```
dataset
```

Out[4]:

| | Avg. Area Income | Avg. Area House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area Population | Price | Addr |
|---|---|---|---|---|---|---|---|
| **0** | 79545.458574 | 5.682861 | 7.009188 | 4.09 | 23086.800503 | 1.059034e+06 | 208 Michael Ferry 674\nLaurabury, 370 |
| **1** | 79248.642455 | 6.002900 | 6.730821 | 3.09 | 40173.072174 | 1.505891e+06 | 188 Johnson Vi Suite 079\nL Kathleen, C |
| **2** | 61287.067179 | 5.865890 | 8.512727 | 5.13 | 36882.159400 | 1.058988e+06 | 9127 Elizab Stravenue\nDanielto WI 0648 |
| **3** | 63345.240046 | 7.188236 | 5.586729 | 3.26 | 34310.242831 | 1.260617e+06 | USS Barnett\nFPO 44 |
| **4** | 59982.197226 | 5.040555 | 7.839388 | 4.23 | 26354.109472 | 6.309435e+05 | USNS Raymond\nF AE 09 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **4995** | 60567.944140 | 7.830362 | 6.137356 | 3.46 | 22837.361035 | 1.060194e+06 | USNS Williams\nF AP 30153-7 |
| **4996** | 78491.275435 | 6.999135 | 6.576763 | 4.02 | 25616.115489 | 1.482618e+06 | PSC 9258, 8489\nAPO AA 429 3 |
| **4997** | 63390.686886 | 7.250591 | 4.805081 | 2.13 | 33266.145490 | 1.030730e+06 | 4215 Tracy Gar Suite 076\nJoshuala VA 0 |
| **4998** | 68001.331235 | 5.534388 | 7.130144 | 5.44 | 42625.620156 | 1.198657e+06 | USS Wallace\nFPO 73 |
| **4999** | 65510.581804 | 5.992305 | 6.792336 | 4.07 | 46501.283803 | 1.298950e+06 | 37778 George Rid Apt. 509\nEast H NV |

5000 rows × 7 columns

In [127]:
```
dataset.head(6)
```

Out[127]:

| | 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 |
| **5** | 80175.754159 | 4.988408 | 6.104512 | 4.04 | 26748.428425 | 1.068138e+06 | 06039 Jennifer Islands Apt. 443\nTracyport, KS... |

In [6]:
```
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
```

In [7]:
```
dataset.describe()
```

Out[7]:

| | 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 |

In [8]:
```
dataset.columns
```

Out[8]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

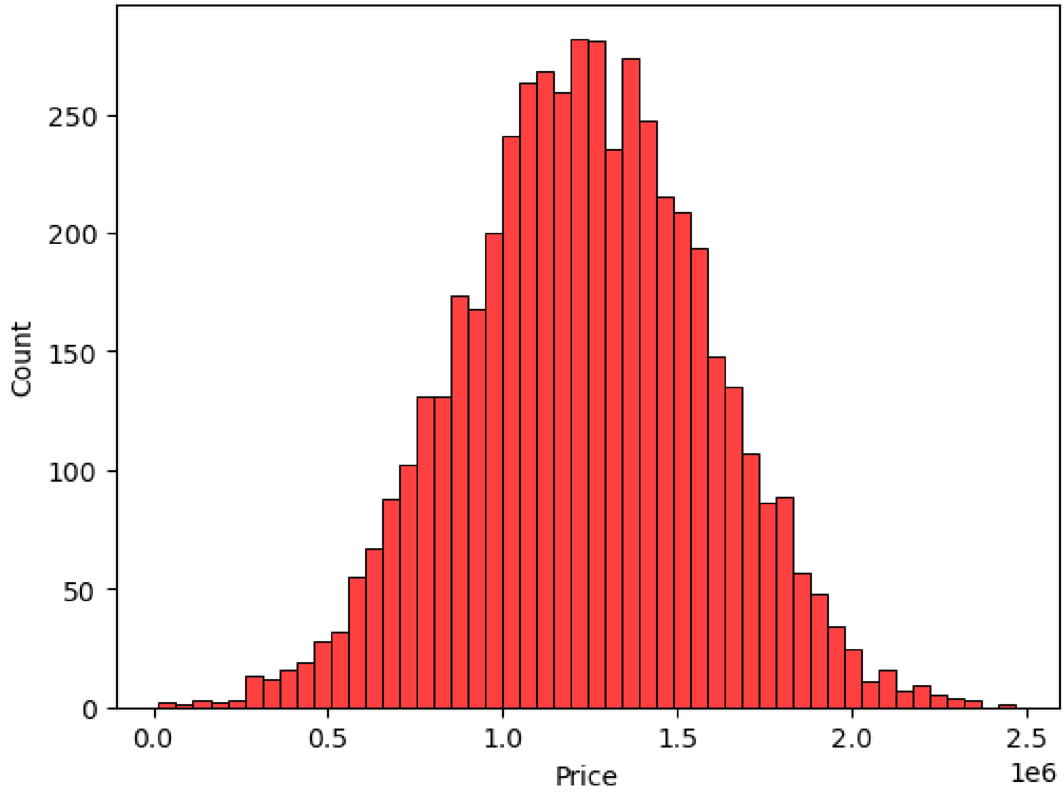
In [163]:
```
import pandas as pd
first_row = dataset.iloc[0]
specific_row = dataset.iloc[5]
print("First row:")
print(first_row)

print("\nSpecific row:")
print(specific_row)
```

```
First row:
Avg. Area Income                                                         79545.4585
74
Avg. Area House Age                                                          5.6828
61
Avg. Area Number of Rooms                                                    7.0091
88
Avg. Area Number of Bedrooms                                                     4.
09
Area Population                                                          23086.8005
03
Price                                                                   1059033.557
87
Address                         208 Michael Ferry Apt. 674\nLaurabury, NE 370
1...
Name: 0, dtype: object

Specific row:
Avg. Area Income                                                         80175.7541
59
Avg. Area House Age                                                          4.9884
08
Avg. Area Number of Rooms                                                    6.1045
12
Avg. Area Number of Bedrooms                                                     4.
04
Area Population                                                          26748.4284
25
Price                                                                  1068138.0743
94
Address                         06039 Jennifer Islands Apt. 443\nTracyport, K
S...
Name: 5, dtype: object
```

# Pre-Processing and Visualisation of Data

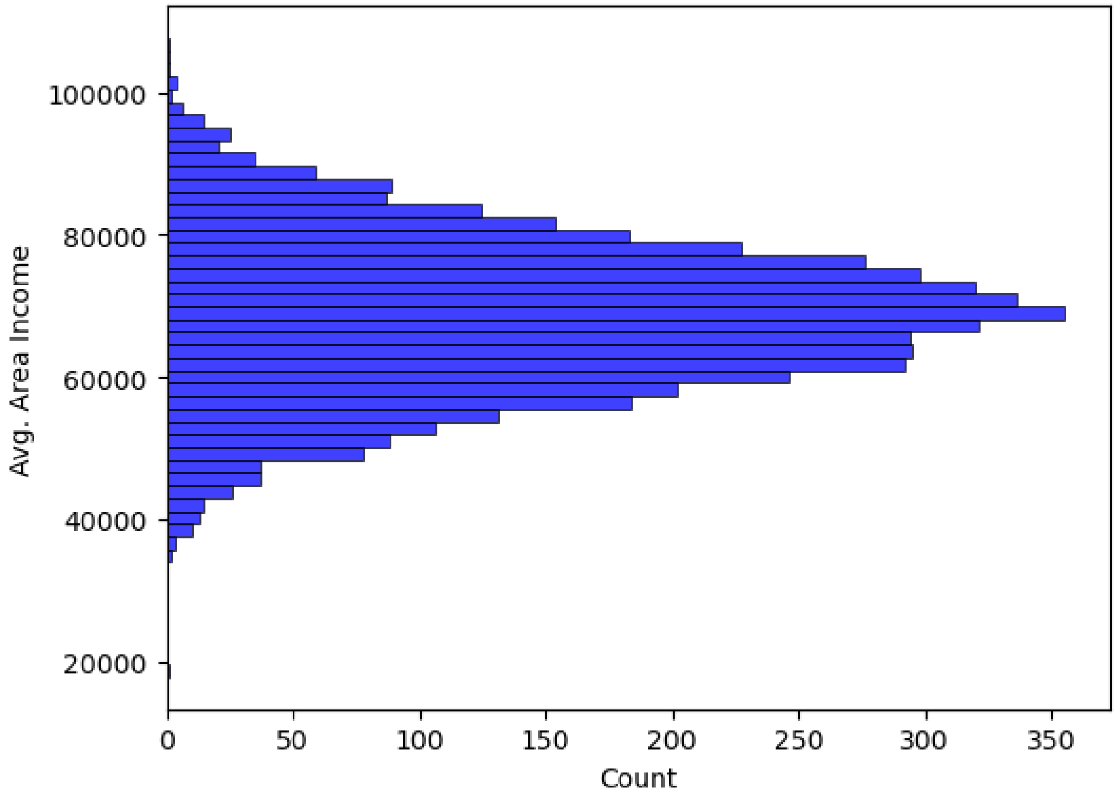
In [9]:
```
sns.histplot(dataset, x='Price', bins=50, color='r')
```

Out[9]: <Axes: xlabel='Price', ylabel='Count'>

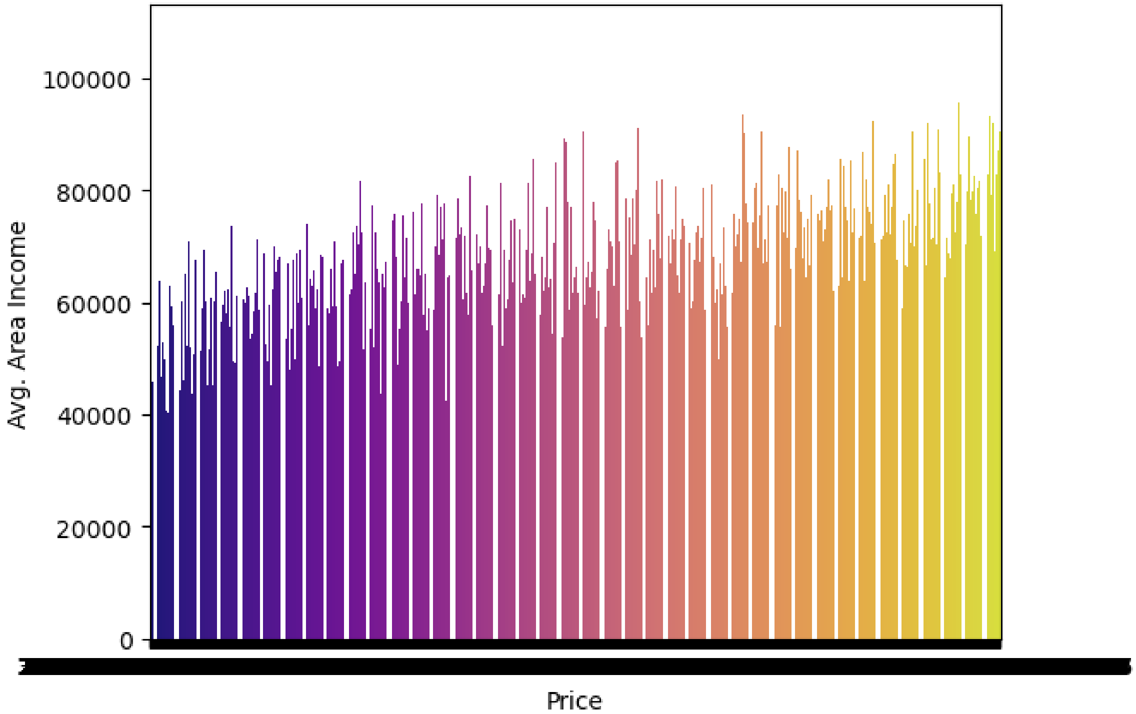



In [10]:
```
sns.histplot(dataset, y='Avg. Area Income', bins=50, color='b')
```

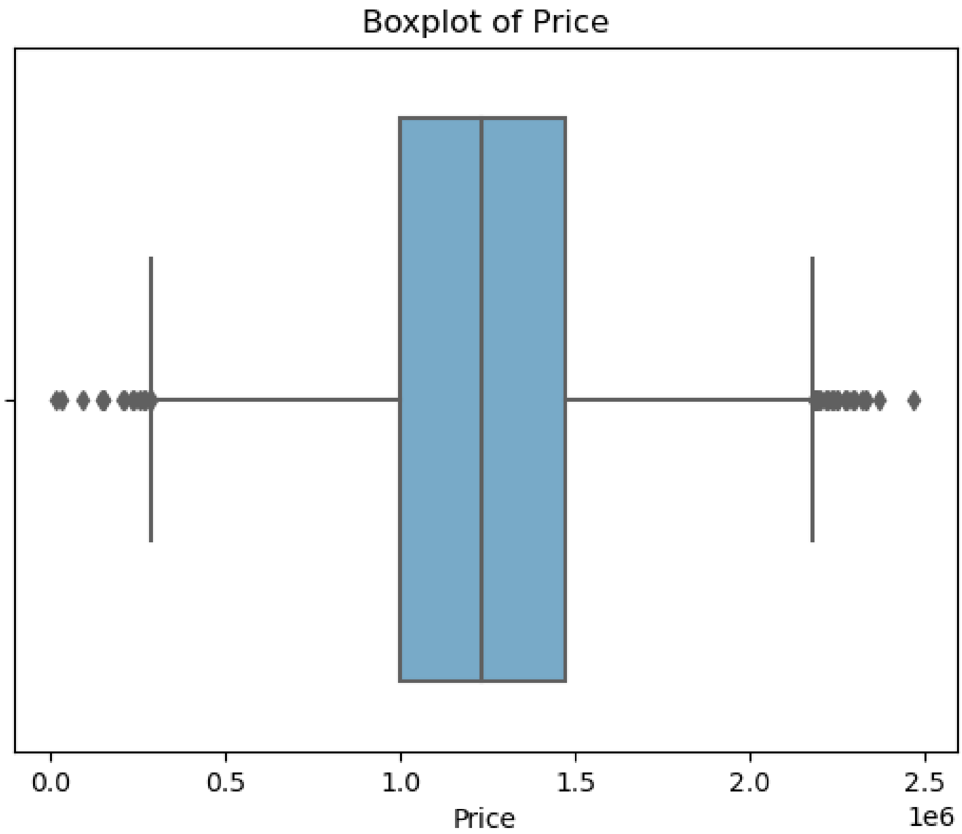
Out[10]: <Axes: xlabel='Count', ylabel='Avg. Area Income'>



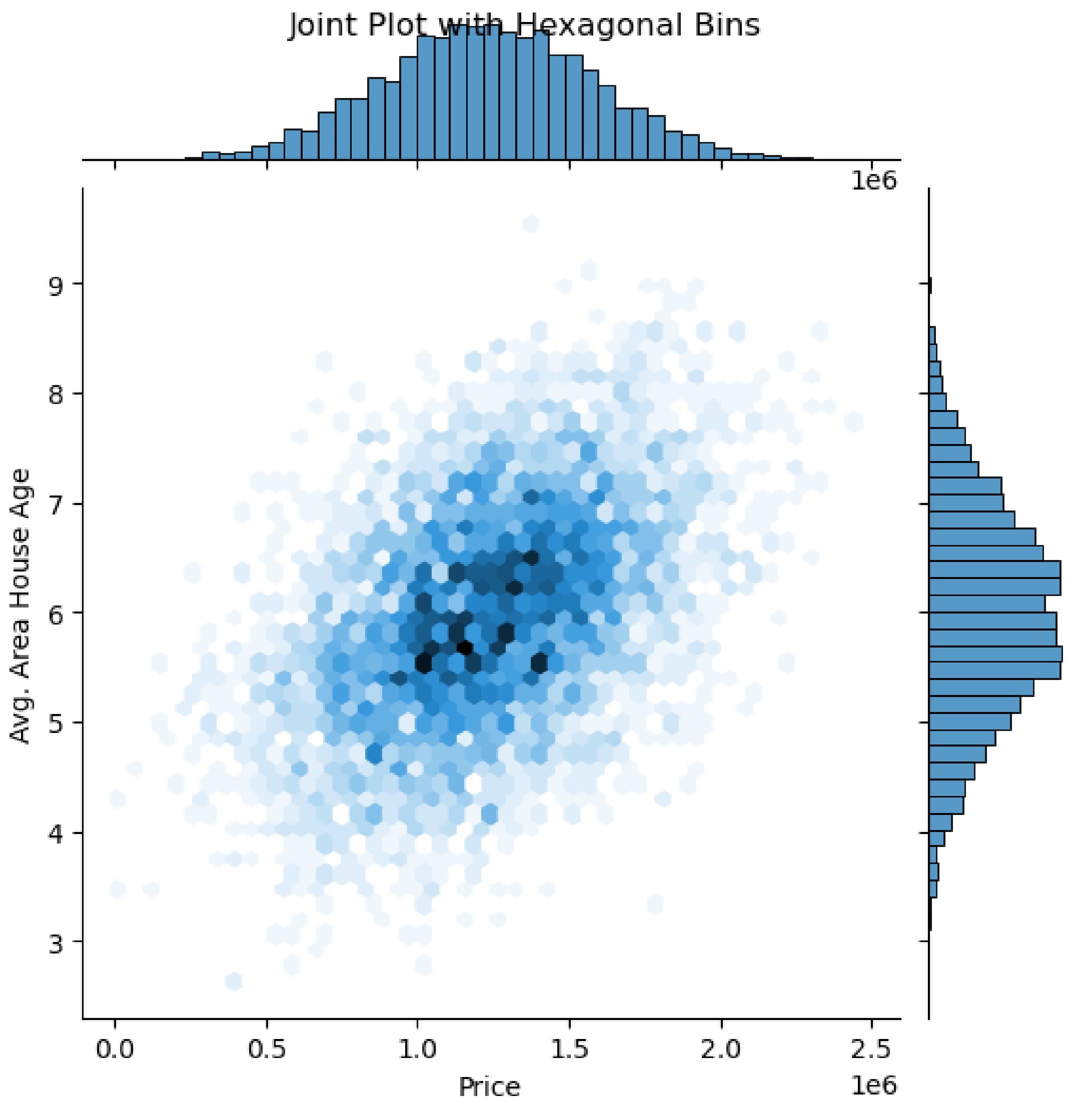
In [12]:
```
import pandas as pd
import seaborn as sns
dataset = pd.read_csv("USA_Housing.csv")
sns.barplot(x='Price', y='Avg. Area Income', data=dataset, palette='plasma')
plt.show()
```



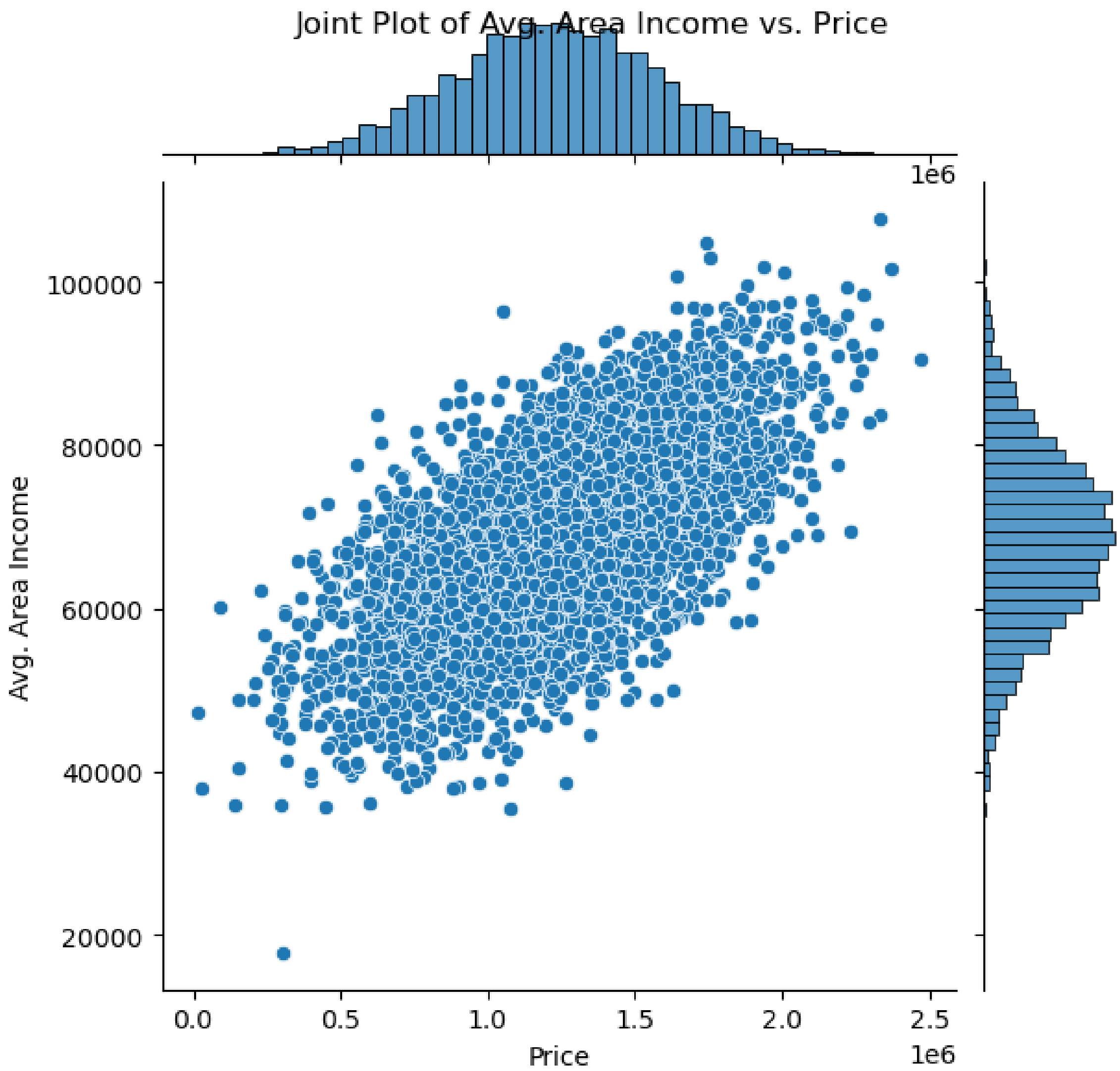
In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(data=dataset, x='Price', palette='Blues')
plt.xlabel('Price')
plt.title('Boxplot of Price')

plt.show()
```



In [13]:
```
import pandas as pd
import seaborn as sns
dataset = pd.read_csv("USA_Housing.csv")
import matplotlib.pyplot as plt

sns.jointplot(data=dataset, x='Price', y='Avg. Area House Age', kind='hex')
plt.xlabel('Price')
plt.ylabel('Avg. Area House Age')
plt.suptitle('Joint Plot with Hexagonal Bins')

plt.show()
```

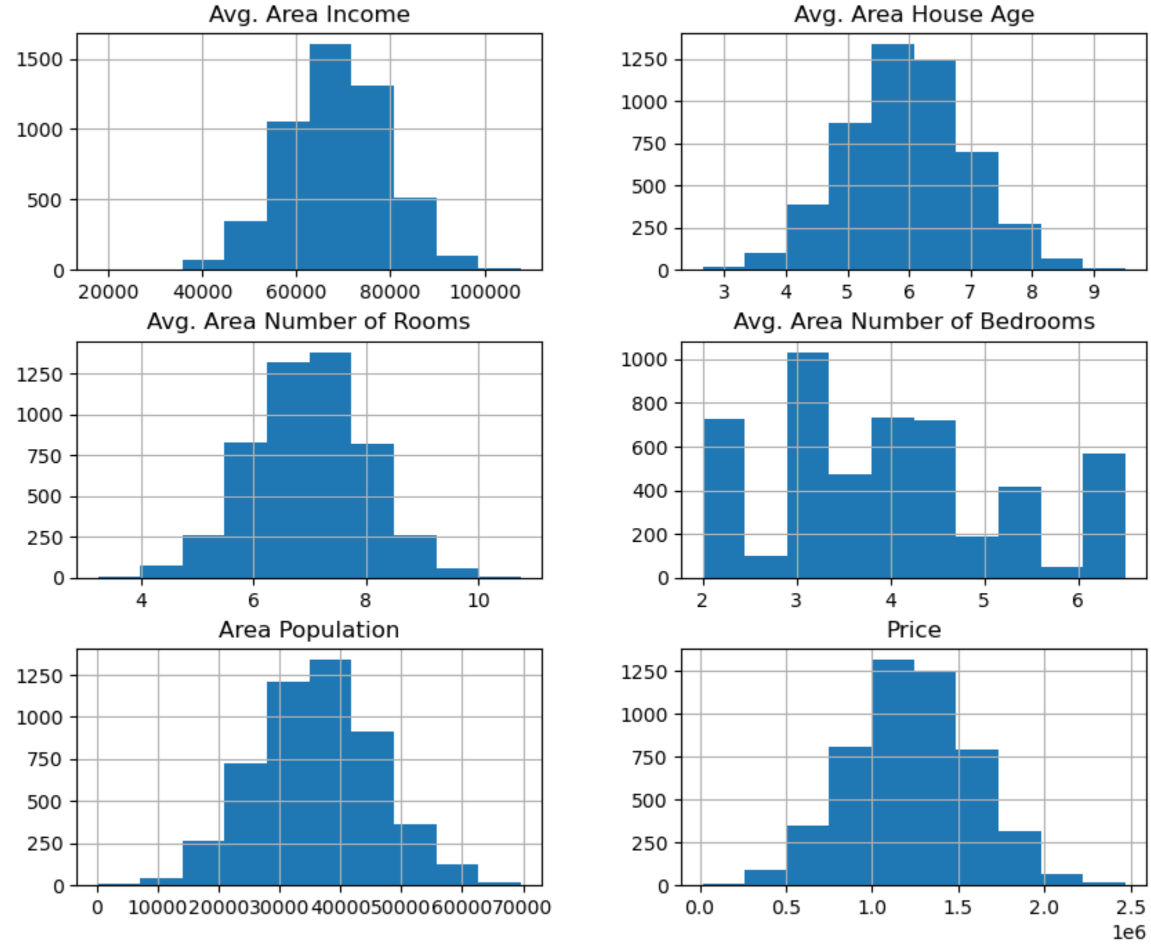


In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.jointplot(data=dataset, x='Price', y='Avg. Area Income')
plt.xlabel('Price')
plt.ylabel('Avg. Area Income')
plt.suptitle('Joint Plot of Avg. Area Income vs. Price')
plt.show()
```



In [15]:
```
dataset.hist(figsize=(10,8))
```

Out[15]:
```
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

In [130]:
```
dataset = dataset.select_dtypes(include=[np.number])
correlation_matrix = dataset.corr()
```

In [17]:
```
dataset.corr(numeric_only=True)
```

Out[17]:

| | 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 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 |

In [18]:
```
import pandas as pd
dataset = pd.read_csv("USA_Housing.csv")
# Filter the DataFrame to include only numeric columns
numeric_columns = dataset.select_dtypes(include=['number'])
correlation_matrix = numeric_columns.corr()
print(correlation_matrix)
```

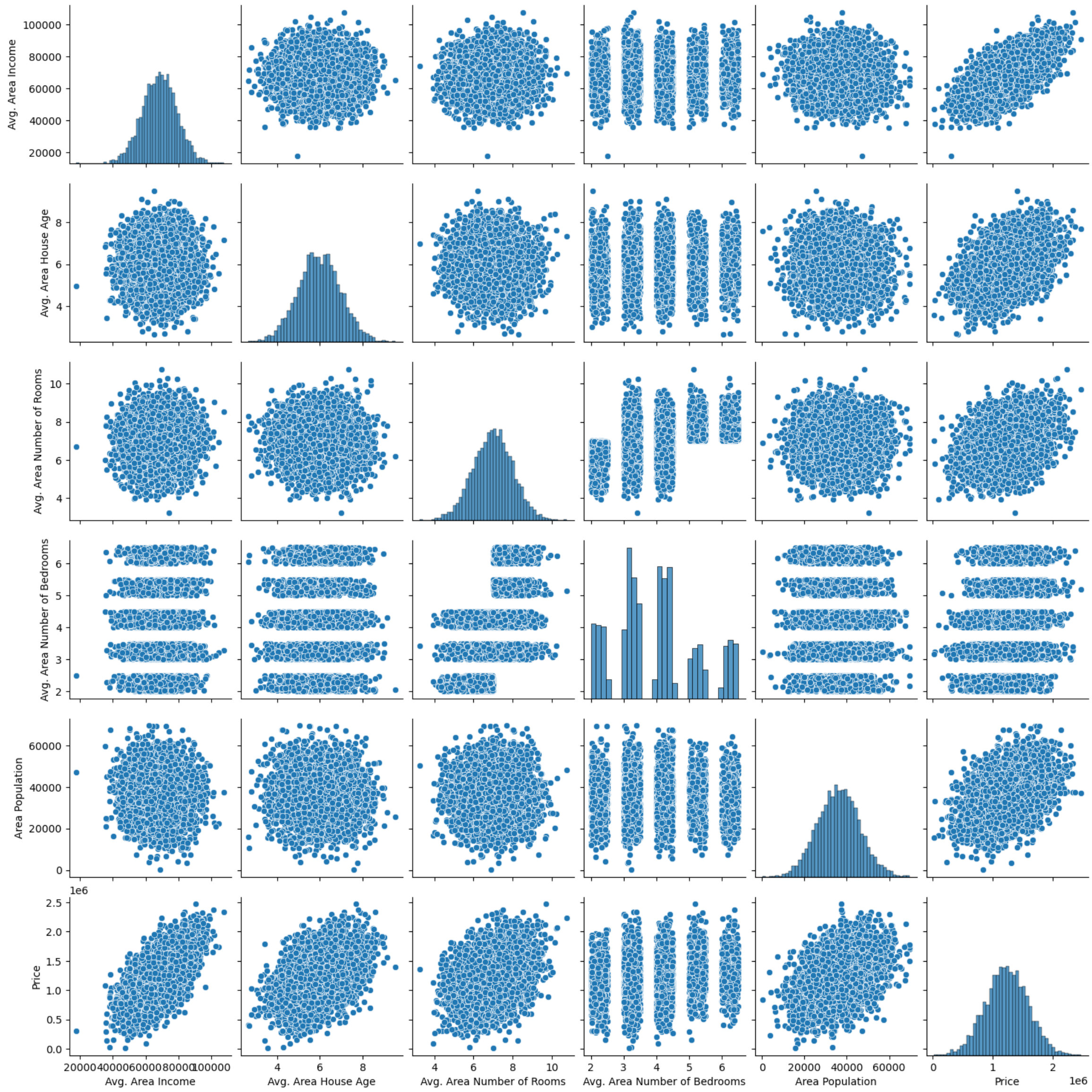
```
                              Avg. Area Income  Avg. Area House Age  \
Avg. Area Income                      1.000000            -0.002007   
Avg. Area House Age                  -0.002007             1.000000   
Avg. Area Number of Rooms            -0.011032            -0.009428   
Avg. Area Number 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  
```

In [19]:
```
plt.figure(figsize=(10,5))
sns.heatmap(dataset.corr(numeric_only = True), annot=True)
```

Out[19]: <Axes: >

In [69]:
```
plt.figure(figsize=(12,8))
sns.pairplot(dataset)
```

Out[69]:
```
<seaborn.axisgrid.PairGrid at 0x287aac16d90>

<Figure size 1200x800 with 0 Axes>
```




In [70]:
```
dataset.isnull().sum()
```

Out[70]:
```
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
```

In [145]:
```
## importing modules

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

In [149]:
```
## Dividing Data
```

In [23]:
```
X = dataset['Price']
Y = dataset[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
       'Avg. Area Number of Bedrooms', 'Area Population']]
```

In [133]:
```
# TRAINING PHASE
```

In [25]:
```
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_
```

In [26]:
```
Y_train.head()
```

Out[26]:

| | Avg. Area Income | Avg. Area House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area Population |
|---|---|---|---|---|---|
| **3413** | 69048.788093 | 6.619712 | 6.123813 | 4.33 | 36817.368760 |
| **1610** | 67866.899929 | 5.393978 | 9.359022 | 5.44 | 43122.574176 |
| **3459** | 56636.238191 | 5.497667 | 7.121872 | 6.10 | 47541.431763 |
| **4293** | 79310.361977 | 4.247434 | 7.518204 | 4.38 | 43982.188957 |
| **1039** | 72821.247664 | 6.480819 | 7.116655 | 5.33 | 40594.059297 |

In [27]:
```
Y_train.shape
```

Out[27]: (4000, 5)

In [132]:
```
# TESTING PHASE
```

In [28]:
```
Y_test.head()
```

Out[28]:

| | Avg. Area Income | Avg. Area House Age | Avg. Area Number of Rooms | Avg. Area Number of Bedrooms | Area Population |
|---|---|---|---|---|---|
| **1718** | 66774.995817 | 5.717143 | 7.795215 | 4.32 | 36788.980327 |
| **2511** | 62184.539375 | 4.925758 | 7.427689 | 6.22 | 26008.309124 |
| **345** | 73643.057298 | 6.766853 | 8.337085 | 3.34 | 43152.139577 |
| **2521** | 61909.041438 | 6.228343 | 6.593138 | 4.29 | 28953.925377 |
| **54** | 72942.705059 | 4.786222 | 7.319886 | 6.41 | 24377.909049 |

In [29]:
```
Y_test.shape
```

Out[29]: (1000, 5)

In [30]:
```
# Standardizing the data
```

In [31]:
```
from sklearn.preprocessing import StandardScaler
import numpy as np
sc = StandardScaler()
X_train_reshaped = X_train.to_numpy().reshape(-1, 1)
X_test_reshaped = X_test.to_numpy().reshape(-1, 1)
X_train_scal = sc.fit_transform(X_train_reshaped)
X_test_scal = sc.transform(X_test_reshaped)
```

In [32]:
```
# Model Building and Evaluation
## Model 1 - Linear Regression
```

In [33]:
```
from sklearn.linear_model import LinearRegression
model_lr = LinearRegression()
```

In [34]:
```
model_lr.fit(X_train_scal, Y_train)
```

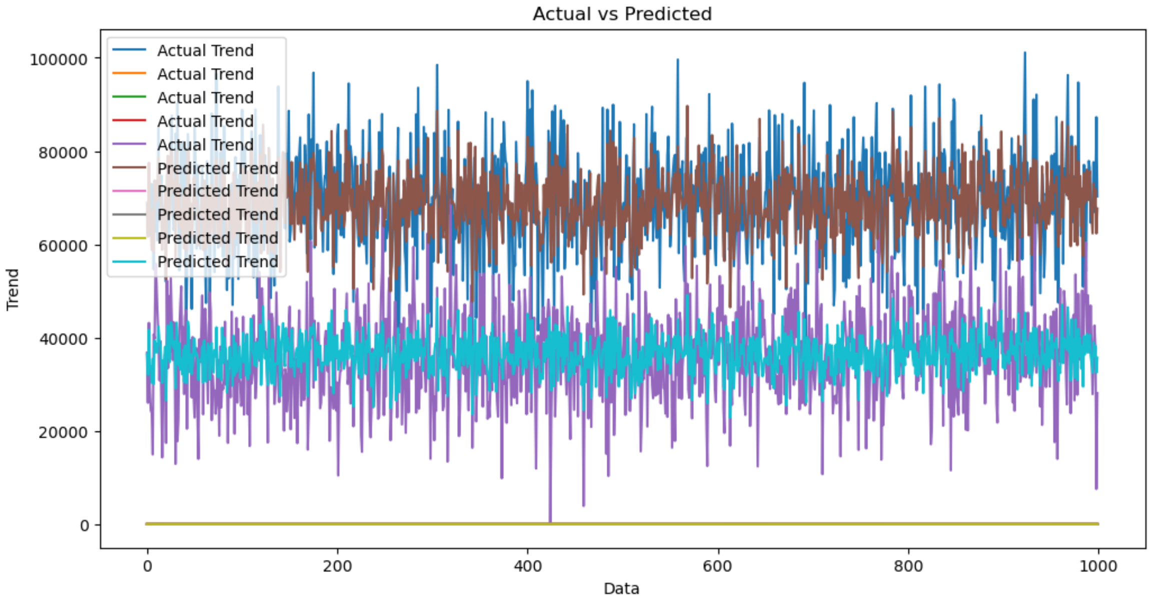
Out[34]:
```
▾ LinearRegression
LinearRegression()
```

In [35]:
```
## Predicting Prices
```

In [36]:
```
Prediction1 = model_lr.predict(X_test_scal)
```

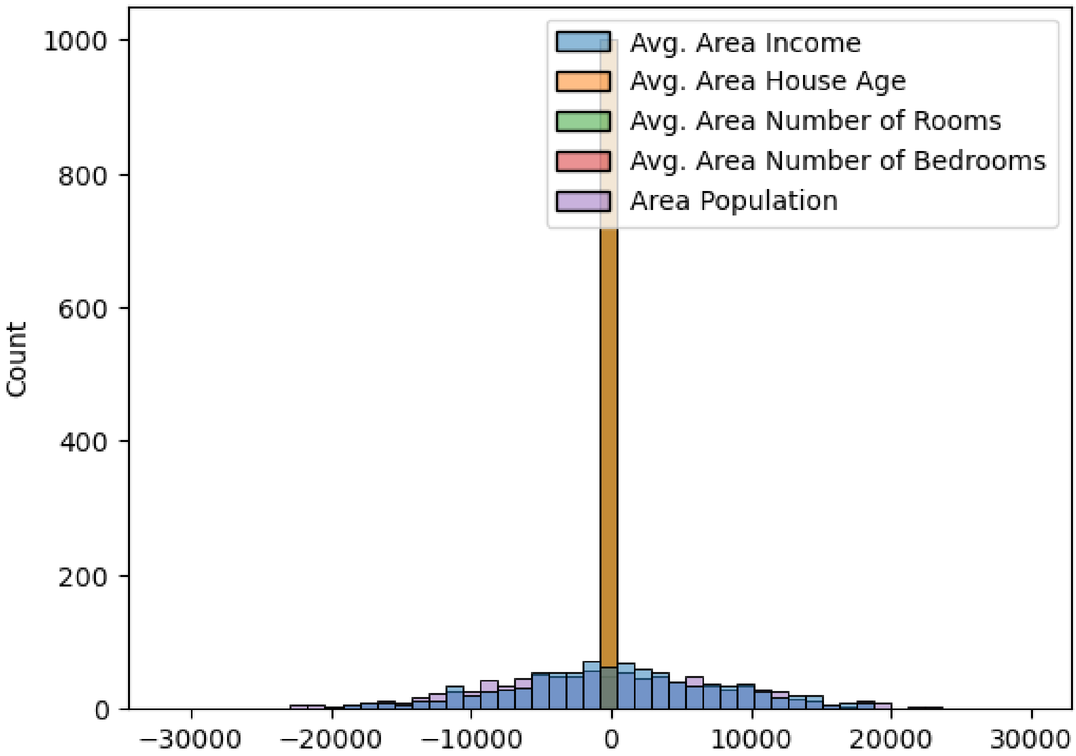
In [37]:
```
## Evaluation of Predicted Data
```

In [38]:
```
import matplotlib.pyplot as plt
import numpy as np
if len(Y_test) == len(Prediction1):
    plt.figure(figsize=(12, 6))
    plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
    plt.plot(np.arange(len(Prediction1)), Prediction1, label='Predicted Trend')
    plt.xlabel('Data')
    plt.ylabel('Trend')
    plt.legend()
    plt.title('Actual vs Predicted')
else:
    print("Lengths of Y_test and Prediction1 do not match.")
```




In [39]:
```
sns.histplot((Y_test-Prediction1), bins=50,color='r')
```

Out[39]: <Axes: ylabel='Count'>



In [40]:
```
print(r2_score(Y_test, Prediction1))
print(mean_absolute_error(Y_test, Prediction1))
print(mean_squared_error(Y_test, Prediction1))
```

```
0.1866240420321335
2752.0362321666653
30247095.384854764
```

In [76]:
```
import numpy as np

# Assuming X_train is a Pandas Series
X_train_reshaped = X_train.values.reshape(-1, 1)

# Then, proceed with fitting the model using the reshaped data
model_lr.fit(X_train_reshaped, Y_train)
```

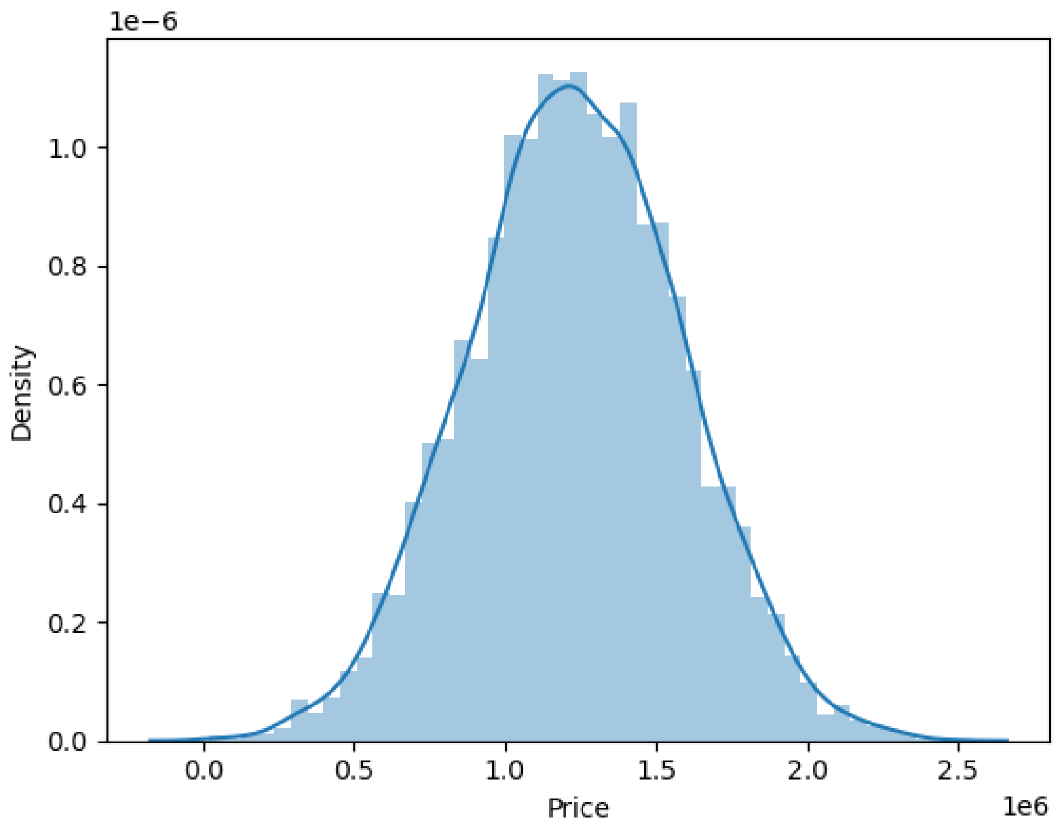
Out[76]:
```
▾ LinearRegression
LinearRegression()
```

In [78]:
```
print(model_lr.intercept_)
```

```
[4.47830789e+04 4.44480736e+00 5.81749836e+00 3.23285804e+00
 2.17417850e+04]
```

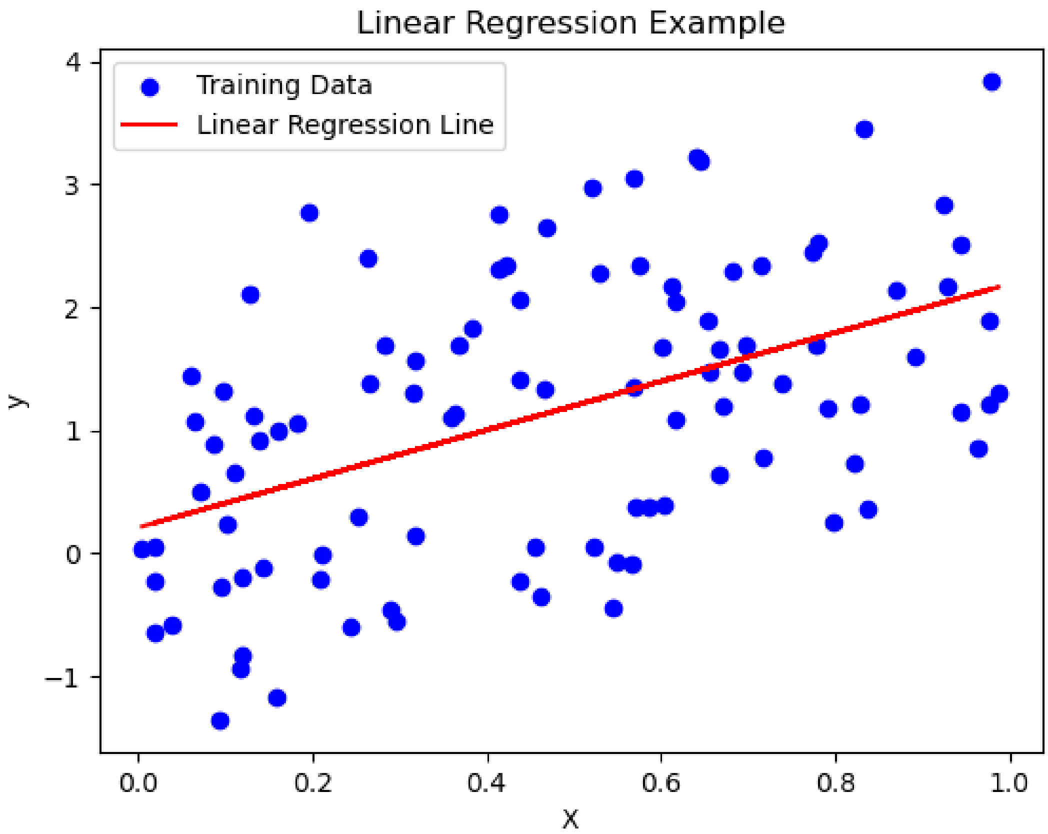
In [79]:
```
sns.distplot(dataset["Price"])
```

Out[79]: <Axes: xlabel='Price', ylabel='Density'>



In [142]:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
np.random.seed(0)
X = np.random.rand(100, 1)
y = 2 * X.squeeze() + np.random.randn(100)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Creating and fitting the Linear Regression model
lm = LinearRegression()
lm.fit(X_train, y_train)

# Making predictions on the test set
predictions = lm.predict(X_test)

# Plotting the training data and the linear regression line
plt.scatter(X, y, color='blue', label='Training Data')
plt.plot(X, lm.predict(X), color='red', label='Linear Regression Line')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Linear Regression Example')
plt.legend()
plt.show()
```



In [144]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

np.random.seed(0)
X = np.random.rand(100, 1)  # Feature
y = 2 * X.squeeze() + np.random.randn(100)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Creating and fitting the Linear Regression model
lm = LinearRegression()
lm.fit(X_train, y_train)

# Creating a DataFrame with coefficients
coeff_df = pd.DataFrame(lm.coef_, columns=['Coefficient'])
print(coeff_df)
```

```
   Coefficient
0     1.980518
```

In [86]:
```
## Model 2 - Support Vector Regressor
```

In [42]:
```
model_svr = SVR()
```

In [43]:
```
import numpy as np
Y_train_array = Y_train.to_numpy()
```

In [44]:
```
from sklearn.multioutput import MultiOutputRegressor
from sklearn.svm import SVR
base_model = SVR(kernel='linear', C=1.0)

# Wrap it with MultiOutputRegressor
model_svr = MultiOutputRegressor(base_model)
model_svr.fit(X_train_scal, Y_train)
```

Out[44]:

**MultiOutputRegressor**

**estimator: SVR**

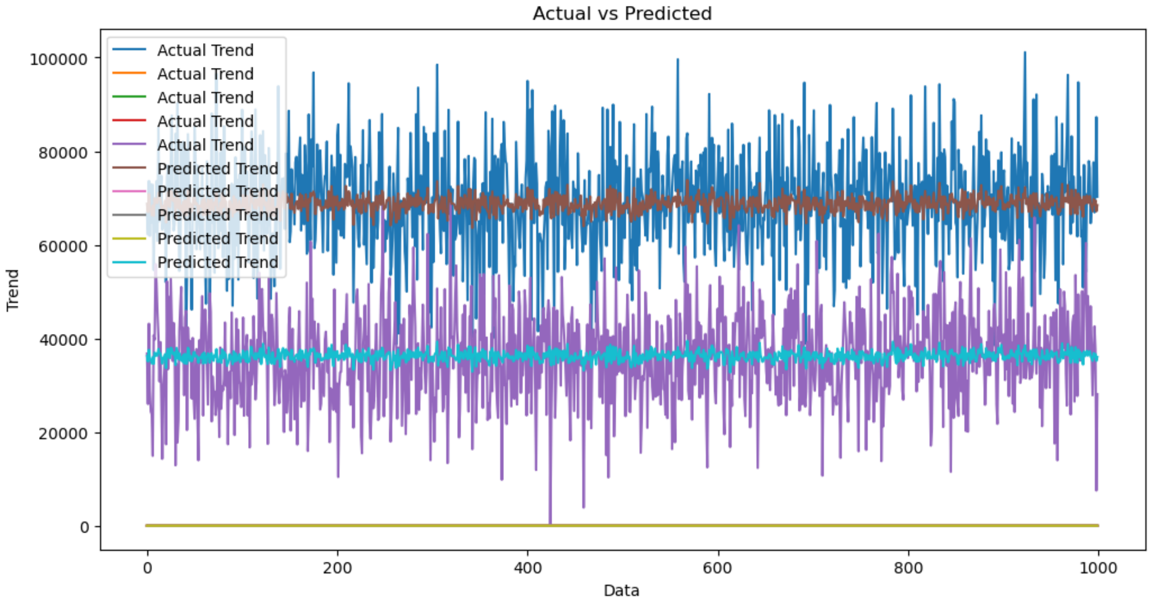
SVR

In [45]:
```
## Predicting Prices
```

In [46]:
```
Prediction2 = model_svr.predict(X_test_scal)
```

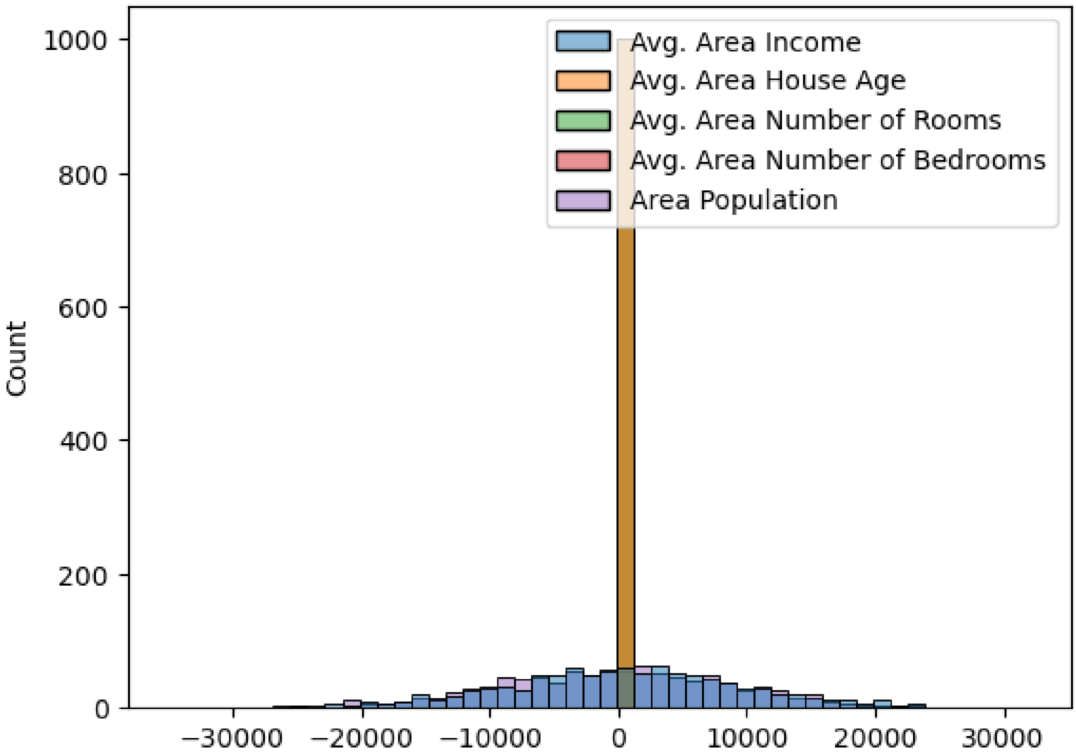
In [47]:
```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction2, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[47]:
```
Text(0.5, 1.0, 'Actual vs Predicted')
```




In [48]:
```
sns.histplot((Y_test-Prediction2), bins=50)
```

Out[48]:
```
<Axes: ylabel='Count'>
```



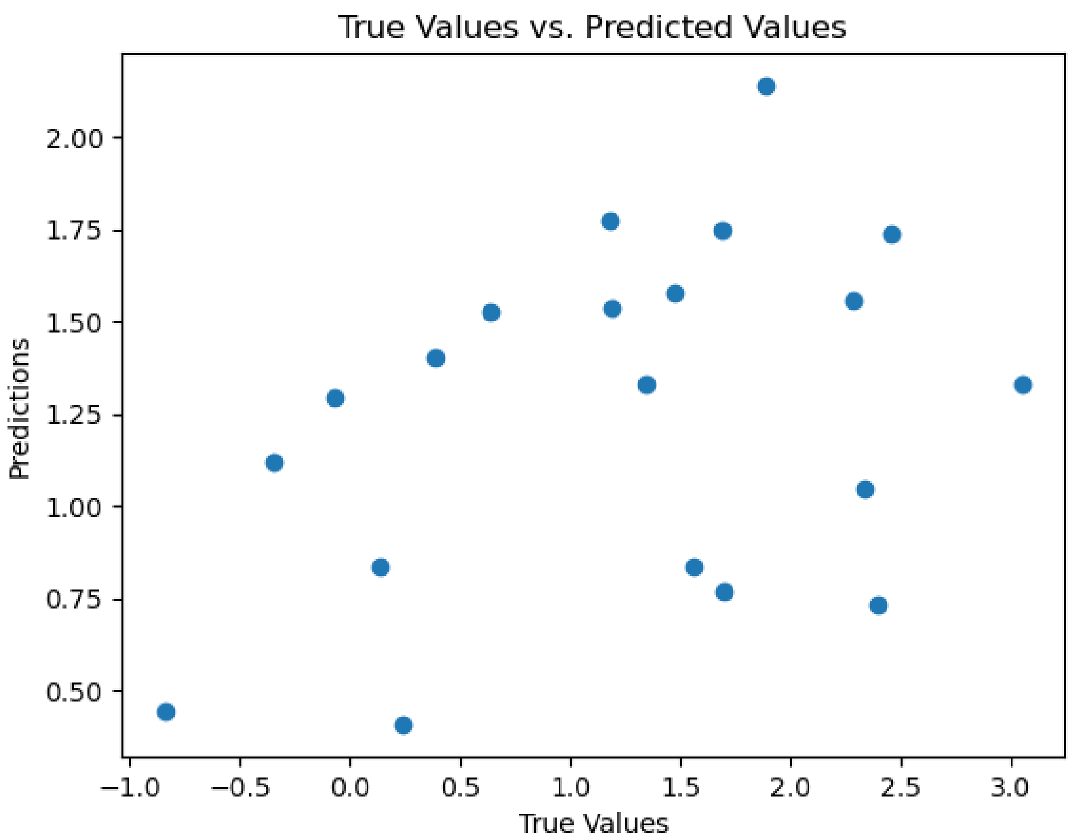
In [49]:
```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
0.12092808540883056
3048.284745891538
37172622.89729237
```

In [157]:
```

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Creating and fitting the Linear Regression model
lm = LinearRegression()
lm.fit(X_train, y_train)

# Creating a DataFrame with coefficients
coeff_df = pd.DataFrame(lm.coef_, columns=['Coefficient'])
print(coeff_df)
```

```
   Coefficient
0     1.980518
```

In [156]:
```
import matplotlib.pyplot as plt

# Scatter plot of true test values vs. predicted values
plt.scatter(y_test, predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('True Values vs. Predicted Values')
plt.show()
```



In [50]:
```
## Model 3 - Lasso Regression
```

In [51]:
```
model_lar = Lasso(alpha=1)
```

In [52]:
```
model_lar.fit(X_train_scal,Y_train)
```

Out[52]:
```
▾ Lasso
Lasso(alpha=1)
```

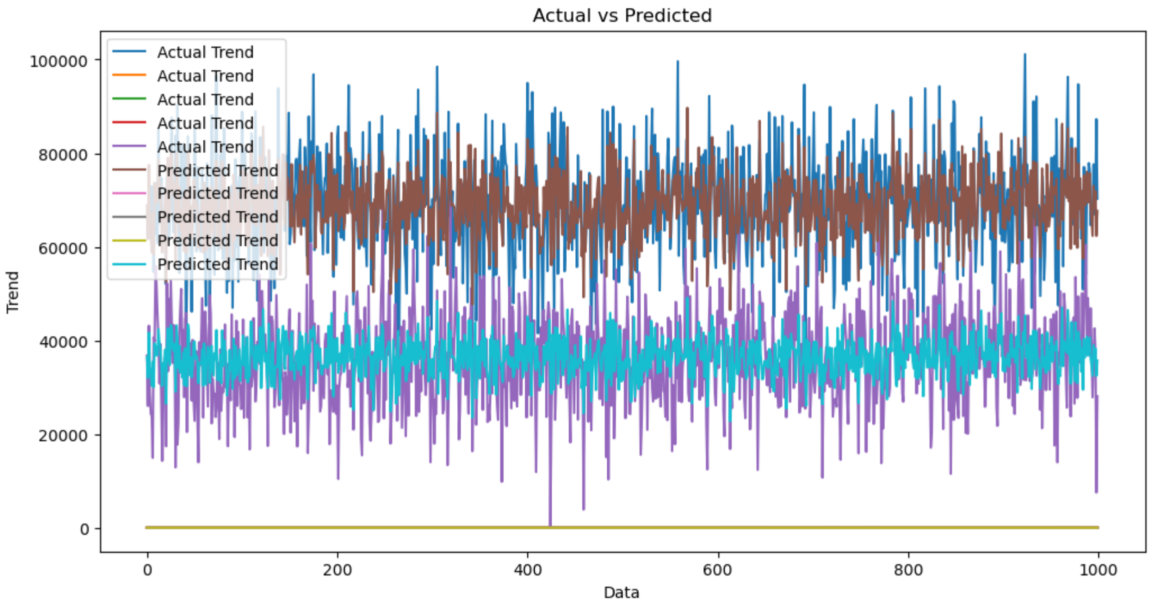
In [53]:
```
## Predicting Prices
```

In [54]:
```
Prediction3 = model_lar.predict(X_test_scal)
```

In [55]:
```
## Evaluation of Predicted Data
```

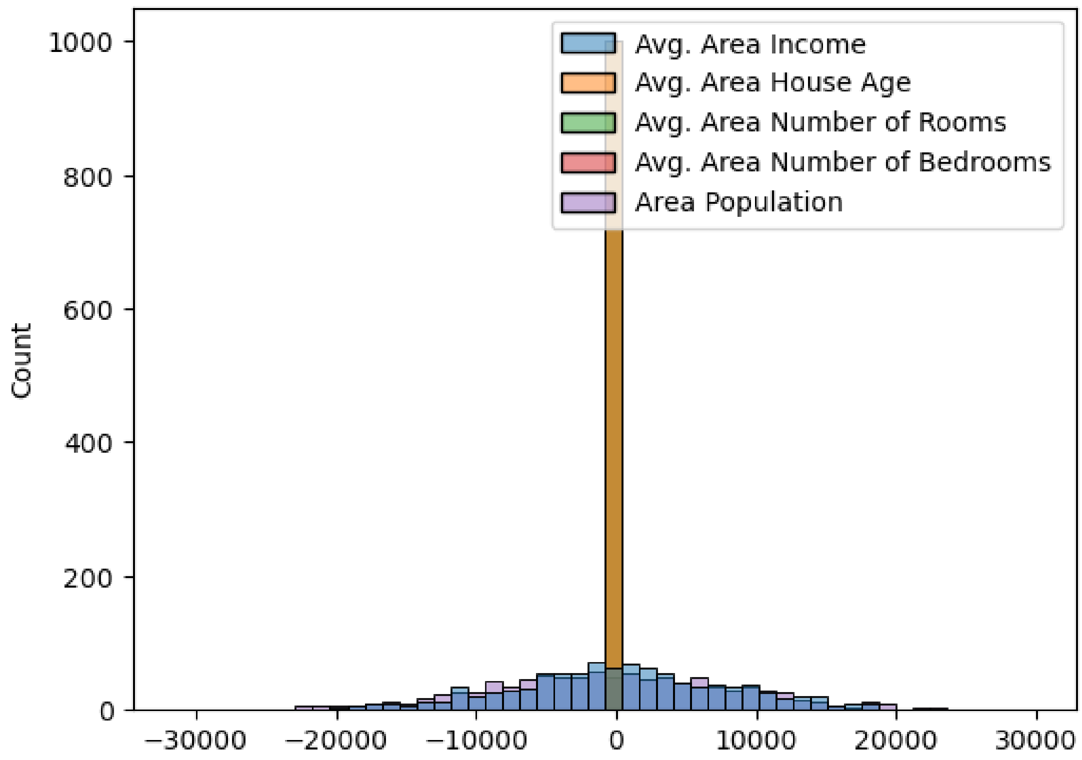
In [56]:
```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction1, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[56]: Text(0.5, 1.0, 'Actual vs Predicted')



In [57]:
```
sns.histplot((Y_test-Prediction3), bins=50)
```

Out[57]: <Axes: ylabel='Count'>




In [58]:
```
print(r2_score(Y_test, Prediction2))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
0.12092808540883056
3048.284745891538
37172622.89729237
```

In [59]:
```
## Model 4 - Random Forest Regressor
```

In [60]:
```
model_rf = RandomForestRegressor(n_estimators=50)
```

In [61]:
```
model_rf.fit(X_train_scal, Y_train)
```

Out[61]:
```
RandomForestRegressor
RandomForestRegressor(n_estimators=50)
```

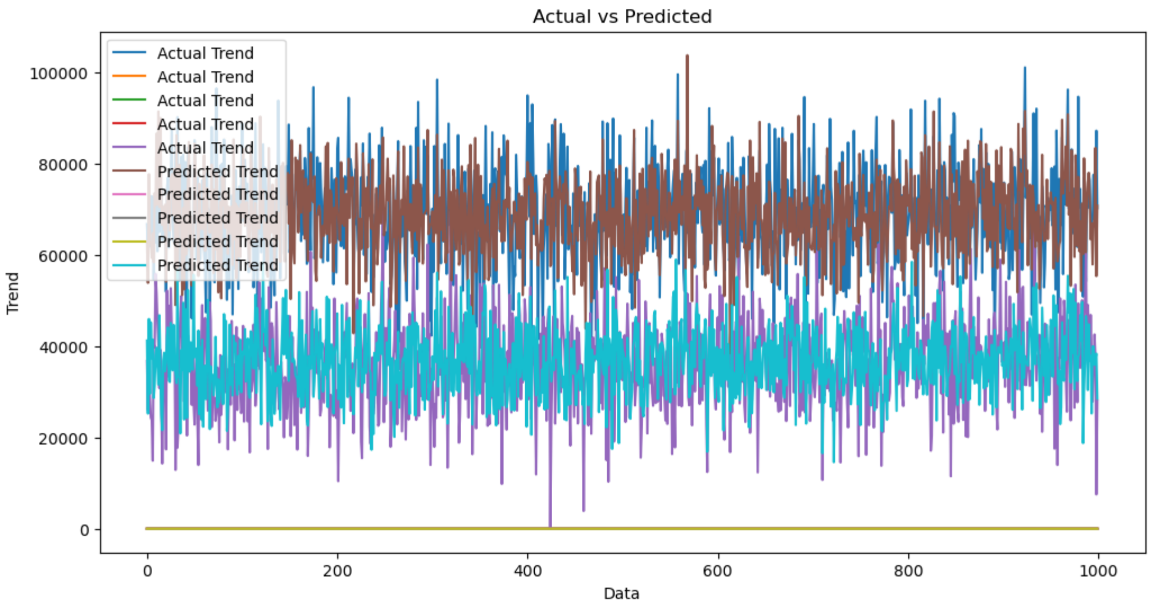
In [62]:
```
## Predicting Prices
```

In [63]:
```
Prediction4 = model_rf.predict(X_test_scal)
```

In [64]:
```
## Evaluation of Predicted Data
```

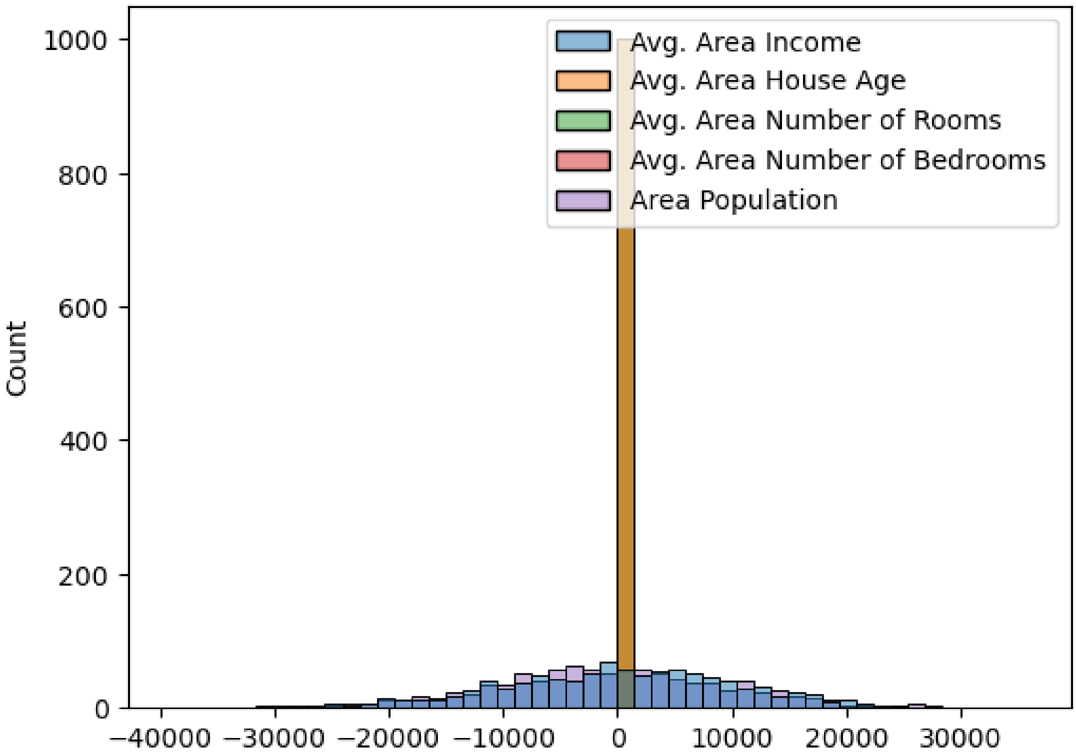
In [65]:
```
plt.figure(figsize=(12,6))
plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
plt.plot(np.arange(len(Y_test)), Prediction4, label='Predicted Trend')
plt.xlabel('Data')
plt.ylabel('Trend')
plt.legend()
plt.title('Actual vs Predicted')
```

Out[65]: Text(0.5, 1.0, 'Actual vs Predicted')




In [66]:
```
sns.histplot((Y_test-Prediction4), bins=50)
```

Out[66]: <Axes: ylabel='Count'>



In [67]:
```
print(r2_score(Y_test, Prediction3))
print(mean_absolute_error(Y_test, Prediction2))
print(mean_squared_error(Y_test, Prediction2))
```

```
0.11034396748383535
3048.284745891538
37172622.89729237
```

In [ ]:

In [167]:
```
income_value = dataset.at[0, 'Avg. Area Income']
population_value = dataset.at[2, 'Area Population']
price_value = dataset.at[3, 'Price']
rooms_value = dataset.at[1, 'Avg. Area Number of Rooms']
Bedroom_value = dataset.at[4,'Avg. Area Number of Bedrooms']
```

In [164]:
```
income_value
```

Out[164]: 79545.45857431678

In [165]:
```
population_value
```

Out[165]: 36882.15939970458

In [96]:
```
price_value
```

Out[96]: 1260616.8066294468

In [166]:
```
rooms_value
```

Out[166]: 6.730821019094919

In [168]:
```
Bedroom_value
```

Out[168]: 4.23

In [99]:
```
import numpy as np
import pandas as pd  # Use 'pandas' instead of 'panda'

# Load your dataset into a Pandas DataFrame
dataset = pd.read_csv("USA_Housing.csv")

# Calculate the average
average_price = dataset['Price'].mean()
print("Average Price:", average_price)
```

```
Average Price: 1232072.654142357
```

In [100]:
```
from sklearn.metrics import accuracy_score
```

In [102]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

# Load your dataset into a Pandas DataFrame
dataset = pd.read_csv('USA_Housing.csv')

# Assuming 'Price' is your target column
# Split the dataset into features (X) and the target variable (y)
X = dataset.drop(columns=['Price', 'Address'])
y = dataset['Price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_

# Create and train a machine learning model (RandomForestRegressor) using X_trai
model = RandomForestRegressor()
model.fit(X_train, y_train)

# Use the trained model to make predictions on the test data (X_test)
y_pred = model.predict(X_test)

# Evaluate the regression model using mean squared error and R-squared (R2) scor
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print or use the evaluation metrics
print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```

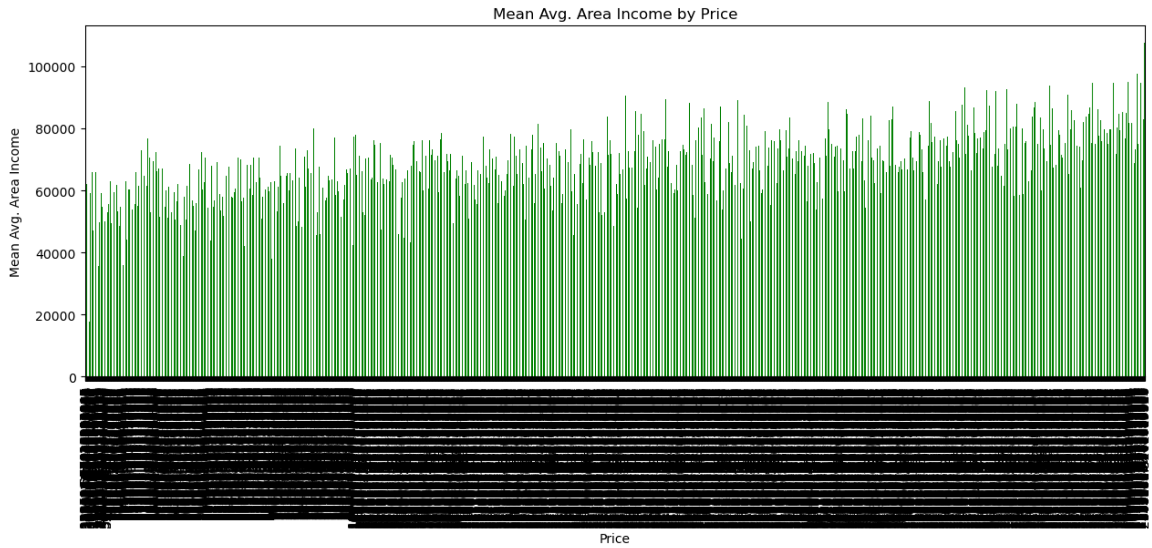
```
Mean Squared Error: 15010250912.2045
R-squared (R2) Score: 0.8828507619666764
```

In [136]:
```
!pip install basemap
```

```
Requirement already satisfied: basemap in c:\users\vijayraj r\anaconda3\lib\sit
e-packages (1.3.8)
Requirement already satisfied: basemap-data<1.4,>=1.3.2 in c:\users\vijayraj r
\anaconda3\lib\site-packages (from basemap) (1.3.2)
Requirement already satisfied: pyshp<2.4,>=1.2 in c:\users\vijayraj r\anaconda3
\lib\site-packages (from basemap) (2.3.1)
Requirement already satisfied: matplotlib<3.8,>=1.5 in c:\users\vijayraj r\anac
onda3\lib\site-packages (from basemap) (3.7.2)
Requirement already satisfied: pyproj<3.7.0,>=1.9.3 in c:\users\vijayraj r\anac
onda3\lib\site-packages (from basemap) (3.6.1)
Requirement already satisfied: numpy<1.26,>=1.21 in c:\users\vijayraj r\anacond
a3\lib\site-packages (from basemap) (1.24.3)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\vijayraj r\anaconda
3\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\vijayraj r\anaconda3\li
b\site-packages (from matplotlib<3.8,>=1.5->basemap) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\vijayraj r\anacond
a3\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vijayraj r\anacond
a3\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\vijayraj r\anaconda3
\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\vijayraj r\anaconda3\l
ib\site-packages (from matplotlib<3.8,>=1.5->basemap) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\vijayraj r\ana
conda3\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\vijayraj r\anac
onda3\lib\site-packages (from matplotlib<3.8,>=1.5->basemap) (2.8.2)
Requirement already satisfied: certifi in c:\users\vijayraj r\anaconda3\lib\sit
e-packages (from pyproj<3.7.0,>=1.9.3->basemap) (2023.7.22)
Requirement already satisfied: six>=1.5 in c:\users\vijayraj r\anaconda3\lib\si
te-packages (from python-dateutil>=2.7->matplotlib<3.8,>=1.5->basemap) (1.16.0)
```

In [104]:
```
from mpl_toolkits.basemap import Basemap
```

In [135]:
```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 5))
bar_graph = dataset.groupby('Price')['Avg. Area Income'].mean()
bar_graph.plot(kind='bar', color='Green')
plt.xlabel('Price')
plt.ylabel('Mean Avg. Area Income')
plt.title('Mean Avg. Area Income by Price')
plt.show()
```



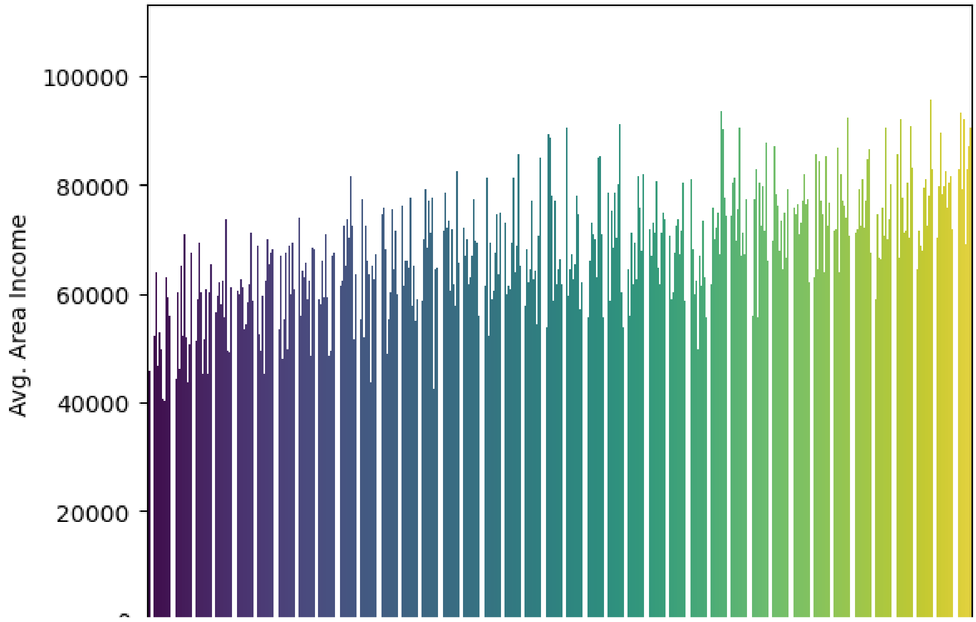

In [113]:
```
len(X_test)
```

Out[113]: 1000

In [114]:
```
len(Y_test)
```

Out[114]: 1000

In [158]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

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

# Extracting the data
X = dataset['Price']
Y = dataset['Avg. Area Income']

sns.barplot(x=X, y=Y, data=dataset, palette='viridis')
plt.show()
```




In [ ]: