ai-phase4

November 5, 2023

1 PREDICTING HOUSE PRICES USING MACHINE LEARNING

Development Part 2

Loading the dataset:

The first step is reading the dataset from the csv file we downloaded.

[ ]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
dataset = pd.read_csv("/content/USA_Housing.csv")
```

[ ]:
```
pd.options.display.float_format = '{:20.2f}'.format
dataset.head(n=5)
```

[ ]:
```
      Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0             79545.46                 5.68                       7.01
1             79248.64                 6.00                       6.73
2             61287.07                 5.87                       8.51
3             63345.24                 7.19                       5.59
4             59982.20                 5.04                       7.84

   Avg. Area Number of Bedrooms      Area Population                Price  \
0                          4.09             23086.80           1059033.56
1                          3.09             40173.07           1505890.92
2                          5.13             36882.16           1058987.99
3                          3.26             34310.24           1260616.81
4                          4.23             26354.11            630943.49

                                             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
```

let’s get statistical information about the numeric columns in our dataset.

```
[ ]: dataset.describe(include=[np.number], percentiles=[.5]).transpose().
     ↪drop("count", axis=1)
```

```
[ ]:                                     mean           std  \
     Avg. Area Income                68583.11      10657.99
     Avg. Area House Age                 5.98          0.99
     Avg. Area Number of Rooms           6.99          1.01
     Avg. Area Number of Bedrooms        3.98          1.23
     Area Population                 36163.52       9925.65
     Price                         1232072.65     353117.63

                                          min           50%  \
     Avg. Area Income                17796.63      68804.29
     Avg. Area House Age                 2.64          5.97
     Avg. Area Number of Rooms           3.24          7.00
     Avg. Area Number of Bedrooms        2.00          4.05
     Area Population                   172.61      36199.41
     Price                           15938.66    1232669.38

                                          max
     Avg. Area Income               107701.75
     Avg. Area House Age                 9.52
     Avg. Area Number of Rooms          10.76
     Avg. Area Number of Bedrooms        6.50
     Area Population                 69621.71
     Price                         2469065.59
```

Then, we move to see statistical information about the non-numerical columns in our dataset:

Data Preprocessing:

Data cleaning:

Dealing with Missing Values: We should deal with the problem of missing values because some machine learning models don't accept data with missing values. Firstly, let's see the number of missing values in our dataset. We want to see the number and the percentage of missing values for each column that actually contains missing values.

```
[ ]: num_missing = dataset.isna().sum()

     num_missing = num_missing[num_missing > 0]

     percent_missing = num_missing * 100 / dataset.shape[0]

     pd.concat([num_missing, percent_missing], axis=1,
               keys=['Missing Values', 'Percentage']).\
               sort_values(by="Missing Values", ascending=False)
```

```
[ ]: Empty DataFrame
     Columns: [Missing Values, Percentage]
     Index: []
```

```
[ ]: dataset["Price"].value_counts()
```

```
[ ]: 1059033.56    1
     1521141.34    1
     1148372.40    1
     2065710.16    1
     1749820.01    1
                  ..
     1444701.33    1
      788427.84    1
      875904.53    1
      984421.23    1
     1298950.48    1
     Name: Price, Length: 5000, dtype: int64
```

```
[ ]: dataset['Price'].fillna('No feature', inplace=True)
```

let's check if there is any remaining missing value in our dataset:

```
[ ]: dataset.isna().values.sum()
```

```
[ ]: 0
```

Converting categorical features into numerical representations:

We will encode our categorical features using one-hot encoding technique which transforms the categorical variable into a number of binary variables based on the number of unique categories in the categorical variable.

```
[ ]: dataset[['Address']].head()
```

```
[ ]:                                              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 = pd.get_dummies(dataset)
```

```
[ ]: address_oneHot = [c for c in dataset.columns if c.startswith("Address")]
     dataset[address_oneHot].head()
```

```
[ ]:    Address_000 Adkins Crescent\nSouth Teresa, AS 49642-1348  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
     4                                                  0

        Address_000 Todd Pines\nAshleyberg, KY 90207-1179  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
     4                                                  0

        Address_001 Steve Plaza\nJessicastad, UT 25190  \
     0                                               0
     1                                               0
     2                                               0
     3                                               0
     4                                               0

        Address_0010 Gregory Loaf\nSouth Ericfort, VA 34651-0718  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
     4                                                  0

        Address_00149 Raymond Knolls\nNew Jason, UT 75026  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
     4                                                  0

        Address_002 Katherine Flat\nHartmanland, AZ 37973-3049  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
     4                                                  0

        Address_0022 Young Rest\nLake Kevin, CA 25438-1821  \
     0                                                  0
     1                                                  0
     2                                                  0
     3                                                  0
```

```
4                                                 0

   Address_0029 Melinda Neck Apt. 591\nLake Gregoryshire, HI 60267  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   Address_003 Erica Passage Apt. 274\nEast Ritaton, KS 97261-8617  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   Address_003 Fernando Gateway Suite 145\nEast Danielshire, GA 76041  …  \
0                                                 0                    …
1                                                 0                    …
2                                                 0                    …
3                                                 0                    …
4                                                 0                    …

   Address_Unit 9446 Box 0958\nDPO AE 97025  \
0                                         0
1                                         0
2                                         0
3                                         0
4                                         0

   Address_Unit 9463 Box 0963\nDPO AE 49984-2796  \
0                                              0
1                                              0
2                                              0
3                                              0
4                                              0

   Address_Unit 9494 Box 2307\nDPO AE 58622  \
0                                         0
1                                         0
2                                         0
3                                         0
4                                         0

   Address_Unit 9664 Box 1605\nDPO AA 30902  \
0                                         0
1                                         0
```

```
2                                        0
3                                        0
4                                        0

   Address_Unit 9732 Box 1846\nDPO AE 69898-3304  \
0                                             0
1                                             0
2                                             0
3                                             0
4                                             0

   Address_Unit 9774 Box 4511\nDPO AE 44963  \
0                                        0
1                                        0
2                                        0
3                                        0
4                                        0

   Address_Unit 9778 Box 2114\nDPO AP 59374  \
0                                        0
1                                        0
2                                        0
3                                        0
4                                        0

   Address_Unit 9785 Box 0790\nDPO AP 60371-0797  \
0                                             0
1                                             0
2                                             0
3                                             0
4                                             0

   Address_Unit 9831 Box 7128\nDPO AA 54705  \
0                                        0
1                                        0
2                                        0
3                                        0
4                                        0

   Address_Unit 9871 Box 9037\nDPO AP 37275-9289
0                                             0
1                                             0
2                                             0
3                                             0
4                                             0

[5 rows x 5000 columns]
```

Feature Selection:

Selecting the most relevant features for predicting house prices.

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

Model Building and Evaluation:

Splitting the data:

we need a training dataset to train our model and a test dataset to evaluate the model. So we will split our dataset randomly into two parts, one for training and the other for testing.

```
[ ]: 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_state=101)
```

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

```
[ ]: 3413          1305210.26
     1610          1400961.28
     3459          1048639.79
     4293          1231157.25
     1039          1391232.53
     Name: Price, dtype: float64
```

```
[ ]: Y_train.shape
```

```
[ ]: (4000,)
```

Standardizing the data:

To make all algorithms work properly with our data, we need to scale the features in our dataset.

```
[ ]: from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_train_scal = sc.fit_transform(X_train)
     X_test_scal = sc.fit_transform(X_test)
```

Modeling:

We have used xgboost regression algorithm.

```
[ ]: from xgboost import XGBRegressor
     from sklearn.model_selection import RandomizedSearchCV

     parameter_space = \
         {
             "max_depth": [4, 5, 6],
             "learning_rate": [0.005, 0.009, 0.01],
             "n_estimators": [700, 1000, 2500],
             "booster": ["gbtree",],
             "gamma": [7, 25, 100],
             "subsample": [0.3, 0.6],
             "colsample_bytree": [0.5, 0.7],
             "colsample_bylevel": [0.5, 0.7,],
             "reg_alpha": [1, 10, 33],
             "reg_lambda": [1, 3, 10],
         }

     clf = RandomizedSearchCV(XGBRegressor(random_state=3),  parameter_space, cv=3,␣
      ↪n_jobs=4,scoring="neg_mean_absolute_error" ,random_state=3, n_iter=10)
     clf.fit(X_train, Y_train)
     print("Best parameters:")
     print(clf.best_params_)
```

```
Best parameters:
{'subsample': 0.6, 'reg_lambda': 10, 'reg_alpha': 10, 'n_estimators': 2500,
'max_depth': 4, 'learning_rate': 0.009, 'gamma': 7, 'colsample_bytree': 0.7,
'colsample_bylevel': 0.5, 'booster': 'gbtree'}
```

we build our XGBoost model with the best parameters found:

[ ]:
```
xgb_model = XGBRegressor(**clf.best_params_)
```

we train our model using our training set:

[ ]:
```
xgb_model.fit(X_train_scal, Y_train);
```

Predicting Prices:

[ ]:
```
y_pred = xgb_model.predict(X_test_scal)
print(y_pred)
```

```
[1245976.5   782677.56 1693791.1   965170.56  979755.2   646750.8
 1057040.    830071.8  1431099.9  1197648.1  1397959.1  1316015.9
 1695341.4  1276931.8  1377154.1  1182152.1   576097.4   954917.
 1199935.4  1202382.5   516015.78 1721938.8  1761200.2  1185399.9
 1075547.1  1781688.4  1726844.8  1442473.4  1334613.2  1507267.1
  771825.25 1715043.9  1415200.6   994509.44 1226292.9   898196.56
 1211128.    995412.94 1306071.1   766165.4  1392926.2   710576.3
  816781.25 1829430.1  1610588.    964923.4  1100228.1   834148.5
 1152134.9  1455172.5  1426714.6  1155611.6  1084399.   1355186.5
  811177.06  981478.9  1127577.1  1262368.2  1412904.    553008.
 1401322.1  1119077.1   666898.2  1205247.2  1317447.4  1312847.
  770164.1  1462357.   1378863.1   903097.25  832729.1  1193881.4
 1065133.6  1805220.5   938752.75  815205.75  824481.6  1387455.2
  716566.75 1629987.9  1048518.44 1337776.   1272283.8  1161088.6
  780454.2   653229.56 1037503.75 1359368.6  1030076.   1259515.5
  530128.7  1665477.9   839129.25 1585533.2  1003546.7   914340.94
 1177607.9  1497553.9   904518.1   963354.94 1293866.   1594332.2
  977592.6  1425910.6  1020598.1  1429731.2   875411.    811756.7
  983460.06 1781253.6  1850731.2  1374917.   1020346.6  1054523.2
 1443067.6   989831.4  1138557.2  1353490.4  1251818.5  1522629.
  761960.2  1375488.9  2003338.9  1408435.8  1309945.9  1260702.1
  901227.1   787137.94 1658782.9  1156781.4   804902.9   797000.75
 1500750.2   807861.06  612415.8  1214568.2  1015149.2  1127418.2
 1743913.8  1146609.9   587511.44 1366554.2  1340100.8  1682448.6
 1146391.8   940850.2  1714339.6  1071124.9  1603927.9  1404990.2
 1355839.6   936140.9  1743655.9  1220214.4  1189614.1   954079.9
 1215688.5  1232872.4  1235361.4  1130275.8  1501174.   1518622.1
  754458.94 1507055.8  1249687.4  1043025.4  1285001.2  1590073.4
 1125203.1   452490.9  1585616.8  1305087.5  1691244.6  1657147.4
 1643898.4  1789545.5  1540740.2   863019.1  1398801.9  1553175.9
 1375250.6  1098951.8  1423329.4   862656.25  933576.4  1135039.2
 1161830.4  1075511.2  1315606.1  1121894.5  1005345.   1860049.6
 1268599.9  1158853.6  1864908.6  1138624.2   638460.7  1494228.8
 1192738.2  1098275.8  1691183.1   839590.2   959550.44 1312380.8
  887882.3  1292466.4  1218735.8   963816.1  1199474.   1955396.6
 1469954.2  1115057.   1897259.   1531206.4  1229241.8  1255883.4
 1278953.5   290238.72 1097409.2  1261194.4  1360012.6  1709601.5
 1037033.   1021330.06 1419971.5   901380.3   849442.9  1650343.
 1152343.5  1360353.6   805469.75 1499488.2  1294056.4  1788028.1
 1224186.   1209327.5   617314.94 1339496.4   447641.34 1234403.1
 1363001.2  1555654.1  1010476.8  1095950.8  1422622.1  1549270.2
 1318741.5  1767452.8  1668699.1   931584.44 1236547.1  1097623.
 1467367.   1147653.6  1541595.4  1592310.4   432503.12  443719.06
  830936.75 1797482.6  1160272.6  1266255.1   577449.2  1355219.5
 1763136.2   590837.    878033.   1233764.8  1208419.2  1146874.1
 1538212.9  1324988.2   784592.9   891839.06 1703797.9  1019139.5
 1285343.9  1538439.8  1104124.   1068997.2  1706038.9  1395827.1
 1073209.2  1588123.9  1069633.8  1678661.1  1001016.56 1307823.4
 1289877.2  1298618.6  1016938.3  1320756.2  1123396.6  1507990.1
 1721597.5  1959184.9   821205.5   996996.2  1097749.    953779.6
 1372524.6   795225.3  1204508.9  1292104.5  1471064.2  2180610.
  773127.94 1164970.8  1058797.   1250029.8  1599913.    918090.75
  982963.   1549428.6   779682.9   845983.4   580195.56 1767558.6
 1489818.1  1580570.2  1221700.5  1297775.2  1049536.   1167059.9
 1105472.1  1100124.4  1008109.2  1927116.    753471.5  1098812.9
 1307260.1  1546845.4  1085846.   1117448.1  1506715.6  1423899.8
 1132030.4  1006307.1  1661975.8  1089029.4  1856908.    952819.1
  346407.78 1408254.9  1320984.1  1912898.    489201.56 1063913.4
 1359668.2  1016659.8   787136.06 1450740.2  1063137.6  1631813.
 1093017.1  1085331.5  1694577.4  1095436.8   830426.4   835882.
  554968.4  1502822.4  1211074.8  1663131.4   824240.1  1023279.5
 1700703.    620658.75 1187908.2  1560388.9  1354619.   1100883.
```

```
1282813.5   925556.7   771257.94 1482999.2  1369798.5  1213235.6
1190807.2  1482028.5   921034.5  1336724.4  1070298.2   931002.94
 997134.8   929309.94  966817.9   998306.7  1590590.5  1142046.9
1306055.   1607217.1  1124848.   1253416.   1267358.1  1059206.
1026984.4  1421992.9  1277962.2  1044154.75 1901730.   1486683.1
1184002.8  1936545.5   742343.56 1673703.2  1432946.6  1414837.9
1074053.5  1038924.   1172525.9  1030370.56 1064830.1   775582.06
 939073.7   747804.06  922793.94 1366583.1  1062912.   1317879.1
1418336.   1310271.9  1497128.8  1864314.6  1107268.8   689385.94
1574437.8  1467783.6   711861.5  1700591.5  1291877.   1751504.
 862532.94 1285684.6  1559868.1  1343288.2  1619827.1   938728.
1653645.1  1125976.4  1489790.1  1196318.1  2135133.8   984016.9
1055945.4   682549.8  1173353.1  1226275.9  1198459.9   915578.94
1489788.   1271242.2   942266.56 1197275.9  1137599.   1124778.5
1437862.2   654494.56  900762.5   443186.38 1299275.8   782028.25
1218192.9  1070768.4   916558.9  1140144.9  1752021.5   773920.25
 862507.5  1304139.6  1108315.9  1262812.4   807677.5  1200762.9
1356081.1  1402304.5  1209729.9   946211.2   936970.25 1399076.4
1220192.1  1411655.1  1046302.2   401768.38 1824399.8   973521.94
1544128.4  1856346.1   786378.94  969751.6  1987432.2   900800.94
1083391.   1745806.8  1332908.5  1311908.6   780018.   1810830.
1034729.25  830476.1  1219121.2   943481.06  984116.06 1278354.8
1177597.1  1504065.9  1130752.6  1247004.9   669171.3   691544.3
 824334.94 1443703.2  1677580.6   569366.5  1250314.    867449.7
1146322.2   781388.1  1087824.4  1225604.9   771644.5  1343021.1
 931433.7  1052017.4  1089444.4  1554351.    835691.44 1375206.9
1522953.5  1233689.   1833609.   1642582.1  1076420.4  1213561.1
1770265.2  1272218.6  1334652.9  1021813.25 1316050.   1016482.9
 988201.   1427469.5  1532037.9  1188567.8  1246590.8  1251875.6
1421474.4  1130517.6  1600271.9  1353386.1   774400.6  1060507.2
1038687.25 1391861.8  1413668.5  1407016.8  1424464.4  1245281.2
1876200.5  1266697.8  1586297.   1325942.8  1315383.9  1209999.1
1265549.5  1098198.5  1639430.4  1387093.8  2157573.   1187432.2
 877825.3  1225363.1  1464421.4   629806.3  1349396.8   865794.6
1084080.6  1320550.4  1652045.6  1426706.6  1787394.9  1085703.4
1300738.4  1196895.8  1183503.6  1736492.6   932610.06 1470126.5
1252651.    446844.06  692946.44 1679756.5  1147381.6  1321366.1
1968538.6  1523796.1  1775115.4   991818.2   925658.8  1366290.5
1413616.6  1380025.   1687260.1  1366366.2  1564729.6  1609838.
 611232.4   689609.9  1234354.8  1451159.6  1205531.6  1510732.
1014189.7   303631.9   624789.9  1280737.1  1116437.9   797897.2
1212271.9  1155165.5  1776785.4  1030367.06 1074907.9  1096802.8
1420397.9  1449720.2  1322090.2  1657481.   1084365.    783851.25
 629339.8   878952.94 1304738.2  1186189.5  1542826.4  1165603.9
 862954.94 1631025.2  1356114.   1560916.9  1255929.   1481472.6
 928438.3  1001459.4  1996264.2  1285360.1  1309003.5   885674.6
1030029.75 1092262.9  1234420.6  1008874.56 1215803.9  1054272.1
1512556.1  1274990.   1156334.6   697369.9   982182.3   627935.8
```

```
1219436.2  1288516.8  1084584.8   980087.9  1392481.9  1125518.1
1338029.2  1228745.4  1805644.5  1341879.4   869452.4  1479116.2
 331757.34 1127875.   1388183.6  1318914.6  1277145.6  1822160.5
1400598.8  1368933.4  1319854.4  1389816.    801364.9  1275176.5
1359650.8  1812865.1  1279415.4   809123.6  1182148.2  1091760.1
1508063.8  1853060.8   946213.7   587864.2  1539335.1  1062738.2
1408702.9   916358.3  1032541.75 1231909.6  1449071.8  1544221.6
 928575.1  1323936.5  1676390.8  1187192.6  1255788.6   797818.06
1393172.6  1270070.6   621111.25 1170629.2   850614.2  1014719.9
 901985.5  1699878.5  1214295.1  1215884.4  1409923.6   882242.25
1157256.1   898086.6   616174.1   530164.   1378557.    901861.94
1552457.   1310285.5  1378908.5  1097680.2  1247210.6  1465812.4
 764144.06 1191786.1  1273239.2  1132335.6  1444914.9  1522958.1
 872887.   1390783.6  1083704.4  1598745.6  1048219.8  1525205.9
1075228.9   892859.6  1400960.1   887289.7   636850.06 1629932.5
1218258.6  1165690.1  1633724.4  1412738.   1474008.   1022468.25
1305780.5  1164340.1   940028.   1233231.1  1134003.2   900107.4
1492382.2   860287.2   783484.25 1020520.44 1288748.1  1539281.2
 957283.25 1349522.8  1138144.4  1371796.1   894565.1   318679.84
 911370.2  1261531.5  1150554.9  1030250.94  708236.44 1444839.9
1346901.4  1267581.4  1517509.2  1478923.1  2265659.8  1845236.9
1399298.6   760599.75  792399.94 1869568.5   784597.1  1552403.9
1434101.4  1123988.8  1206693.4   707274.44 1265355.6  1171994.8
1676765.2  1135779.2  1489720.8  1434538.9  1342014.5  2095187.9
1448472.5  1352305.8   981513.25 1451969.1   898102.1  1233375.6
 753672.25 1207163.9  1014065.2  1214686.8  1428926.6  1492431.2
 557040.5  1434635.4  1738693.2  1272745.6  1090518.8  1218287.
1471589.1  1247179.9  1054782.5  1070079.6  1292918.6  1697415.4
1174217.9  1359843.6   815448.1  1115188.9  1411011.8  2029898.4
1398824.4  1109956.   1532414.2   569935.8  1563458.9  1458894.4
1233438.2  1215933.    894181.1   965583.8  1489627.4  1180626.5
1633399.9  1509939.5  1389534.9  1523796.1  1432239.1  1248323.5
 575496.44  875970.44 1608503.1  1182655.   1306509.4  1970749.6
 522989.53  917944.8  1696131.1   776021.44 1258215.   1317308.8
1284588.1   775757.75 1718330.2   999868.94 1385762.8  1847418.2
1149017.9  1116997.5   807198.5  1343219.   1802343.1  1456662.1
1747298.8  2056707.6  1288537.6  1268475.    965236.3  1159365.
1764642.4  1266590.2  1341641.8  1685384.2  1238580.9  1508624.5
1375219.8   828293.7  1380072.2   946751.1   912664.25 1300478.5
1051415.2  1678548.8  1474428.   1385545.4  1948774.9  1513439.1
1114300.6  1217379.1  1052901.4  1105137.2   948342.56 1316873.4
1025434.25 1147679.5  1475264.1  1666635.2  1158569.4  1440629.8
1260026.8  1143126.5  1604010.6   598211.94 1867749.   1459621.
1326905.9  1358410.9   956941.94 1709474.8  1387140.1  1927115.8
1172539.4  1407902.9  1199941.4   867723.06 1452420.1  1524343.
 560629.1  1257197.9  1760381.6  1478394.4  1061130.   1706774.2
1108828.2  1259384.9   840677.    681229.2  1021644.9  1635088.8
1036119.2  1033206.25 1037025.   1025052.3  1536212.   1257599.6
```

```
  886728.1    959474.75 1370379.8  1766732.1  1256239.   1460836.1
 1217907.4  1289199.   1540044.9  1040962.25  727866.4  1183574.1
 1109017.4  1511241.   2076077.2  1169826.1  1439917.4  1391185.4
 1451599.1  1505049.4  1965671.4  1277948.5  1559252.8   619865.56
 1984418.5  1054574.4  1507661.6   892922.5  1751824.4  1137804.6
  848432.4  1665711.8  1263333.2  1497017.6   982795.   1505217.6
  548114.75 1481378.   1186960.5  1468239.5  1724554.1  1310675.8
 1447338.8  1578446.5  1059445.6  1276654.    904483.06 1393240.8
 1337484.5  1331973.6  1120435.   1049438.8 ]
```

Evaluation of Predicted Data:

Then we evaluate the model performance by comparing its predictions with the actual true values in Y_test using the MAE metric :

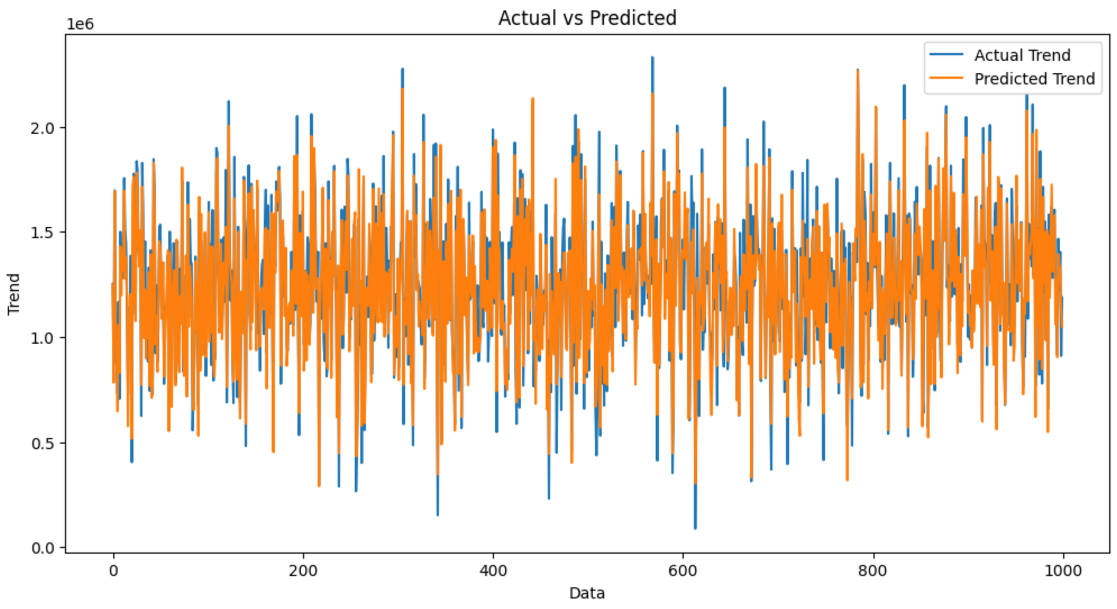
```
[ ]: from sklearn.metrics import mean_absolute_error
     xgb_mae = mean_absolute_error(Y_test, y_pred)
     print("XGBoost MAE =", xgb_mae)
```

```
XGBoost MAE = 88382.87552199
```

Evaluation of predicted data using visualisation tool:

```
[ ]: import matplotlib.pyplot as plt
     plt.figure(figsize=(12,6))
     plt.plot(np.arange(len(Y_test)), Y_test, label='Actual Trend')
     plt.plot(np.arange(len(Y_test)), y_pred, label='Predicted Trend')
     plt.xlabel('Data')
     plt.ylabel('Trend')
     plt.legend()
     plt.title('Actual vs Predicted')
```

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


Actual vs Predicted
1e6
Actual Trend
Predicted Trend
Trend
Data
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
1000