In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('USA_Housing.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addre
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry A 674\nLaurabury, 370
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Vie Suite 079\nLa 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 448
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nF AE 093

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

Out[4]:

	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 [5]:
```
df.shape
```

Out[5]:
```
(5000, 7)
```

In [6]:
```
df.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]:
```
df.isna().sum()
```

Out[7]:
```
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 [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
x = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms'
y = df['Price']
```

In [10]:
```
from sklearn.linear_model import LinearRegression
```

In [11]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3, random_
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
```

In [13]:
```
# scaler = StandardScaler()
# x_train_scale = scaler.fit_transform(x_train)
# x_test_scale = scaler.transform(x_test)
```

In [14]:
```
lr = LinearRegression()
```

In [15]:
```
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)
```

In [16]:
```
from sklearn.metrics import r2_score
```

In [17]:
```
r2 = r2_score(y_test,y_pred)
r2
```

Out[17]:
```
0.9146818498754012
```

In [18]:
```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 81135.56609336908
MSE: 10068422551.400915
RMSE: 100341.52954485454
```

In [ ]:
```

```