دوره دیتا ساینس کاربردی

Linear Regression

رگرسیون خطی

dataroadmap

مدرس: مونا حاتمی

جلسه سیزدهم

Predict the house Price

https://www.kaggle.com/datasets/ashydv/housing-dataset

Predict the House Price

$50\ m^2$: $10,000

$100\ m^2$: $15,000

$150\ m^2$: $20,000

$200\ m^2$: $25,000

$250\ m^2$: $30,000

$300\ m^2$: $35,000

$350\ m^2$: ??????

Predict the house Price

$350\ m^2 : \$40{,}000$

$100 \times Area + 5{,}000 = price$

$b_1 \quad X \quad b_0 \quad y$

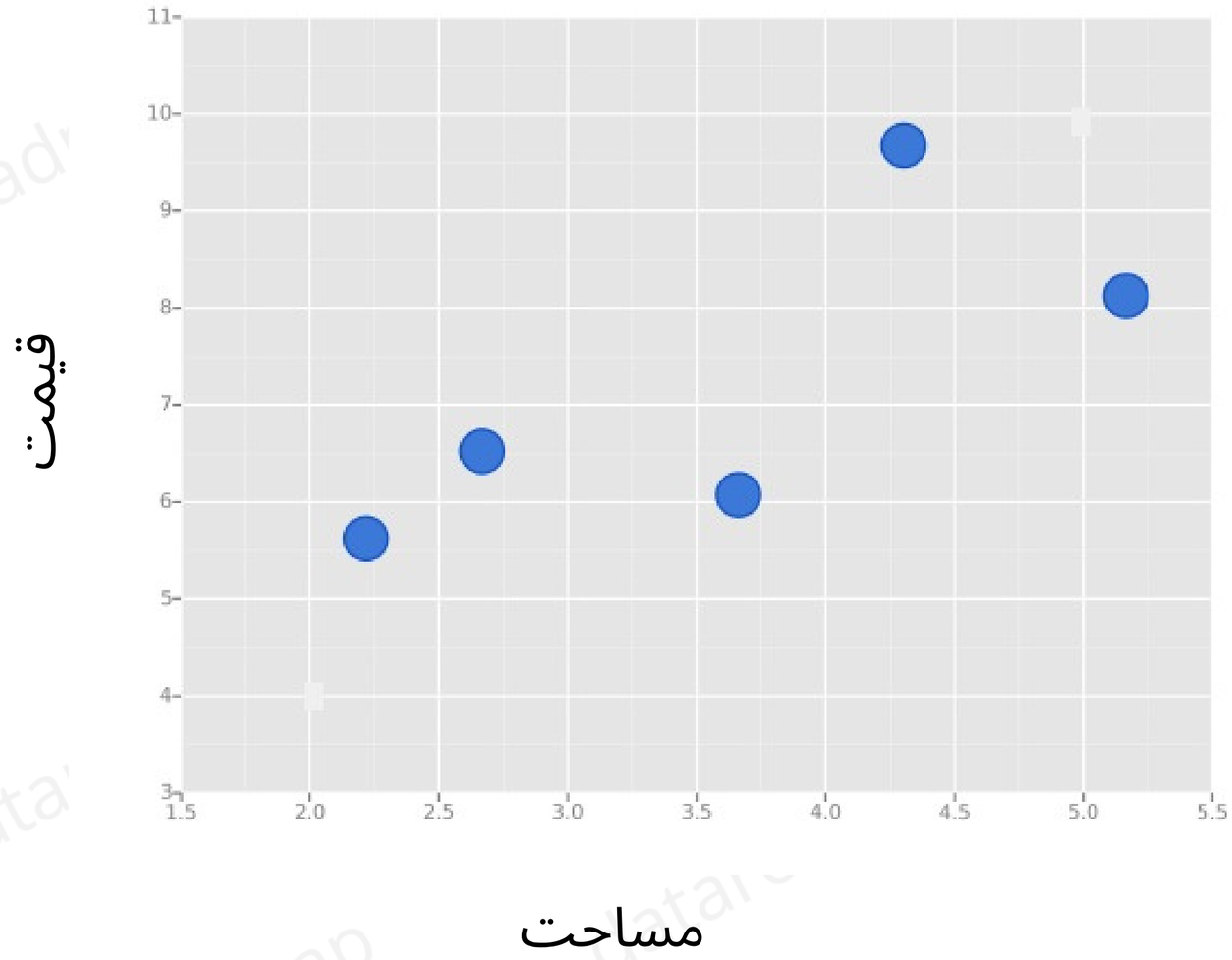
$$b_1 X + b_0 = y$$

معادله خط

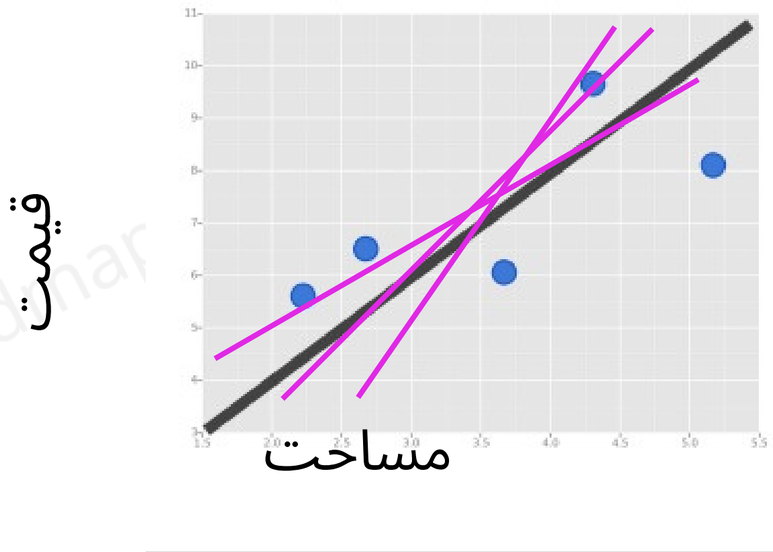
b_1 شیب:

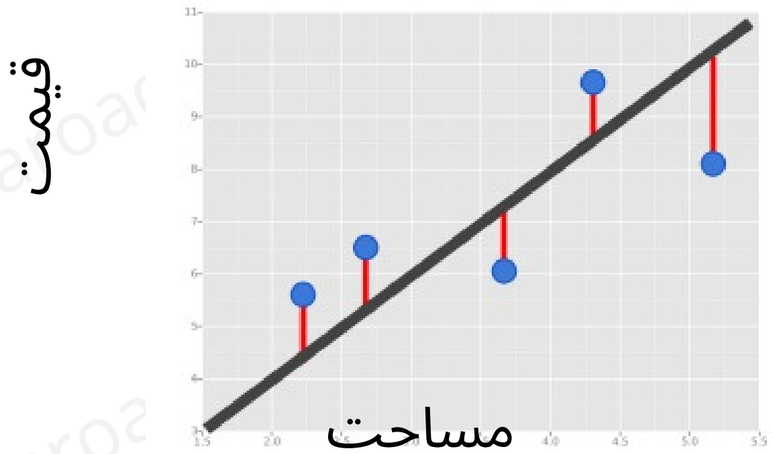
b_0 عرض از مبدا:

Predict the House Price

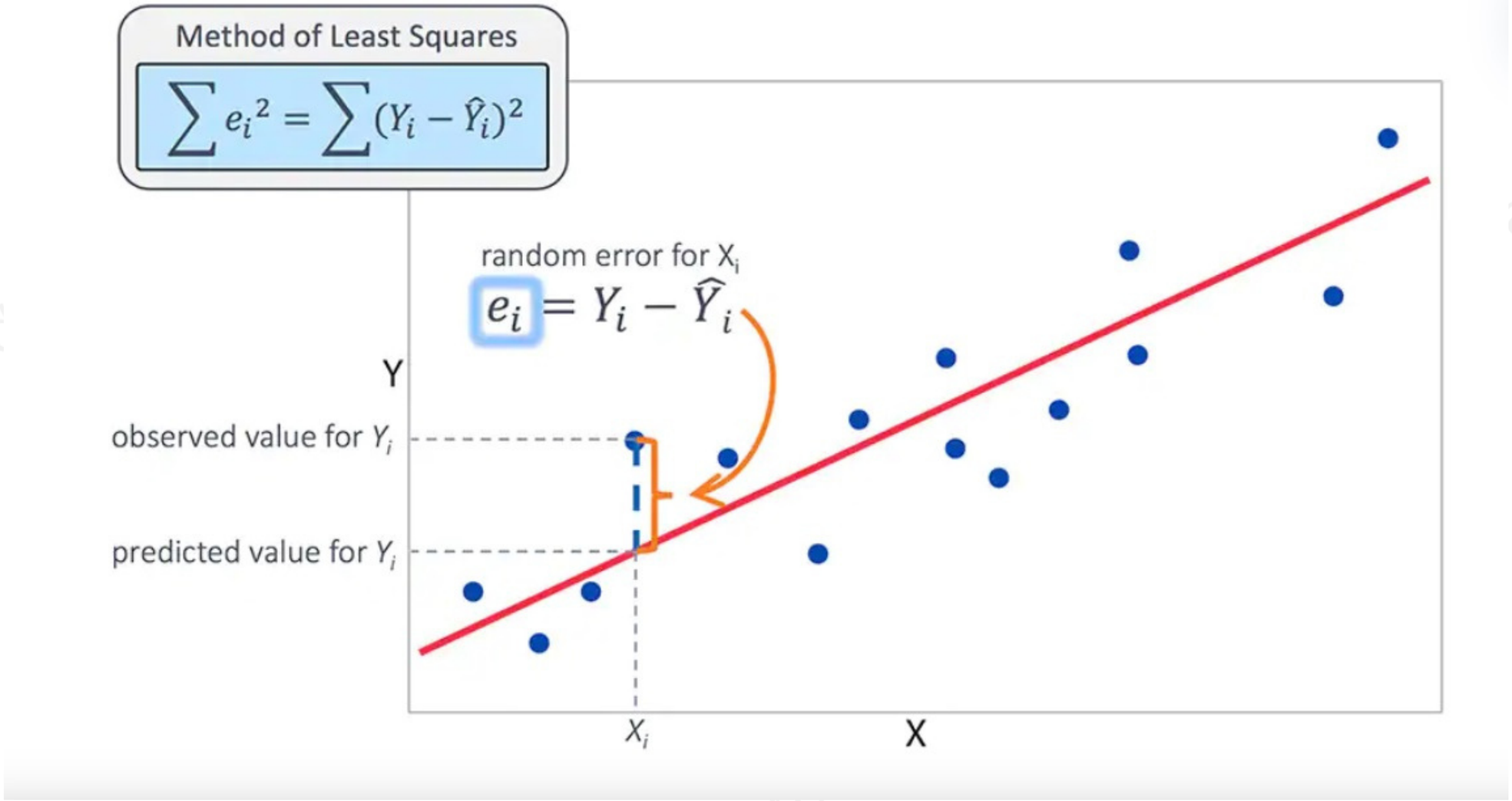


Linear Regression





Least Squares Method



Linear Regression

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Read Dataset

```
USAhousing = pd.read_csv('USA_Housing.csv')
```

```
USAhousing.head()
```

]:

	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
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05

Target

Read Dataset

```
USAhousing.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
```

Read Dataset

'Avg. Area Income': Avg. Income of residents of the city house is located in.درآمد متوسط ساکنان منطقه

'Avg. Area House Age': Avg Age of Houses in same city عمرمتوسط خانه ها

'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city متوسط تعداد اتاقهای هر خانه

'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city متوسط تعداد اتاق خوابهای هر خانه

'Area Population': Population of city house is located in جمعیت شهر

'Price': Price that the house sold at قیمت فروخته شده هر خانه

'Address': Address for the house آدرس

describe()

```
USAhousing.describe()
```

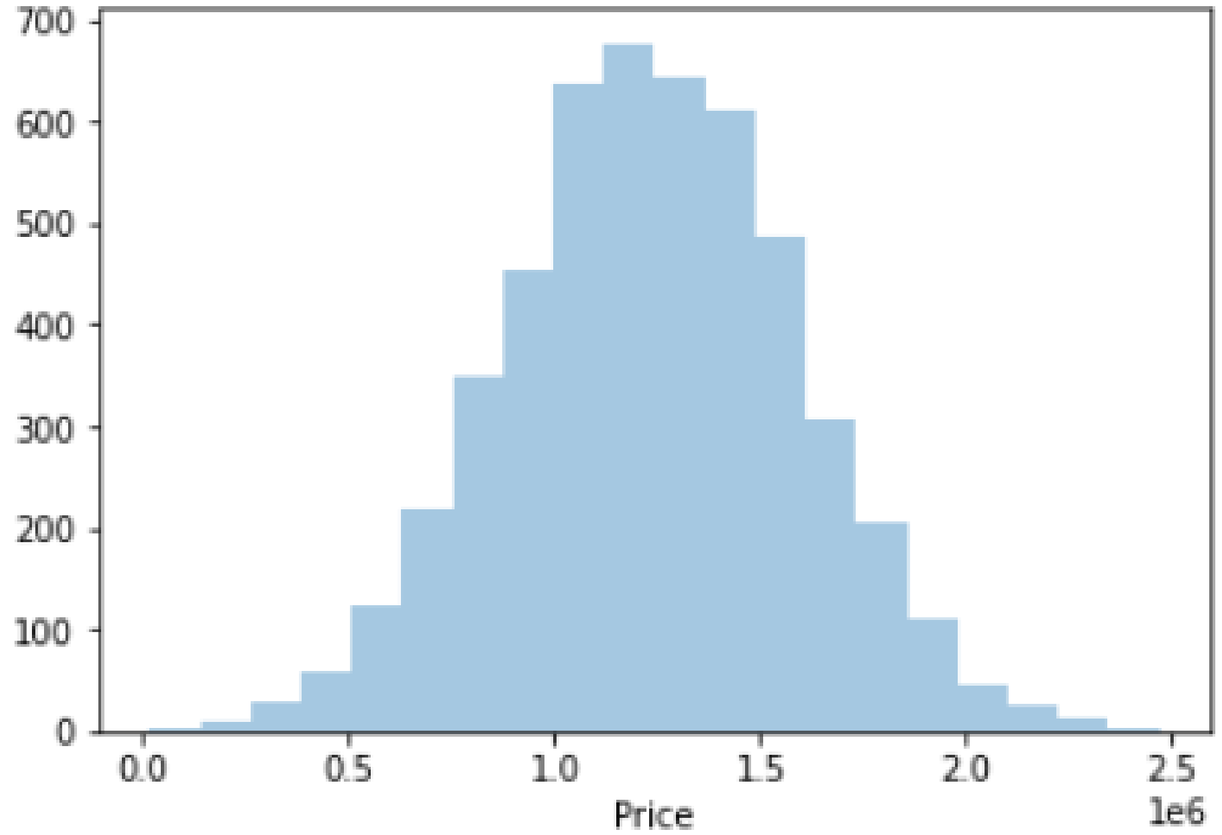
5]:

	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

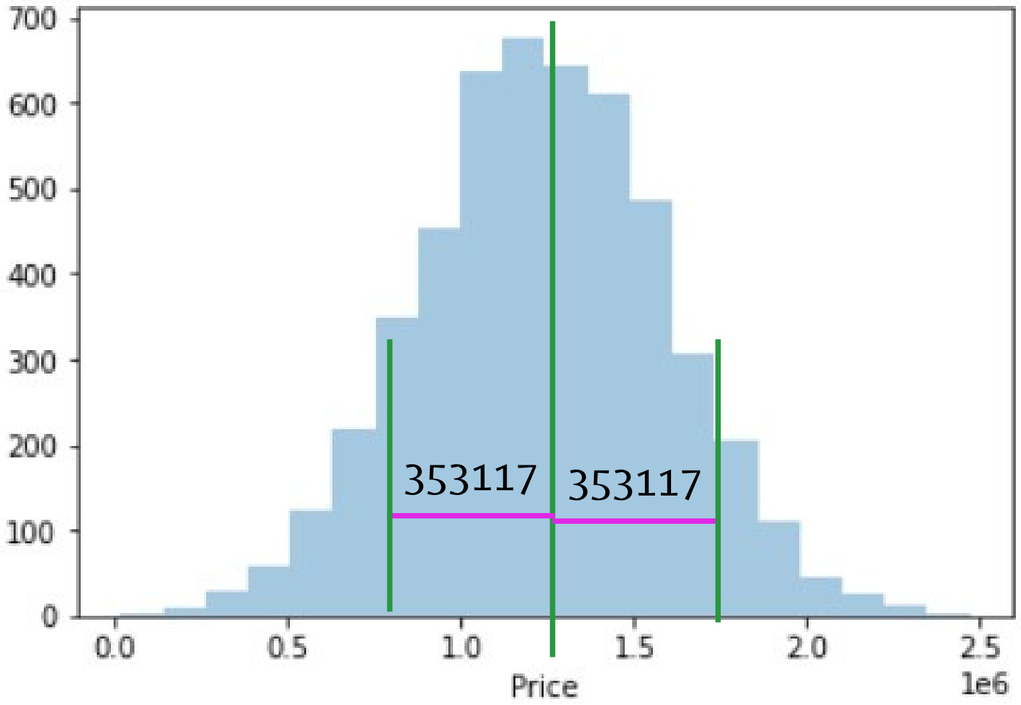
Exploratory data analysis (EDA)

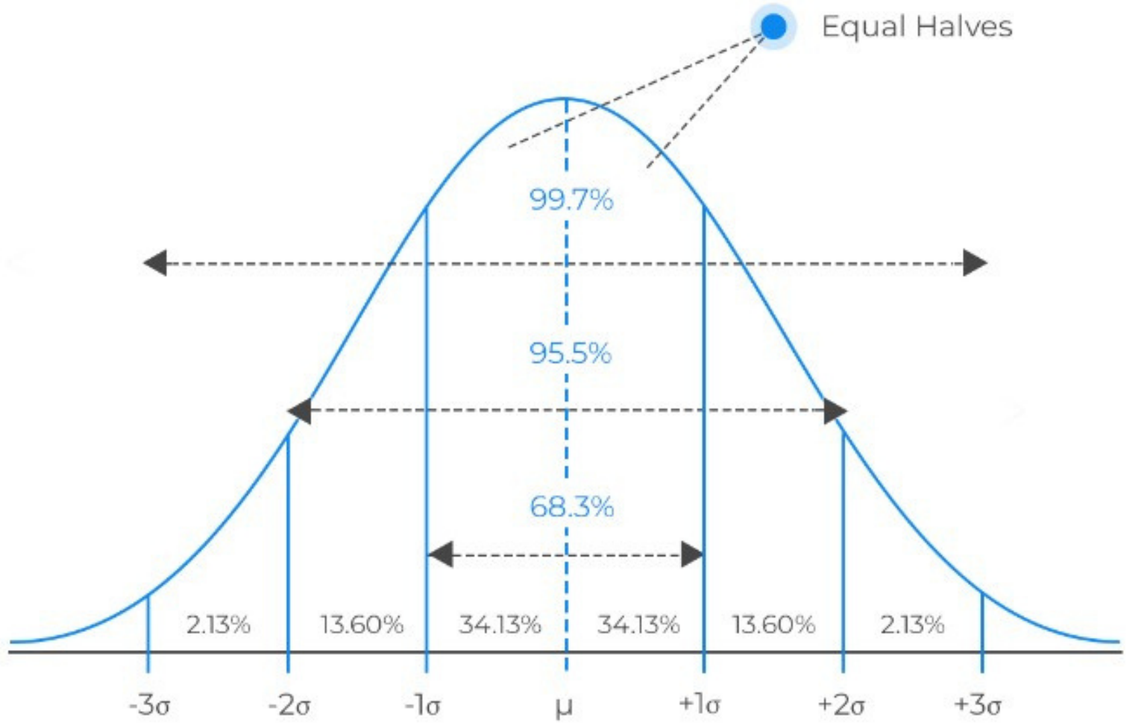
```
sns.distplot(USAhousing['Price'],bins=20,kde=False)
```

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



Price distribution plot






```
USAhousing['Price'].mean()
```

```
1232072.65414236
```

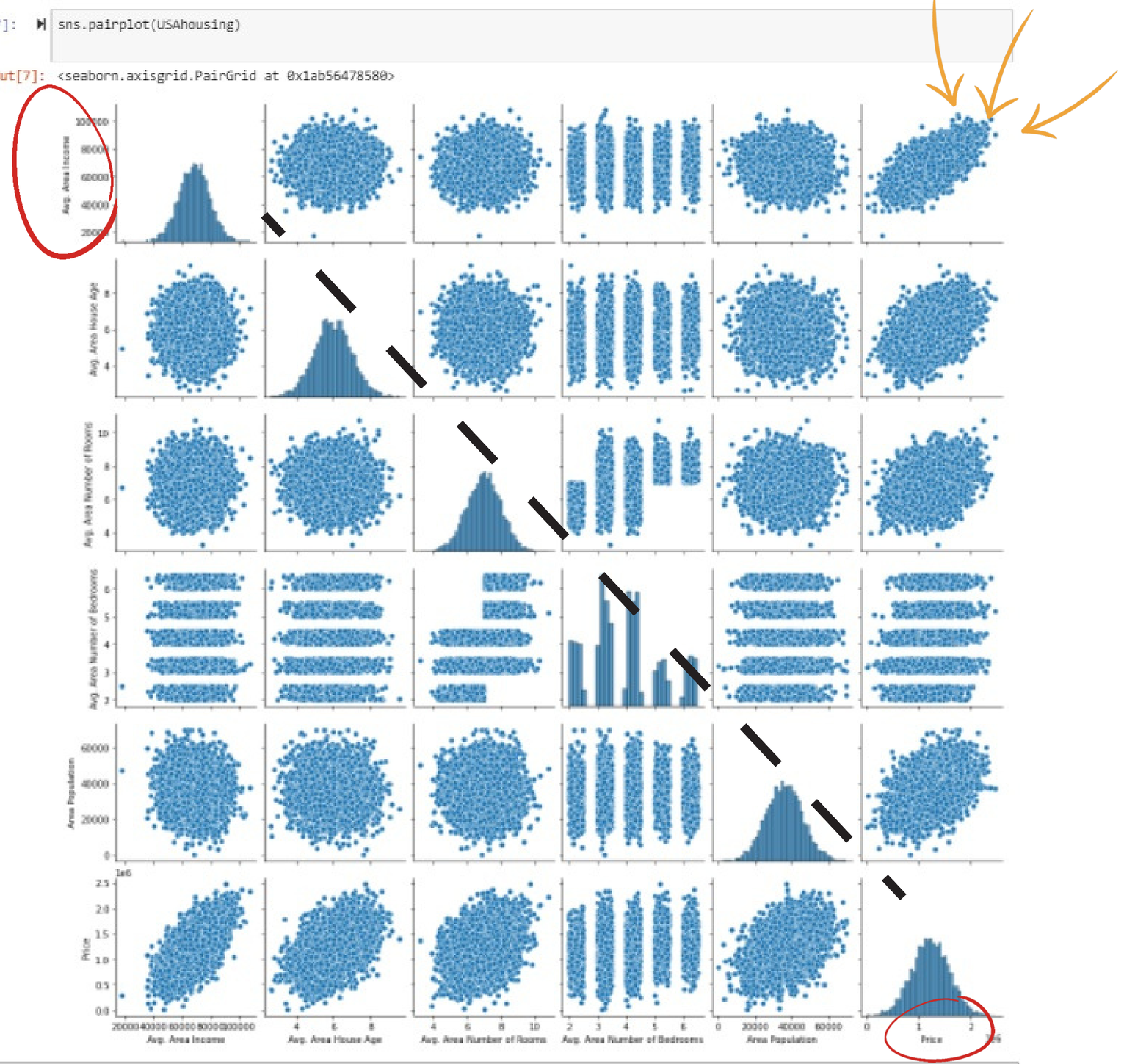
```
USAhousing['Price'].std()
```

```
353117.6265810608
```

Exploratory data analysis (EDA)

```
In [7]: sns.pairplot(USAhousing)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x1ab56478580>
```



Supervised-Learning

یادگیری با نظارت

یادگیری با نظارت یکی از عمومی ترین روش های ماشین لرنینگ یا همان یادگیری ماشین است که در آن به دنبال یافتن رابطه بین ورودیهای سیستم و خروجی سیستم (هدف سیستم) هستیم.

رگرسیون خطی Linear Regression یک مدل یادگیری با نظارت میباشد. در یادگیری با نظارت مدل با ورودیها و خروجیهای مرتبط با ورودیها آموزش میبیند.

مدل پس از آموزش دیدن قابلیت پیش بینی خروجی یک ورودی جدید را خواهد داشت.

$$y = b_0 + b_1*x_1 + b_2*x_2 + \dots + b_n*x_n$$

Define X & y

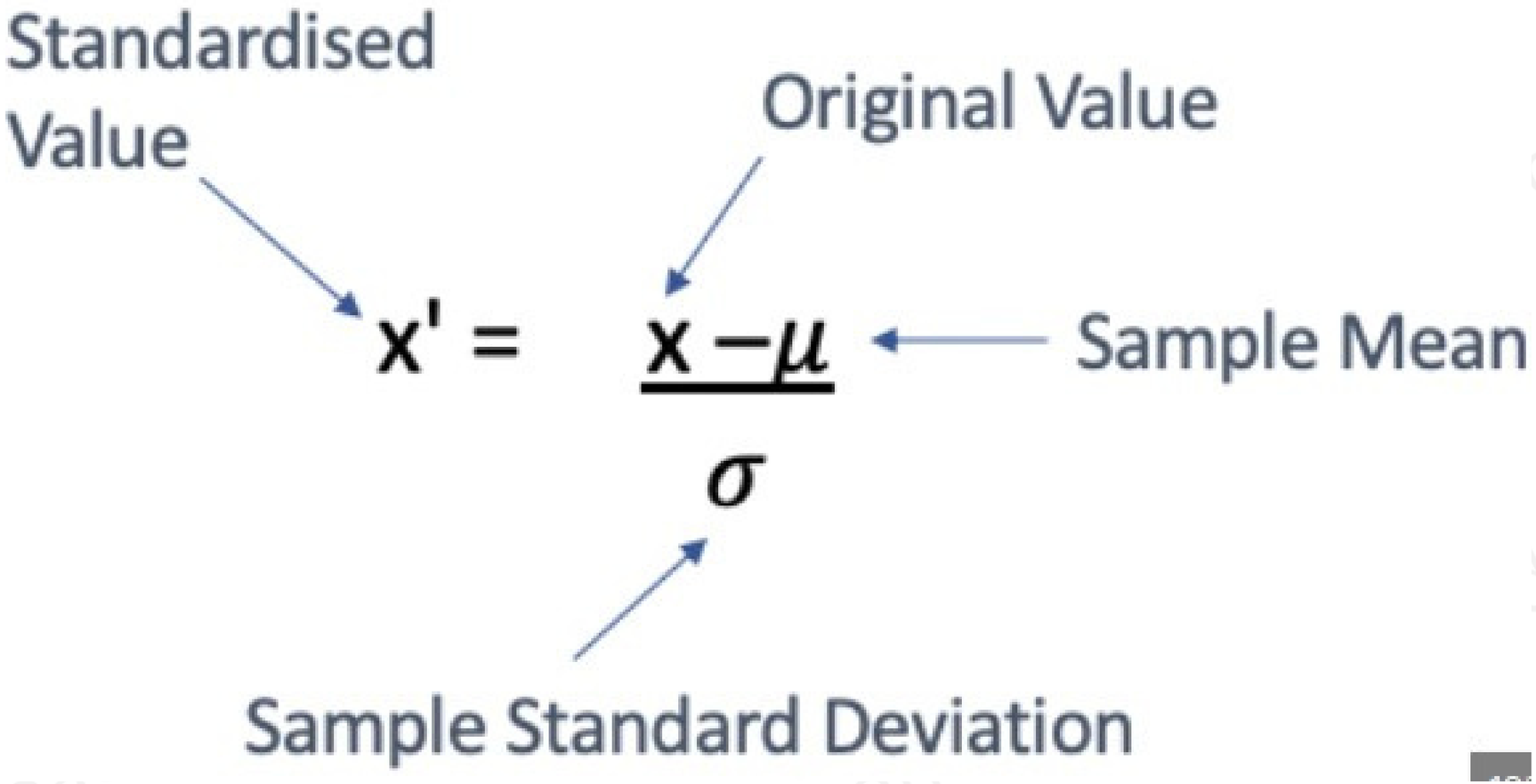
```
[]: X = USAhousing.iloc[:, :5]
    y = USAhousing.iloc[:,-2]
```

X

y

	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\nDanielstown, 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

Standardize Formula



Standardize in Scikit learn, fit_transform()

```
# Preprocessing allows us to standarsize our data
from sklearn import preprocessing
```

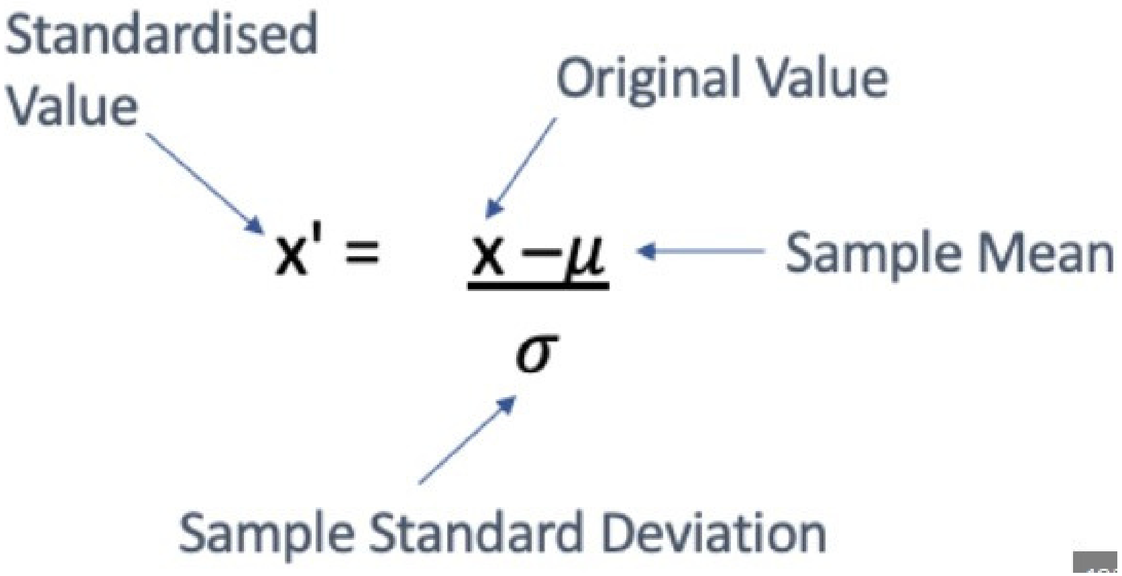
```
transform = preprocessing.StandardScaler()
x_scaled = transform.fit_transform(X)
x_scaled
```

```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
        -2.15665546e-01, -1.85695338e-01, -1.05999788e-01],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -2.15665546e-01, -1.85695338e-01, -1.05999788e-01],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
        -2.15665546e-01, -1.85695338e-01, -1.05999788e-01],
       ...,
       [ 1.63592675e+00,  1.99100483e+00,  3.49060516e+00, ...,
        -2.15665546e-01, -1.85695338e-01, -1.05999788e-01],
```

Standardize in Scikit learn, fit_transform()

.fit()

.transform()



Standardize before or after train test split??

Before: Model receives values in the same range for both training and testing

After: Avoid data leaking

Train data- Test data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state = 101)
```

.fit(), .transform(), .fit_transform()

```
In [15]: from sklearn.preprocessing import StandardScaler

In [16]: sc = StandardScaler()

In [17]: sc.fit(X_train)

Out[17]: StandardScaler
         StandardScaler()

In [18]: X_train_scaled=sc.transform(X_train)

In [19]: X_train_scaled1 = sc.fit_transform(X_train)
```

X_train

In [20]:
```
X_train_scaled
```

Out[20]:
```
array([[-0.03006744, -0.62464758,  0.51983299, -0.70752587,  0.84401035],
       [ 0.69237821, -0.40250734,  0.66044973,  0.19342597, -0.61479079],
       [ 0.23420757,  0.39631758,  0.26603627,  1.17554465,  0.24991721],
       ...,
       [-1.20939274, -2.29617876, -0.11754806, -1.35686054,  2.71014286],
       [-0.50737839, -0.03842068, -1.07647137,  0.10414246, -0.34136517],
       [ 0.63081009,  1.71652039,  1.46161649,  0.29082617,  2.00523872]])
```

In [21]:
```
X_train_scaled1
```

Out[21]:
```
array([[-0.03006744, -0.62464758,  0.51983299, -0.70752587,  0.84401035],
       [ 0.69237821, -0.40250734,  0.66044973,  0.19342597, -0.61479079],
       [ 0.23420757,  0.39631758,  0.26603627,  1.17554465,  0.24991721],
       ...,
       [-1.20939274, -2.29617876, -0.11754806, -1.35686054,  2.71014286],
       [-0.50737839, -0.03842068, -1.07647137,  0.10414246, -0.34136517],
       [ 0.63081009,  1.71652039,  1.46161649,  0.29082617,  2.00523872]])
```

In [22]:
```
col=X.columns
X_train_scaled = pd.DataFrame(X_train_scaled, columns=col)
X_train_scaled
```

Out[22]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	-0.030067	-0.624648	0.519833	-0.707526	0.844010
1	0.692378	-0.402507	0.660450	0.193426	-0.614791
2	0.234208	0.396318	0.266036	1.175545	0.249917
3	0.470747	-1.033067	0.934095	0.266476	0.228497

X_test

```
X_test
```

3]:

	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
...	...	...	...	...	...
1776	65173.050438	7.679469	6.602618	4.23	44125.540782
4269	42969.659393	6.295501	7.885507	4.38	29594.089863
1661	48735.924512	5.543730	6.091906	2.43	19682.347295
2410	65081.584048	5.433570	9.212518	5.14	37594.493458
2302	65969.707036	7.325976	8.020966	4.09	61772.756810

2000 rows × 5 columns

```
X_test_scaled=sc.transform(X_test)
```

```
col=X.columns
X_test_scaled = pd.DataFrame(X_test_scaled, columns=col)
X_test_scaled
```

5]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	-0.154550	-0.262335	0.813343	0.282709	0.065749
1	-0.588708	-1.074748	0.444244	1.824879	-1.014292
2	0.495021	0.815267	1.357532	-0.512725	0.703229
3	-0.614765	0.262448	-0.393878	0.258359	-0.719191
4	0.428782	-1.217991	0.335979	1.979096	-1.177630

X_train

X_test

```
X_train= X_train_scaled
X_test= X_test_scaled
```

Linear Regression Model

```
from sklearn.linear_model import LinearRegression
```

```
lm = LinearRegression()
```

```
lm.fit(X_train,y_train)
```

1]:

```
LinearRegression
LinearRegression()
```

Linear regression model coefficients

Multiple Linear Regression

$$y = b_0 + b_1*x_1 + b_2*x_2 + \ldots + b_n*x_n$$

Price: y_train

```
# print the intercept(b0)
print(lm.intercept_)
```

```
1228450.318381689
```

```
# print the coefficients(b1, b2, ...)
print(lm.coef_)
```

```
[227623.46356492 160615.53988471 121847.23968262   2752.11164653
 151227.42362   ]
```

Prediction from the model

$$y = b_0 + b_1*x_1 + b_2*x_2 + \ldots + b_n*x_n$$

```
predictions = lm.predict(X_test)
```

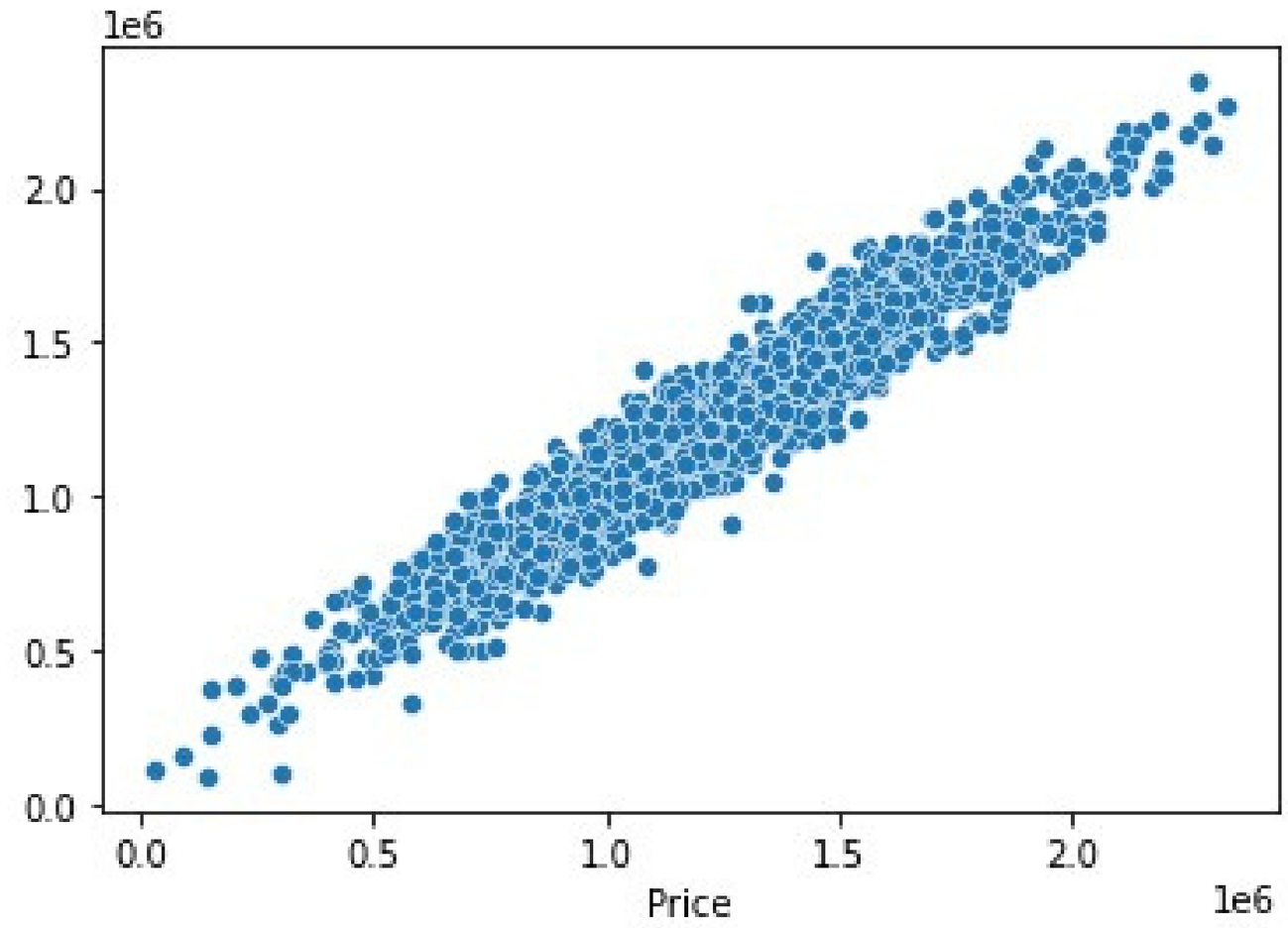
```
predictions
```

```
]: array([1260960.70567625,  827588.75560362, 1742421.24254321, ...,
          372191.40626968, 1365217.15140893, 1914519.54178798])
```

y_test vs predictions

```
sns.scatterplot(x=y_test, y=predictions)
```

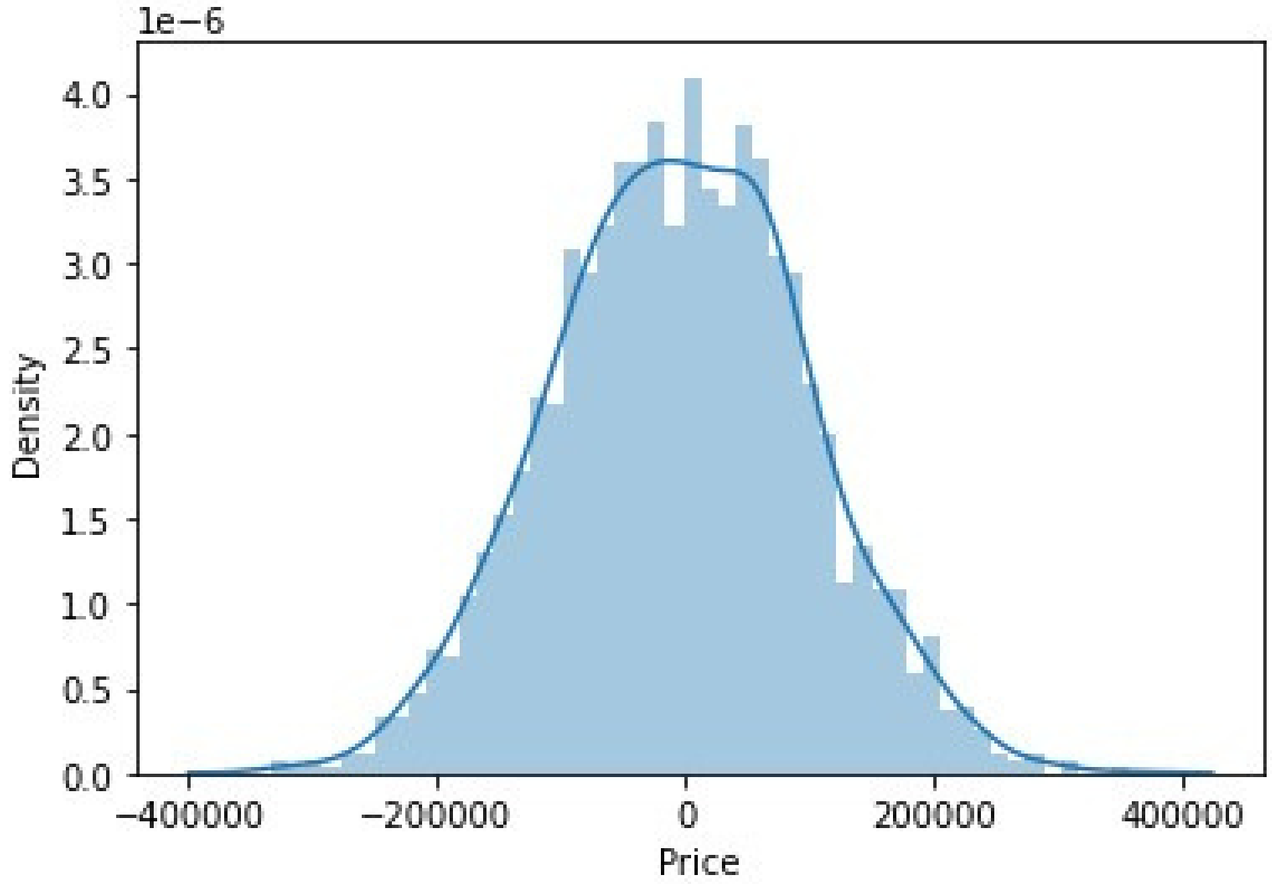
```
2]: <AxesSubplot:xlabel='Price'>
```



Residual Histogram

```
[55]: residual= y_test-predictions
```

```
[56]: sns.distplot(residual,bins=50);
```



Regression Evaluation Metrics

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Metrics in sklearn

```
from sklearn import metrics
```

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 82288.22251914964
MSE: 10460958907.209742
RMSE: 102278.8292229127
```

Dump & Load the Model

```
!pip install pickle
```

```
import pickle
```

```
pickle.dump(lm, open('housing.pkl','wb'))
```

```
model_pk=pickle.load(open('housing.pkl','rb'))
```

Single Sample Prediction

```
my_sample={'Avg. Area Income': 79500, 'Avg. Area House Age': 7, 'Avg. Area Number of Rooms':5,
           'Avg. Area Number of Bedrooms': 3, 'Area Population': 23000}
my_sample=pd.DataFrame([my_sample])
```

```
my_sample_scaled = sc.transform(my_sample)
my_sample = my_sample_scaled
```

```
model_pk.predict(my_sample)
```

```
array([1192525.53724609])
```

Assignment:

تمرین:

کدهای ارائه شده در درس را بررسی و اجرا کنید.

نوتبوک Single_sample_prediction را برای مقادیر مختلف بررسی کنید.