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

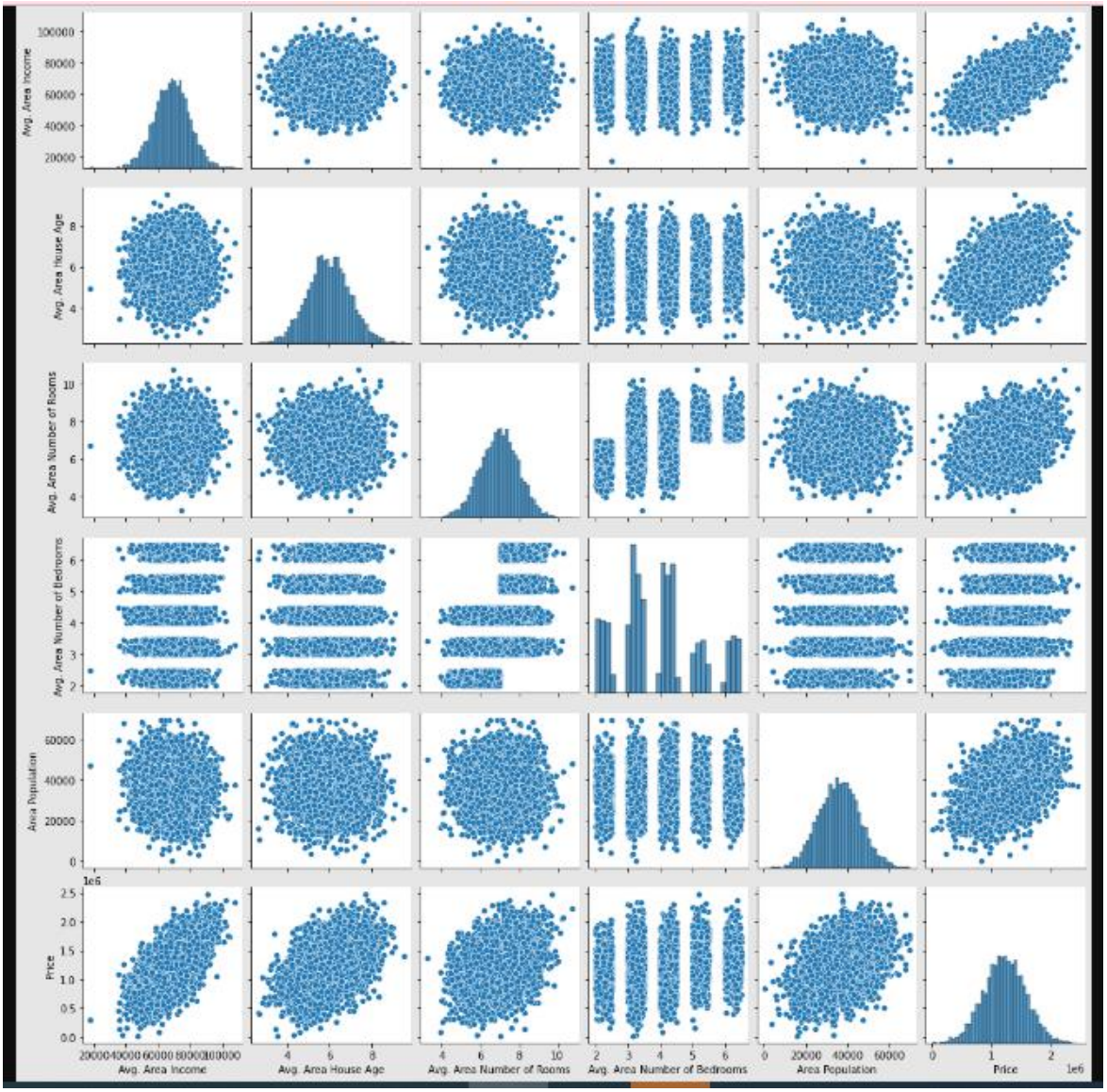
data.duplicated().sum()

0

sb.pairplot(data = data)

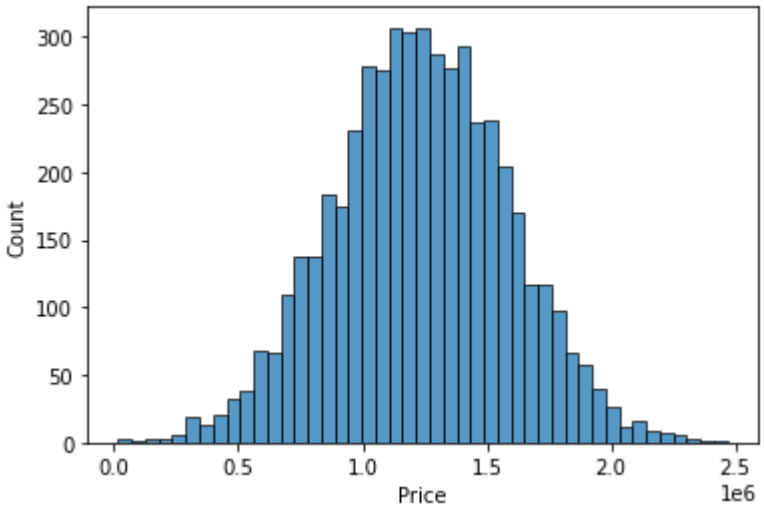
/opt/conda/lib/python3.10/site-packages/seaborn/axisgrid.py:118:
self._figure.tight_layout(*args,**kwarg)

<seaborn.axisgrid.PairGrid at 0x79d70cabb5b0>

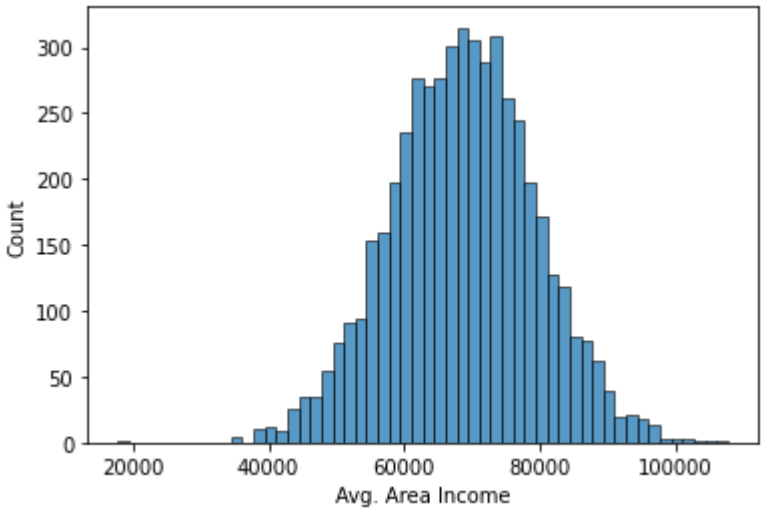


Visualisation & Pre processor:

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

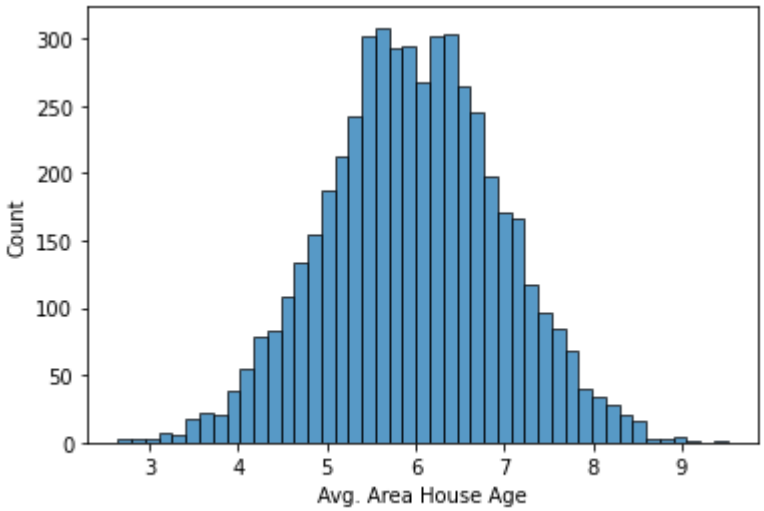


sb.histplot(x = data['Avg. Area Income']);

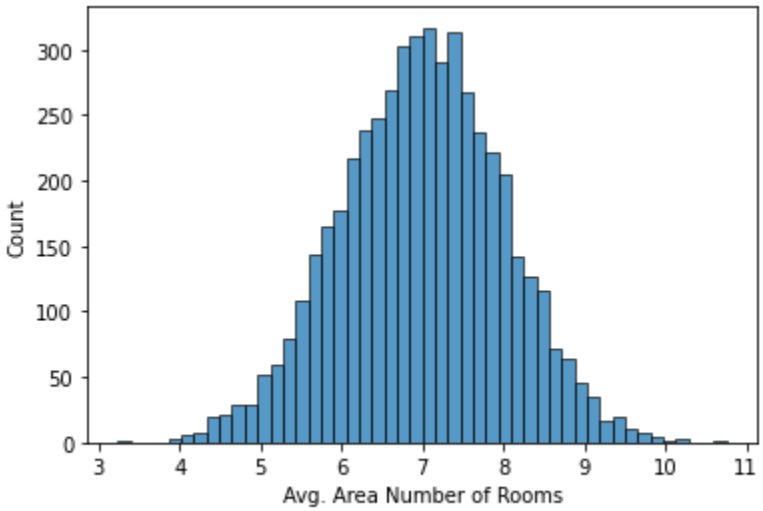


sb.histplot(x = data['Avg. Area House Age'])

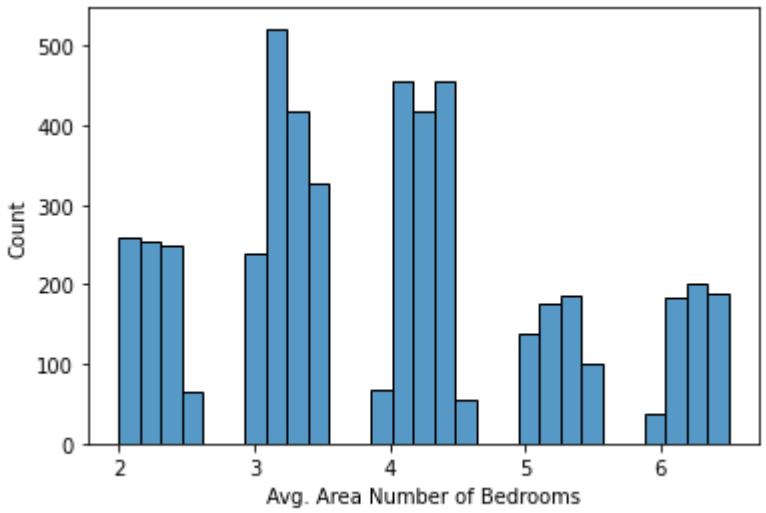
<AxesSubplot:xlabel='Avg. Area House Age', ylabel='Count'>



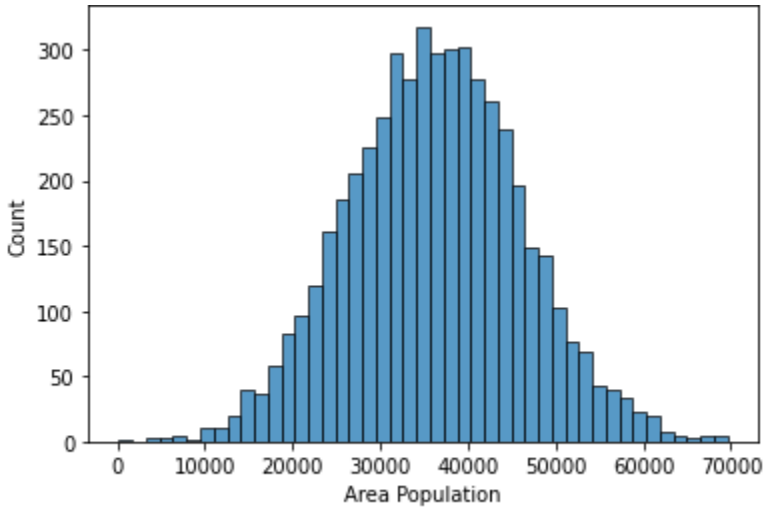
sb.histplot(x = data['Avg. Area Number of Rooms']);



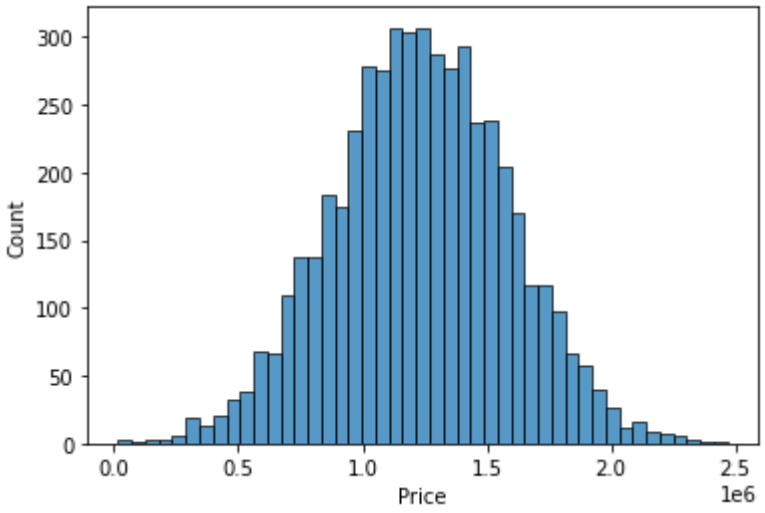
sb.histplot(x = data['Avg. Area Number of Bedrooms']);



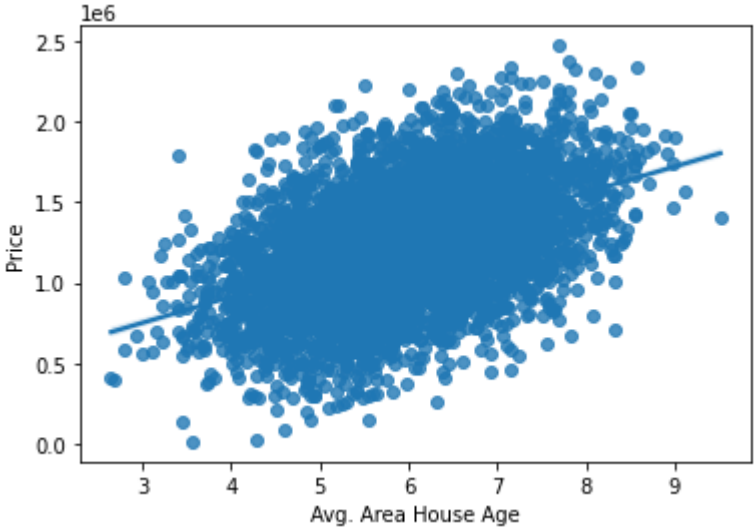
sb.histplot(x = data['Area Population']);



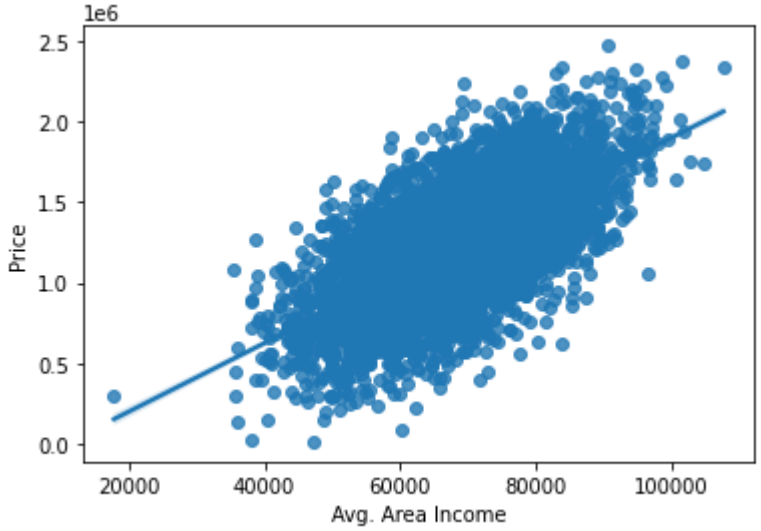
sb.histplot(x = data['Price']);



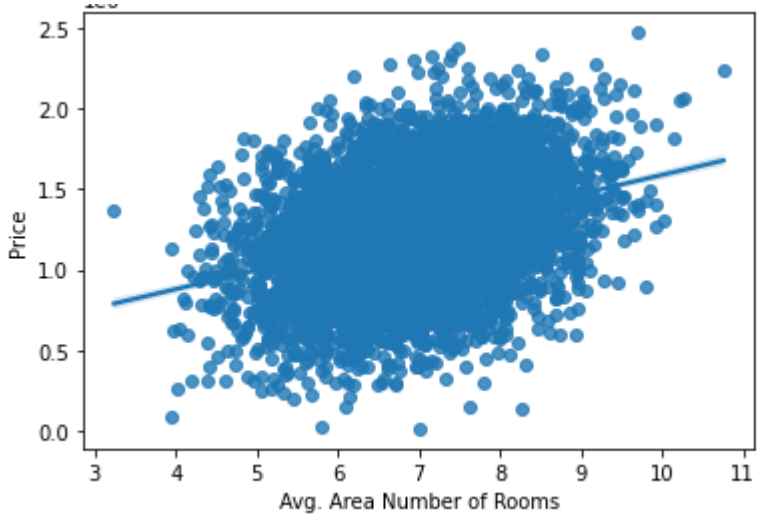
sb.regplot(x = data['Avg. Area House Age'], y = data['Price']);



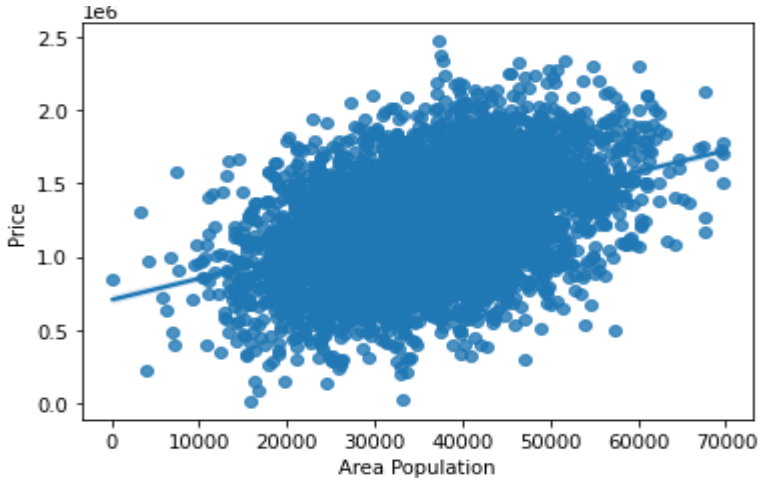
sb.regplot(x = data['Avg. Area Income'], y = data['Price']);



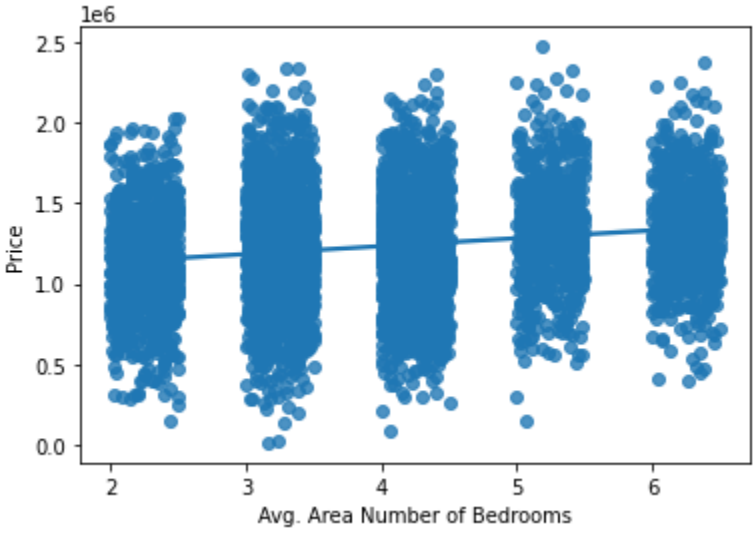
sb.regplot(x = data['Avg. Area Number of Rooms'], y = data['Price']);



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))

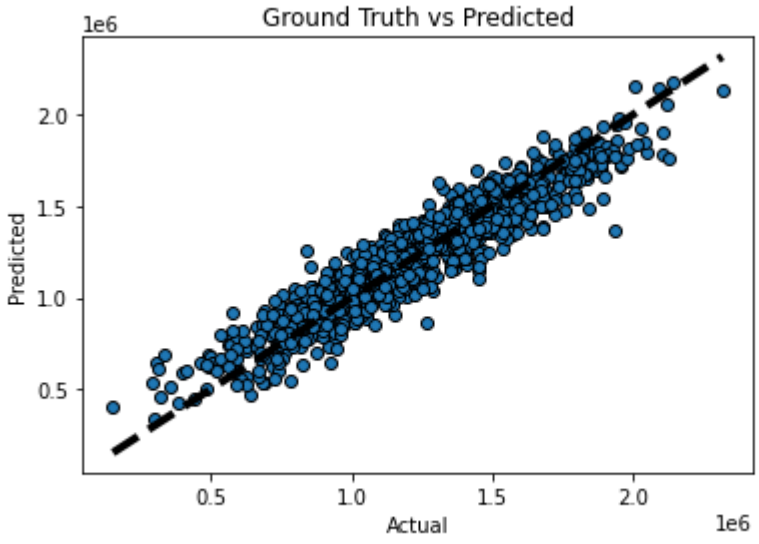
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)

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

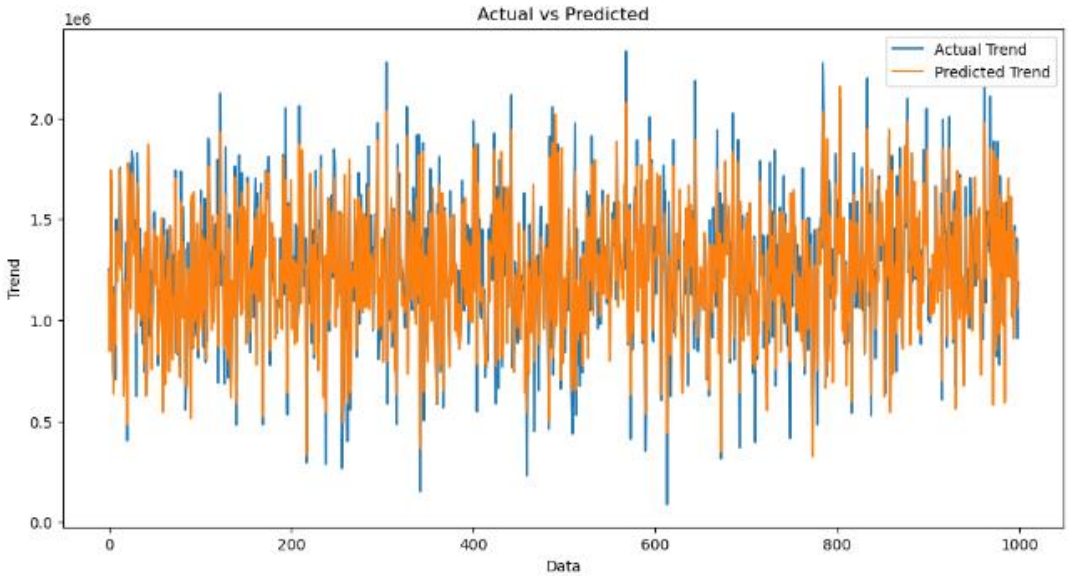
RandomForestRegressor(n_estimators=50)

Predicting Prices:

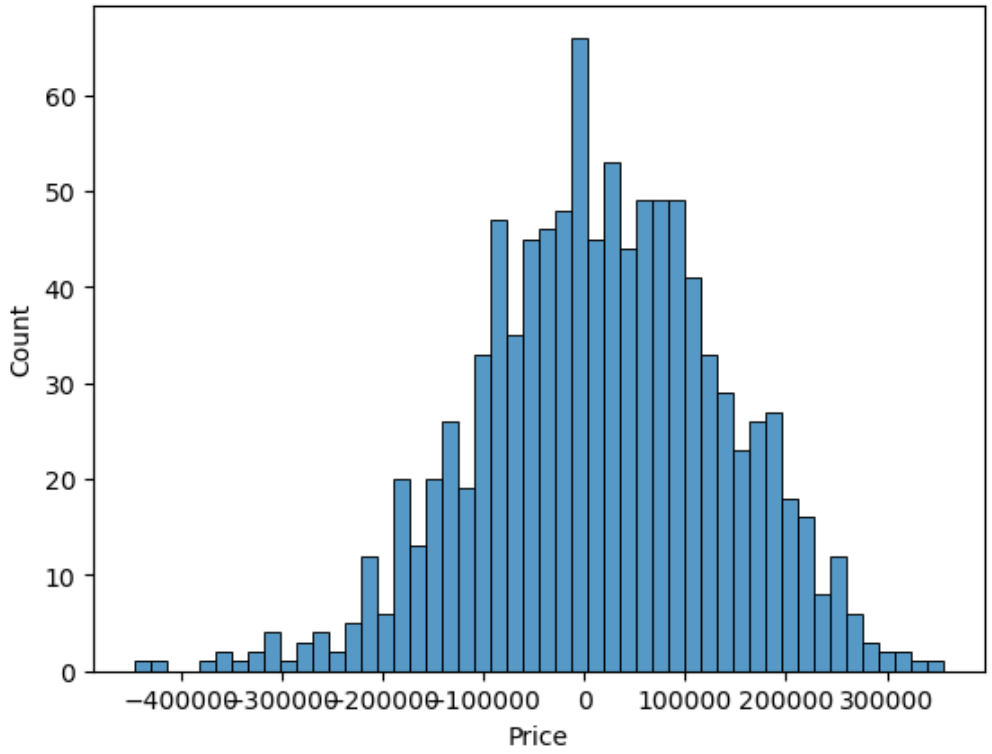
```
Prediction4 = model_rf.predict(X_test_scal)
```

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

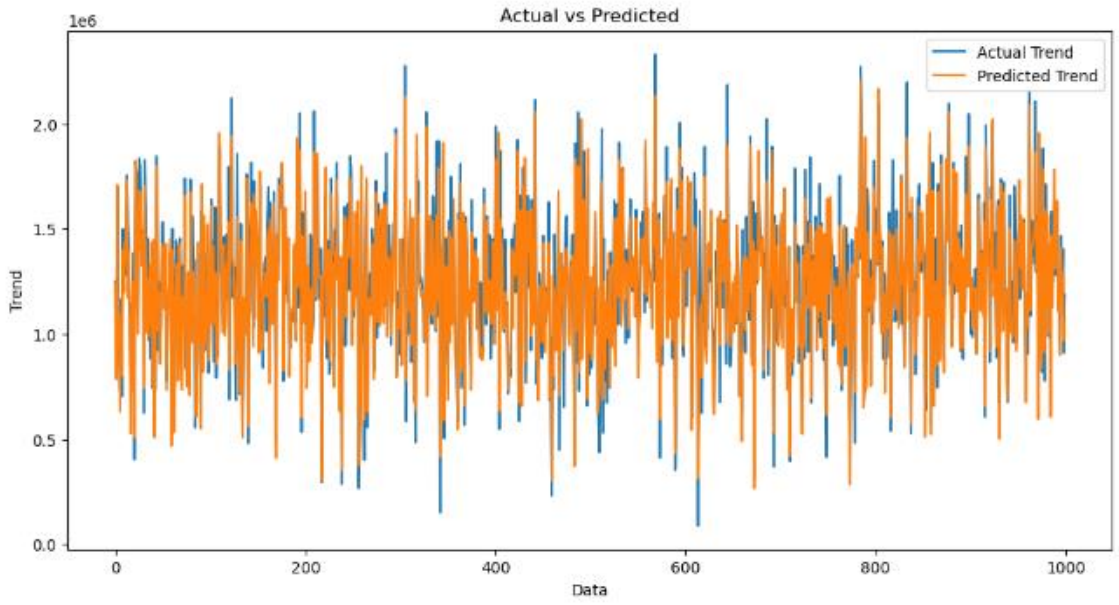
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, 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)
```

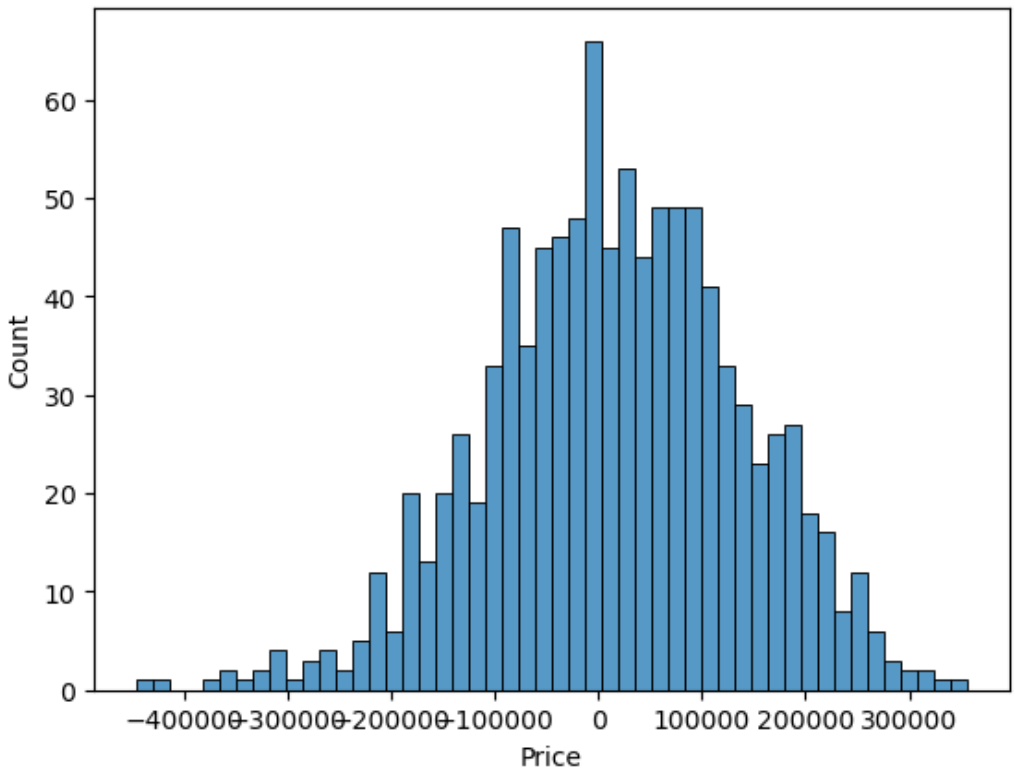
Evaluation of Predicted Data:

```
  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
```