Regresión Lineal Múltiple: Gradiente Descendente (Sklearn)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
```

Queremos predecir los precios de una serie de casas, a partir de las siguientes variables:

- Avg. Area Income: Renta media de los residentes de la ciudad donde está la casa
- Avg. Area House Age: media de antigüedad de las casas de esa ciudad
- Avg. Area Number of Rooms: Número medio de habitaciones en las casas de esa ciudad
- Avg Area Number of Bedrooms: Número medio de dormitorios en las casas de la ciudad
- Area Population: Población de la ciudad
- Price: Precio de la casa (variable objetivo o variable target)
- Address: Dirección

In [2]:
```
USA_Housing = pd.read_csv('data/USA_Housing.csv')
USA_Housing.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Micha 674\nL
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Jo Sui K
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9 Stravenue\
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barr
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Ray

Exploremos el dataset:

In [3]:
```
USA_Housing.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
```

Son todas numéricas.

In [4]:
```
USA_Housing.describe()
```

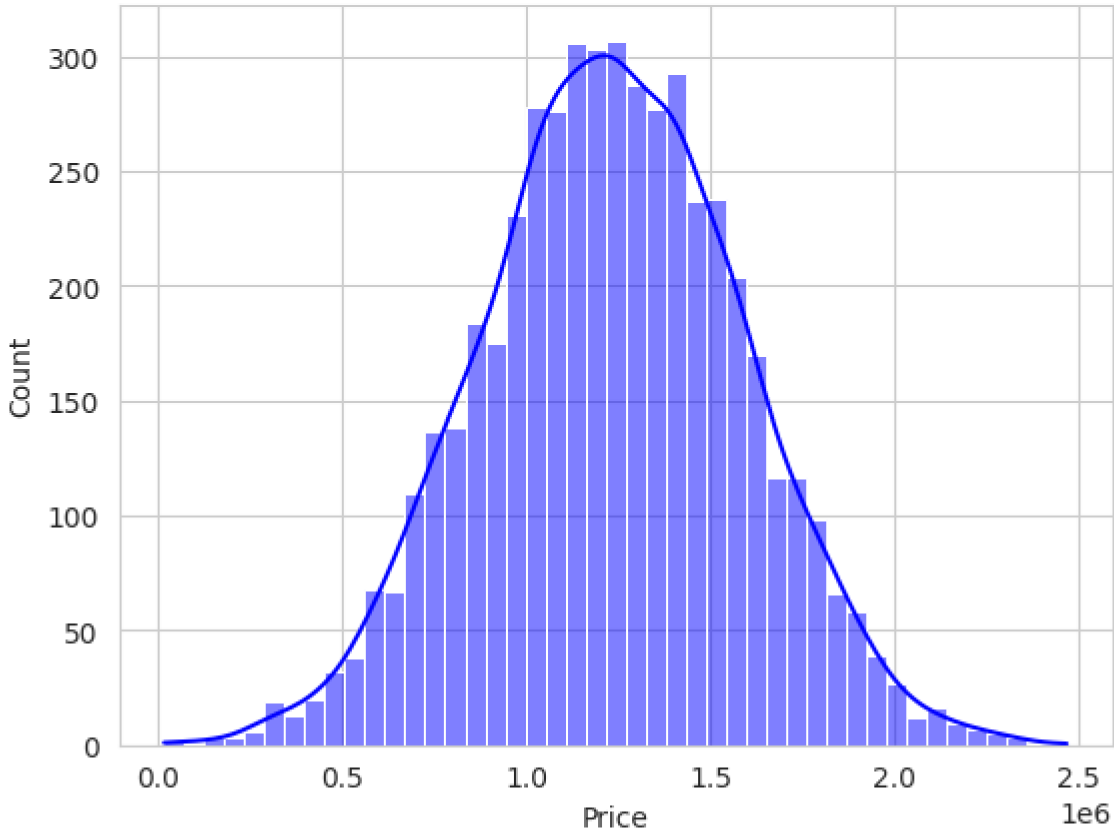
Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Pri
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+(
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+(
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+(
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+(
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+(
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+(
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+(
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+(

EDA: Exploratory Data Analysis

Variable target/objetivo: El Precio

In [5]:
```python
sns.histplot(USA_Housing['Price'], color='Blue', kde = True)
plt.show()
```



Tiene una distribución en forma de campana de Gauss, una distribución normal, y por tanto eso es una buena señal para aplicar regresión lineal.

Multivariante

Vamos a analizar las correlaciones de las numéricas entre sí, pero en especial con la variable Target:

In [6]:
```
USA_Housing.corr(numeric_only= True)
```

Out[6]:

	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 [7]:
```
sns.heatmap(USA_Housing.corr(numeric_only = True),annot=True);
plt.show();
```

Deberíamos esperar que la variable que más influya sea el income del area (Avg. Area Income), como nos pasó con el modelo de ejemplo en la unidad de familiarización del sprint anterior.

Entrenar un modelo de Regresión Lineal

En primer lugar, dividimos en train y test. Esto se debería haberhecho antes del EDA (apartado anterior).

In [8]:
```
USA_Housing.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 [9]:
```
# Feautures
X = USA_Housing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of
       'Avg. Area Number of Bedrooms', 'Area Population']]

# Target
y = USA_Housing['Price']
```

In [10]:
```
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 [11]:
```
print(X.shape)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(5000, 5)
(4000, 5)
(1000, 5)
(4000,)
(1000,)
```

Y ya sí que tienes que ir tomando nota de cómo lo creamos a partir de `sklearn`:

In [12]:
```
from sklearn.linear_model import LinearRegression

# Creamos un objeto
lm = LinearRegression()
```

Entrenamos

Entrenar es básicamente obtener el valor de los parámetros (en este caso de regresión lineal también les llamaremos pesos) a partir de los datos de train. En este caso mediante optimización con gradiente descendente del error cuadrático medio (para que se te quede la terminología). Ese entrenamiento es lo que esconde el método `fit` (ojo el método fit por dentro hará otras cosas en otro tipo de modelos, sólo que Sklearn tiene la gracia de mantener la sintáxis independientmente del tipo de algoritmo y modelo)

In [13]:
```
# Entrenamos con los datos de train
lm.fit(X_train, y_train)
```

Out[13]:
```
▾ LinearRegression
LinearRegression()
```

Interpretación de los pesos

Recuerda que el modelo por dentro es como una "función" del tipo:

$$y = w_0 + w_1x_1 + w_2x_2 + ...$$

Veamos el parámetro w_0, también conocido como intercept y que nos da el valor para el caso de que los valores de todas las features sea 0:

In [14]:
```
lm.intercept_
```

Out[14]:
```
-2635072.900933358
```

Y ahora los coeficientes (pesos o parámetros):

In [15]:
```
lm.coef_
```

Out[15]:
```
array([2.16522058e+01, 1.64666481e+05, 1.19624012e+05, 2.44037761e+03,
       1.52703134e+01])
```

Vistos como un dataframe para interpretarlos mejor:

In [16]:
```
coef_df = pd.DataFrame(lm.coef_, X.columns,
                       columns=['Coefficient'])
coef_df
```

Out[16]:

	Coefficient
Avg. Area Income	21.652206
Avg. Area House Age	164666.480722
Avg. Area Number of Rooms	119624.012232
Avg. Area Number of Bedrooms	2440.377611
Area Population	15.270313

Interpretación de los coeficientes

Manteniendo fijos el resto de coeficientes:

- Un incremento de 1 unidad en **Avg. Area Income** equivale a un incremento de **21.64 dólares**
- Un incremento de 1 unidad en **Avg. Area House Age** equivale a un incremento de **164,666 dólares**
- Un incremento de 1 unidad en **Avg. Area Number of Rooms** equivale a un incremento de **119,624 dólares**
- Un incremento de 1 unidad en **Avg. Area Number of Bedrooms** equivale a un incremento de **2,440 dólares**
- Un incremento de 1 unidad en **Area Population** equivale a un incremento de **15.27 dólares**

En definitiva nuestro modelo de regresión implementa la siguiente función:

$Precio = 21.652206 \times \text{Avg. Area Income}+ 164666.480722 \times \text{Avg. Area House Age} + 119624.012232 \times \text{Avg. Area Number of Rooms} + 2440.377611 \times \text{Avg. Area Number of Bedrooms} + 15.270313 \times \text{Area Population}$

Predicciones de nuestro modelo

Primero probemos a ver que precio asignaría a una casas en un área de ingresos medios de 100000 dolares, 20 años de antigüedad media, 8 habitaciones de media, 8 dormitorios de media y población media 100000 habitantes:

In [17]:
```
new_home = np.array([[100000, 20, 8, 8, 100000]])
lm.predict(new_home)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:465: UserWarning: X does n
ot have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[17]:
```
array([5327023.75160817])
```

A partir de las predicciones sobre el conjunto de test, vamos a construir una gráfica que compare valores reales y valores predichos:

In [18]:
```
X_test
```

Out[18]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
1501	61907.593345	7.017838	6.440256	3.25	43828.947207
2586	57160.202243	6.893260	6.921532	3.13	43467.147035
2653	70190.796445	6.745054	6.662567	2.01	29215.136112
1055	69316.796889	6.300409	7.873576	4.28	24448.211461
705	72991.481649	3.412866	6.494081	2.48	50626.495426
...	...	...	...	...	...
4711	77267.656264	3.939501	8.342808	6.09	22487.712072
2313	75967.135085	5.939370	6.111658	2.32	38897.091584
3214	81013.615294	7.149797	7.239105	5.44	45472.049451
2732	86762.882864	6.530193	5.106962	2.09	47724.581355
1926	67071.830617	4.935155	7.632398	5.04	32084.743400

1000 rows × 5 columns

In [19]:
```
predictions = lm.predict(X_test)
predictions
```

Out[19]:
```
array([1308587.92699759, 1237037.22949434, 1243429.34030681,
       1228900.2136037 , 1063320.9071083 , 1544058.05034861,
       1094774.70493019,  833284.72339225,  788412.85578719,
       1469714.86615709,  671728.43662062, 1606818.2197796 ,
       1004166.61331065, 1796798.9759592 , 1288566.96221026,
       1087782.93301076, 1423072.37492533, 1078178.68169677,
        802286.03537898,  930761.03695709, 1134829.86477822,
        916398.42023144, 1489972.69335433, 1284580.15538816,
       1582071.35322737, 1132519.15991992, 1089888.39644517,
        974510.51872155,  924057.96820648, 1740759.72092282,
       1286481.59512311, 1621289.95171608, 1435264.20161719,
       1234014.77924477, 1485434.57300368, 1718335.00753702,
       1538953.74882858,  777106.64791791, 1765201.5224362 ,
       1175972.14199818, 1553707.94323485,  897703.67505179,
       1371049.80326609,  845281.72310359, 1201022.89803887,
       1133285.98450866, 1363128.14557346, 1449814.08768277,
       1574363.90467358, 1233577.50265968, 1484464.01606216,
       1295276.58943552, 1222136.77335268,  990124.416598  ,
       1693824.96035766, 1823785.05665104, 1136495.63903364,
       1282164.4030563 , 1327292.05443142, 1353355.51909084,
        966265.4434595 ,  661906.63917329, 1533750.56860995,
       1002479.76053669,  995799.79959532, 1567349.59707253,
       1500813.6348258 , 1090078.00056418, 1820964.84741698,
       1479856.16724372,  902785.30216071, 1494542.21679596,
       1378859.29762363,  962610.88478574,  712800.76347468,
       1565650.37425303, 1149218.97654373,  931311.21938348,
       1600923.95574327,  506875.42639189, 1592924.03164877,
       1292023.54953336,  681260.98813933,  432977.16109994,
       1395334.6347121 ,  696834.50740128,  663613.864406  ,
       1030075.82567009, 1485134.02140133, 1320071.75329114,
       1271264.11329309, 1422274.66428998,  671601.89583322,
       1149995.6427159 , 1261301.82190815,  784691.75366843,
       1189915.30485926, 1014887.17531198, 1405378.85714967,
       1539076.97995918,  593301.15079211, 1391801.76393823,
       1262742.70864071, 1875132.68141892, 2336899.33387147,
        946302.25269161, 1315655.58002704,  953107.00437483,
       1831758.27483199, 1625882.69024163, 1639245.35999713,
       1267016.02457602, 1804840.84718875, 1220829.4610425 ,
        850670.85934626, 1584187.59595383,  761904.58839427,
       1360032.80169299, 1186062.16197453, 1559470.32105874,
       1770323.74482795, 1608248.53684166, 1573661.86962665,
       1531208.64376206, 1812547.75249824, 1124965.37862364,
        669517.224712  ,  978249.45590806, 1316294.24560647,
       1627775.48942899, 1343329.23943758, 1088746.21767432,
       1516813.82322279, 1333391.56256394, 1421961.53699943,
       1527965.13951325, 1674048.56946955, 1301125.76637805,
        843902.14276135, 1439397.74699529, 1857462.44038405,
       1090388.23396923, 1845464.91767036, 1272015.37672474,
       2036878.56552244,  882686.73444799,  954142.09498712,
       1116676.58455362, 1395124.12307566, 1510721.91006259,
        952949.66077822,  969883.09423411, 1320445.66982241,
       1494548.61488448, 1311797.03936308,  764593.88740277,
       1729945.04026368,  966390.95323435, 1980014.56655174,
       1253548.56040187, 1378043.01470111, 1465705.85673355,
       1041643.25777139, 2016178.12307681, 1425286.3175589 ,
        783766.05598743,  826322.77792498, 1753896.36530123,
       1077079.69502184, 2073701.72252871, 1735967.08463102,
       1049433.84156028, 1829294.89844271, 1128895.7961378 ,
       1629029.35283931, 1634470.42260714, 1524785.70964563,
        722156.7942414 ,  740342.62131184,  547622.83148167,
```

```
 828931.14665374, 1067364.15017838,  969253.52659223,
1227292.55245932,  871755.1279931 ,  504202.0296593 ,
1200044.47873915, 1062310.88487693, 1519500.99297217,
1329223.54689577, 1386542.64820164, 1320937.40536549,
1204359.90632358, 1240619.19034576, 1365626.9148404 ,
1769953.37847158, 1329720.91129389, 1376332.51764146,
1298990.36407937,  834744.40310685, 1272066.20149398,
1451499.03548572, 1289190.82085876,  896873.03598594,
 990437.07011063, 1357842.65851709, 1649364.37151005,
 779324.24924976, 1005588.56759384, 1467831.81435144,
1124908.45994293, 1142768.93653069, 1250196.34738781,
1105424.47695199,  622294.93861749, 1841657.4121255 ,
1182731.53995535,  609160.57633567, 1410661.53419293,
1263529.22101339, 1119673.80759916, 1145399.95692905,
 500449.33290925, 1109883.63039254, 1626981.19957438,
 797675.84432486, 1407092.10326887, 1560733.66870769,
1698894.16984527, 1714320.59543073, 1216152.65562658,
1114774.40354011, 1001067.89613668, 1065269.68436436,
1625179.02290147, 1593962.78450219, 1526221.35988258,
1129983.7733289 ,  735887.82286947, 1593872.80909074,
1732256.512932  , 1264953.46131034, 1216712.64382586,
1539828.39332793, 1748273.31154007,  544482.7116165 ,
1177808.65915496, 1404019.93663772, 1125393.51909426,
 545088.45128038, 1290749.88927799, 1658361.89913343,
1487203.28008246, 1291143.03838428, 2050455.16760487,
 768874.19747942, 1863631.84714419,  929819.1776411 ,
1761083.62018238, 1659001.7508367 , 1247438.96208988,
 713427.40837866, 1778756.53025782,  461932.4961174 ,
2167306.95873267,  664672.42431929, 1801836.62367735,
1265099.18805368, 1139217.30086013, 1173989.48300958,
1748192.41949273, 1147793.31302196, 1029923.90766742,
 947126.10538086, 1182469.49434718,  927601.20005207,
1566514.6168574 , 1175827.55938911,  388087.5480148 ,
1288959.37006155,  986658.19197291, 1340019.64945833,
1044418.8454109 , 1286342.56336184, 1531000.55785367,
1667768.88543345, 1084796.51405691,  720967.43062553,
 811997.72885678, 1328871.66384419, 1519499.84416136,
1379415.47205365, 1457746.58240929,  998979.7397246 ,
2252173.82126787, 1491966.70905194, 1349013.08643536,
1042666.25727798,  486559.46844396, 1792549.96363365,
1193458.84754759, 1247203.26503456, 1122698.47045892,
1004013.2801844 , 1699121.458163  , 1504252.14207274,
1432835.58985638, 1342786.68025942, 1601414.77106588,
 821506.76488333, 1126578.53904205, 1291476.44544071,
1698968.44222494, 1232602.93325317, 1489643.67510287,
1233262.35005555, 1135048.11749452, 1732127.82573492,
1348162.21747027,  621085.06641216, 1501389.10755781,
1201749.81984393, 1326727.70232237, 1097551.72695622,
1221290.66261422,  788262.02543881,  935562.28249792,
1287412.17867094, 1473888.67757759,  609893.86640502,
1183181.37838122, 1035484.44410105,  809285.45806938,
 655567.08949324,  983743.80081315,  986631.41757965,
1320596.67974115, 1196954.70045297,  961010.9197037 ,
1275847.42178952, 1425242.66569545, 1654320.54624645,
1521722.43867609, 1396978.62698709, 1413827.92718159,
1219885.02000691, 1232374.59901518, 1654016.6338654 ,
1328324.05157965, 1321091.57152382, 1299216.74514684,
1347305.90163784, 1688687.80000856, 1574519.36949693,
1045335.69889281, 1356261.70016953, 1439514.97152501,
1504156.15649866, 1138145.82597034, 1500051.72294341,
```

```
       1486912.30437995,  385027.92485996, 1216217.42146599,
       1726329.91434918, 1207805.98489521,  894601.19092623,
       1240643.83179285, 1308440.22054391, 1437359.37891255,
        893239.6816687 , 1792787.36750136, 2076933.42343298,
        922567.65701978, 1458351.25849101, 1249772.50297778,
       1370805.23907779, 1121245.98909361,  963231.49570943,
       1327241.12918487, 1187327.23418151,  943716.62278246,
        424854.57298675, 1223724.58352513, 1411756.82716379,
       1236002.32151798, 1449021.08576   , 1411706.42320675,
        926881.67780973, 1006631.71873007, 1325140.65527614,
        470014.10664421, 1568408.54620669, 1071690.67501271,
       1261709.43785537,  983184.15258549, 1086377.76905145,
        726363.60820775,  814172.12713781,  803025.54888481,
       1619615.99437125,  459335.1366362 , 1300390.90011251,
        961516.33244582, 1025131.04020143, 1384778.89206911,
       1235853.6636986 ,  769788.89201937, 1026166.69741535,
       1405638.84930899, 1756291.11652804, 1280487.11746939,
        622591.98844795, 1449923.84156157, 1198418.94698591,
       1356968.16583373, 1079271.72703224, 1352329.77296904,
       1057107.45852888, 1290462.77150213,  705163.48550143,
       1200364.66997073, 1507695.00401124,  962337.84598504,
       1204232.84729754, 1571860.58065076, 1209464.73857384,
       1549865.08072301,  769645.60725032, 1404674.43922861,
       1240637.52783346, 1029500.080016  , 1335457.58612472,
       1088594.03736502, 1047144.04180909, 1388984.12610591,
        918924.0160505 , 1096407.50174596, 1406966.35948352,
        806952.29876924, 1196187.41426745, 1010141.32136387,
       1394711.98478671, 1257521.42006913, 1299693.11290885,
       1294970.9428875 , 1723105.21964978, 1711247.39032017,
       1174489.95194178,  897740.7411008 , 1356640.83884511,
       1219851.03447608, 1366674.22214344, 1644472.14734961,
        724989.37808678, 1570999.00692573,  924989.3629067 ,
        673283.26497668, 1589694.70648669, 1373578.191451  ,
       1236758.80570283, 1939038.5944139 , 1248340.03854151,
       1393184.15037955, 1703469.17720891, 1059230.86749246,
       1268676.36389753, 1304697.18785125,  892460.12277115,
        964864.99058499,  818820.07759942, 1401580.54747611,
       1180874.94215976, 1259247.43803591, 1702902.04493212,
       1649087.11983957, 1731289.88335348, 1125719.05993076,
        865172.86237615,  968232.22605794,  784711.55849769,
        781822.82332169, 1209737.47415573,  972635.23672595,
       1576919.83616695, 1060076.42506713,  694196.84219451,
       1610587.67993026, 1309658.61630397, 1141463.96987411,
       1205906.98599986, 1281336.18444202, 1171886.92407685,
       1567896.45068229, 1322659.49232211, 1166515.84227933,
        789697.3883071 , 1143720.96771711,  648247.70636865,
       1071126.44204641, 1428574.57982158, 1424322.83234084,
       1057501.45875364, 1430509.55156469, 1324079.70373624,
       1292014.89736633, 1071126.32745123, 1287175.72484579,
       1019167.68338671, 1037478.15143998, 1049936.76364929,
        731789.18467474,  782568.77309977, 1002203.86002292,
       1067321.67597074, 1124201.97076035, 1457954.83235619,
       1108354.56185649, 1598770.0790141 , 1176022.8557993 ,
       1096986.73368016, 1247383.41805666, 1363559.48178172,
        773372.64579653,  822330.64613229,  993393.81206808,
       1543915.09520886, 1493186.91902697, 1175891.67528418,
       1253692.12639576, 1521005.7668647 , 1627802.51180477,
       1686082.39992538, 1980124.37797509,  977460.80106582,
       1109687.22060459, 1113007.40549505, 1211544.64900677,
       1407486.32673228,  789313.25630096, 1234898.66189967,
```

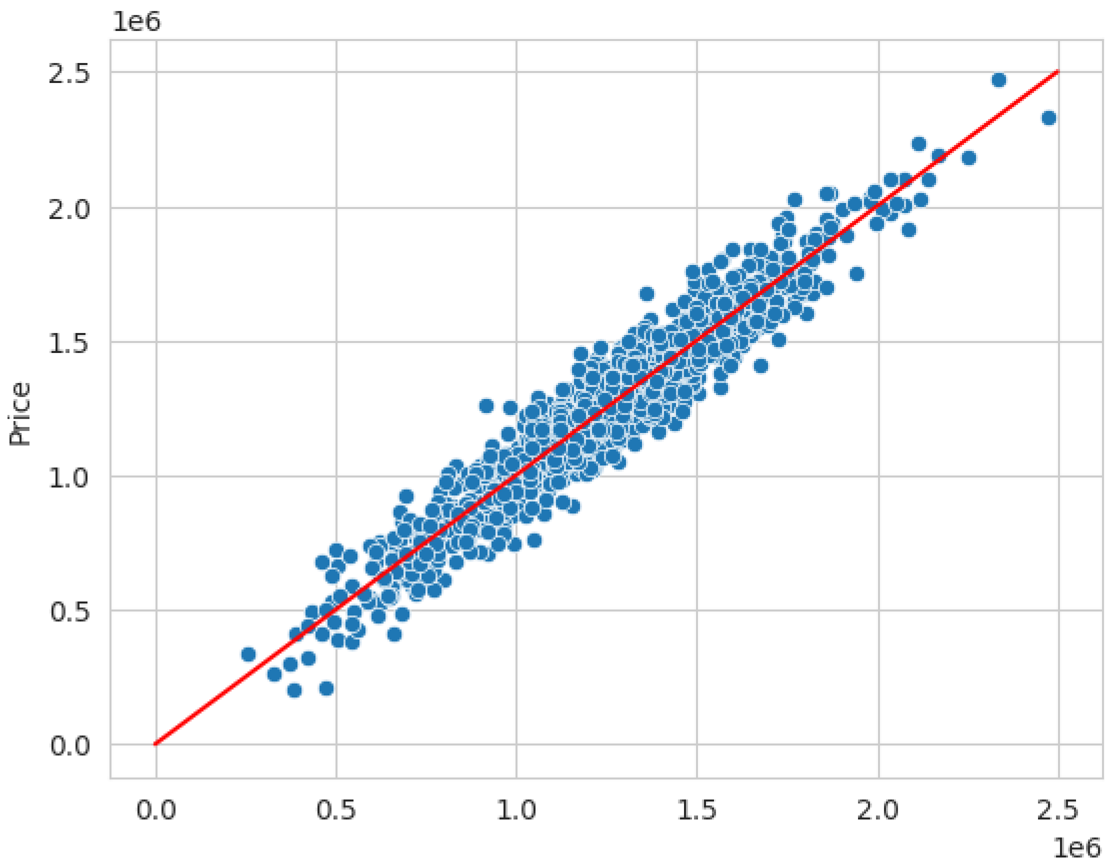
```
1209078.69432804, 1803383.95080046, 2114888.79328758,
1118853.92157738, 1206707.38468575, 1352852.99343918,
1396175.67664054, 1546374.75741952,  666842.59065989,
1481420.71952322, 1021477.45518557,  968486.88335857,
1424082.26366331, 1914787.24107153,  914204.87835356,
1489860.90731122,  693922.34630681, 1031062.51065993,
1352126.03254335, 1373512.52703589, 1049109.83383344,
1486525.47762769, 1310520.395959  , 1647941.99311859,
1283575.2405586 , 1452989.07762646, 1429414.83148144,
1034083.03488037, 1134634.51460329, 1403564.88673514,
1584896.61276078, 1746823.45559169, 1466283.25973611,
1294745.41392545, 1051682.47255808, 1101548.76032234,
1465167.00089535, 1382725.06572448,  939224.93035358,
1337350.76799395, 1120979.5949805 , 1319831.03770852,
 558962.93258731, 1401611.09978705, 1354544.07306021,
1306334.89969639, 2086259.1709201 , 1904134.11165538,
 539895.35619527, 1525528.05146316, 1235213.44650031,
 750639.88255382, 1127965.98258831, 1471490.85051141,
 619419.48787409, 1114997.61194405, 1061464.69123062,
1013604.11379727, 1581434.85227402, 1079499.50514177,
1055056.29175868, 1034589.95466526, 1224314.04495441,
 887268.18862751, 1516254.20491774, 1865327.56742207,
1027371.06429436, 1490055.38180712, 1337289.95134345,
1119421.33157709,  979092.31019062,  679557.25776095,
1325905.50178095, 1171590.73226993, 1567491.12331949,
1143856.1916678 , 1126148.6441649 ,  864154.38857824,
 490147.41546328, 1291071.27278512, 1038384.50618605,
1662679.55956739, 1160624.90934377, 2037284.10126651,
1516262.8932747 , 1229075.23311264, 1547218.08935961,
1434753.65392652, 1709132.86008374, 1281184.09975518,
1524044.07660809,  968015.40091304, 1293671.18547091,
1106474.84699344, 1431104.22750489, 1712068.30269068,
1160254.525011  ,  655742.41586139, 1226710.98389832,
 927585.91056686, 1267635.28046126,  891680.82011575,
1234379.98343492,  921258.67102359,  973622.38506841,
1211118.49159504, 1152861.17191027, 1769241.04652312,
1185559.02437247,  982633.08890845, 1332532.61460762,
1642473.93936996,  965382.57146925, 1281975.93146427,
1406343.90582019,  677673.85740124, 1209557.69139082,
 732921.10636371, 1682132.45242378,  872620.43651359,
1136143.09175228,  510258.75506176, 1288102.88621592,
1371033.33463936, 1310113.78224341, 1436855.3124118 ,
1051232.72974602,  684828.48739319, 1101019.83617955,
1567983.5129489 , 1018012.6821631 , 1050698.11270264,
 825827.00943831, 1173621.66011305, 1591981.75730177,
1595728.9914928 , 1360891.37529096, 1192754.82897989,
 655836.47938465, 1560112.78530447,  813169.25379903,
1132145.75967389, 1564590.92451782, 1118669.76459821,
1569760.45331562, 1383018.33137049, 1609536.23008603,
1045549.71074358, 1174361.92171496, 1447869.79959208,
1824783.76109705, 1639844.91468785, 1488884.78527549,
1427514.92585401, 1231681.33305854, 1368014.03189646,
1424922.73584783, 1551131.49668389,  914701.22781688,
1249841.22219094,  258045.70919502, 1230968.70157305,
1126752.12639155,  858783.44109483,  739043.62099859,
1044214.90471307, 1112387.60767863, 1223250.56468715,
1447909.23403633, 1328619.03653293, 1049307.01433403,
 769590.672671  , 1751850.8883347 , 1463027.89759092,
2139919.10952315, 1718412.13713037, 1115993.45943525,
1995828.29499036, 1193379.31596023,  660188.56028329,
```

```
        660678.46196094, 1798655.5503712 ,  470131.44658328,
       1364102.1650628 ,  889323.70683908,  706548.24954177,
       1399891.22458728, 1416306.27235871,  967240.3261329 ,
       1729652.56536683,  869958.83238847, 1437817.09450812,
       1122484.53203984, 1673255.03691765, 1440039.59230988,
       1500539.40538994,  971584.27416925, 1043184.93567322,
       1317543.16932975, 1007627.13026024, 1323354.4036553 ,
       1135667.17490206, 1510351.76679843,  993056.44789451,
       1208831.65832729, 1988655.64291126, 1230200.72390602,
       1390403.7227665 , 1262914.99801445, 1582327.50953628,
       1028414.83031282, 1672107.67442153, 1265752.82959188,
       1236601.4330695 ,  329710.11232855,  717954.11280875,
       1103733.13974501, 1188712.8982572 , 1261796.72795392,
       1193848.82152227, 1303384.58750897, 1076900.53575828,
       1169750.68418015, 1458470.38909864, 1190650.84525938,
       1360612.20338315, 1497401.76740674,  916161.01831792,
       1311284.98583231, 1579839.09403369, 1155915.54541585,
       1425748.49376125, 1287287.54555759, 1153862.22489226,
       1825980.94614807, 1049734.15221282,  909542.4747019 ,
       1253936.42188571, 1467206.44416935, 1639991.73405574,
       1521767.77491757, 1019769.2038237 ,  652300.40469396,
        643377.65271608, 1142025.91271089, 1263237.98299637,
       1258087.1181922 ,  589606.05536889, 1187007.1645645 ,
       1121740.50402049, 1808959.71031404, 1800055.63690862,
       1123524.63673949, 1425287.86106954, 1510529.71215093,
       1323072.46306192, 1443101.2262092 , 1296655.19033898,
       1551943.68696701, 1237292.00298139,  665445.4948569 ,
       1262709.17153438, 1048886.41492119, 1867303.54970987,
       1562894.27357002,  731321.70170494, 1239100.56464209,
        909236.28293362, 1752100.4228616 , 1602831.78580391,
       1504851.06631703, 1411568.94781546, 1394422.38818576,
        370722.62126879, 1672311.97769414, 2120154.65355491,
        421201.54386695, 1486648.1916993 , 1200115.30671939,
       1531544.96637826, 1031782.76469186,  842914.91778092,
        838950.40676537,  596819.74942471,  855574.21947013,
       1040351.51174884, 1499196.3943261 , 1353188.18645252,
       1167128.79448673,  980462.42074907, 1615652.70345447,
       1192255.12726468, 1280277.40202391,  542769.58059214,
       1121564.11390496, 1430520.80041342, 1187720.59708725,
       1293444.79096478, 1344677.83092678, 1183418.13056164,
       1508087.09800656, 1265155.24508896, 1342605.87097178,
       1277584.98121775,  733241.48650822, 1553389.38956841,
       1398171.05353062, 1187021.49985545, 1371024.29006941,
       1108983.89166003,  975275.643677  , 1202021.72545548,
       1933557.66279022,  753137.60441104,  972296.00805762,
       1448918.48607918, 1859400.9727867 ,  914399.57695675,
       1004535.08620298,  924307.91932906,  964217.30193321,
       1220872.72352638, 1266519.75438384, 1619407.66363033,
       1582521.89907631,  764723.6415234 , 1739052.84078002,
       1593274.88965408, 1677324.99055393,  982954.91497658,
       1370603.38372474,  906582.8568568 , 1631028.42785834,
        690988.17276997, 1573202.07842787, 1466498.8806214 ,
       1650267.04845579, 1182468.07831994, 1173494.86869978,
       1333497.96024403, 1256991.4637588 , 1859915.69316242,
        758934.1049994 , 1392542.74052297, 1711869.49819376,
       1561607.58741326,  886776.00191122, 1312020.95776131,
       1174661.91515472, 1071955.83874377, 1381315.05056435,
       1106592.4501098 , 1170614.82399644,  992963.00044652,
       1471298.27839524, 1531571.13792555, 1609332.5414591 ,
       1374687.26168952,  963514.8883381 , 1516653.6399768 ,
```

```
1161808.9337438 ,  874674.3630556 , 1168468.71316324,
 608242.40551369, 1499895.89008411, 1020508.80251935,
1657914.8618016 ,  870372.42975708, 1469286.71372893,
1585005.10990058, 2474725.66527183, 1312108.9754553 ,
 927776.12992532, 1291387.28478137, 1591848.64102578,
 601180.27473416, 1595036.52624192, 1186556.5830517 ,
1569918.89962038,  775297.74281763, 1498585.74619855,
 731227.25771125, 1504692.54832433, 1046686.00738147,
1305402.58575888, 1275949.48895667,  997331.42657929,
 617836.40659073, 1591112.86332969, 1400763.16673465,
1311860.51933211, 1197044.05948892, 1041880.02840117,
1530068.19561836,  967934.24578167, 1427009.14149174,
1165147.25822176,  869045.49036542, 1541101.84334579,
 952854.45148409, 1649273.5193357 , 1262902.39001211,
 630780.85419621,  576868.98552469, 1709538.25511129,
1627857.63000977,  838666.12045417,  878776.97048188,
1299345.44781369, 1160416.62138426,  749980.452074  ,
 983974.19311228, 1153309.90526991,  943191.01927423,
1123570.59504228, 1547446.72609465, 1298891.03780585,
 694670.38901358, 1042411.7239971 ,  858369.88200379,
1238168.96478324, 1069914.16183725, 1032803.77343619,
1215816.4019039 , 1344361.73315605, 1128683.07322058,
1451580.234031  , 1094467.12735961, 1242819.75475178,
1312634.56732398, 1501591.70729102, 1577868.55728488,
1263870.34957585,  966363.73113401, 1304929.99505727,
 948730.96917376, 1116943.5196458 , 1177297.49011037,
 988220.39094904, 1035263.71039914, 1114925.57767818,
1691529.13991524,  493943.88206235, 1819479.31451055,
1387271.64234129,  632696.71378532, 1224357.78570427,
1180743.31764254, 1036241.70518269,  921234.34851312,
1754054.11179498, 1583328.0618723 , 1563758.96394322,
1342866.59984789, 1173594.23690928, 1042902.57541101,
1318531.80156902, 1870001.48200105, 1663623.85817637,
1045096.55057759])
```

In [20]:

```python
sns.scatterplot(x=predictions,y=y_test);
plt.plot([0,2.5e6],[0,2.5e6],'red')
plt.show()
```



No parece un mal modelo, pero para poder evaluarlo correctamente usemos las métricas de error, comparando train error con test error

Evaluación a partir de las Métricas de error

- **MAE** (Mean Absolute Error): es el error medio (la más fácil de entender)
- **MSE** (Mean Squared Error): es más popular que el MAE ya que penaliza errores grandes
- **RMSE** (Root Mean Squared Error): es todavía más popular que el MSE porque está en las mismas unidades que la variable objetivo y
- **R^2** (Coeficiente de determinación): proporción de la varianza total de la variable objetivo explicada por la regresión

In [21]:
```
from sklearn import metrics
```

Train error

In [22]:
```
pred_train = lm.predict(X_train)
print('MAE train', metrics.mean_absolute_error(y_train, pred_train))
print('MSE train', metrics.mean_squared_error(y_train, pred_train))
print('RMSE train', np.sqrt(metrics.mean_squared_error(y_train, pred_train)))
print('R2 train', lm.score(X_train,y_train))
```

```
MAE train 81509.39331244402
MSE train 10256318867.482721
RMSE train 101273.485510684
R2 train 0.9179787435623722
```

Test Error

In [23]:
```
print('MAE test', metrics.mean_absolute_error(y_test, predictions))
print('MSE test', metrics.mean_squared_error(y_test, predictions))
print('RMSE test', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
print('R2 test', lm.score(X_test,y_test))
```

```
MAE test 80879.0972348982
MSE test 10089009300.894518
RMSE test 100444.06055558745
R2 test 0.9179971706834289
```

Importancia de variables

Recordemos los coeficientes:

In [24]:
```
coef_df.sort_values('Coefficient', ascending=False)
```

Out[24]:

	Coefficient
Avg. Area House Age	164666.480722
Avg. Area Number of Rooms	119624.012232
Avg. Area Number of Bedrooms	2440.377611
Avg. Area Income	21.652206
Area Population	15.270313

Una habitación extra incrementa el precio (y) en 2440 dólares, y un dolar extra en area income incrementa el precio en 21 dólares. Esto es el significado de los coeficientes, pero no quiere decir que el número de habitaciones sea más importante/relevante que el area income

Para conocer la importancia de variables tenemos que **estandarizar** los datos antes de entrenar el modelo

In [25]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_train)

X_train_scaled = scaler.transform(X_train)
```

In [26]:
```
X_train_scaled
```

Out[26]:
```
array([[-0.19049241, -0.12817719, -0.13160635,  0.12038585, -0.82761782],
       [-1.38876401,  0.43080443,  0.80028487, -0.55648895,  1.15829878],
       [-0.35012392,  0.46680752,  1.70375078,  0.03067955, -0.31904298],
       ...,
       [-0.22335061,  0.53809182, -0.36489661, -0.68697084,  0.11908894],
       [-0.92417067,  1.43077434,  2.26846315,  0.2753331 ,  1.39018355],
       [-0.69357335, -0.07762332,  0.89219611,  1.67801341, -0.00681852]])
```

In [27]:
```
lm_scaled = LinearRegression()
lm_scaled.fit(X_train_scaled, y_train)
```

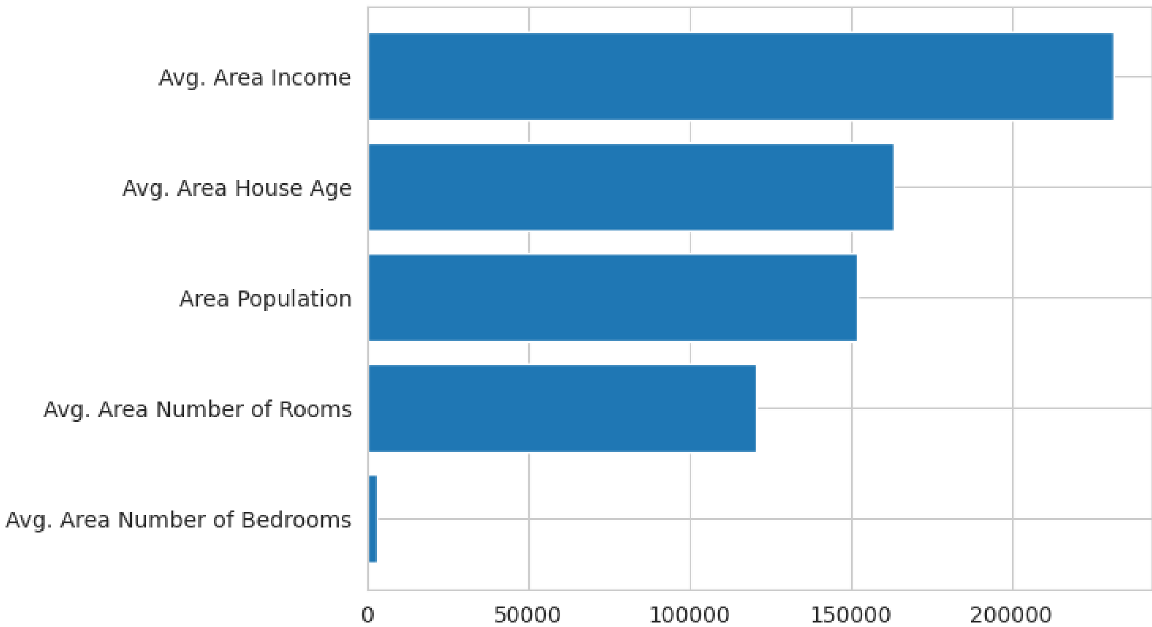
Out[27]:
```
▾ LinearRegression
LinearRegression()
```

In [28]:
```
feat_coef = pd.DataFrame(lm_scaled.coef_,
                         X_train.columns,
                         columns=['importance_standarized']).sort_values('importa
                                                                        ascending
feat_coef
```

Out[28]:

	importance_standarized
Avg. Area Income	231741.876652
Avg. Area House Age	163580.776566
Area Population	152235.900097
Avg. Area Number of Rooms	120724.771387
Avg. Area Number of Bedrooms	2992.449135

In [29]:
```
features = feat_coef.sort_values('importance_standarized')
plt.barh(features.index,features.importance_standarized)
plt.show()
```



Si lo comparamos con las correlaciones que obtuvimos en el EDA veras que hay una relación directa, pero no necesariamente "lineal".

Eliminar variables poco importantes

Para terminar, eliminemos las variables cuya importancia es mínima y veamos como afecta a las métricas del modelo (observa también que rápido es volver a crear un modelo y por lo tanto jugar con él)

In [ ]:
```
USA_Housing.columns
```

In [30]:
```
X_train.drop(columns='Avg. Area Number of Bedrooms',inplace=True)
X_test.drop(columns='Avg. Area Number of Bedrooms',inplace=True)

lm2 = LinearRegression()
lm2.fit(X_train,y_train)

pred2 = lm2.predict(X_test)

print('MAE test', metrics.mean_absolute_error(y_test, pred2))
print('MSE test', metrics.mean_squared_error(y_test, pred2))
print('RMSE test', np.sqrt(metrics.mean_squared_error(y_test, pred2)))
print('R2 test', lm2.score(X_test,y_test))
```

```
MAE test 80857.78944046368
MSE test 10073721633.872656
RMSE test 100367.93130214777
R2 test 0.9181214278738083
```

No hay apenas diferencia y además ganamos en velocidad y simplificamos. En general, no siempre, querremos utilizar el menor número de features posible.