

In [1]:

```
import pandas as pd
import numpy as np
import random as rnd

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam

from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score
from sklearn.metrics import classification_report, confusion_matrix
```

In [2]:

```
df = pd.read_csv('/content/House Price India.csv')
```

In [3]:

```
print(df.columns.values)
```

```
['id' 'Date' 'number of bedrooms' 'number of bathrooms' 'living area'
 'lot area' 'number of floors' 'waterfront present' 'number of views'
 'condition of the house' 'grade of the house'
 'Area of the house(excluding basement)' 'Area of the basement'
 'Built Year' 'Renovation Year' 'Postal Code' 'Latitude' 'Longitude'
 'living_area_renov' 'lot_area_renov' 'Number of schools nearby'
 'Distance from the airport' 'Price']
```

In [4]:

```
df.head()
```

Out[4]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code
0	6762810145	42491	5	2.50	3650	9050	2.0	0	4	5	...	1921	0	122
1	6762810635	42491	4	2.50	2920	4000	1.5	0	0	5	...	1909	0	122
2	6762810998	42491	5	2.75	2910	9480	1.5	0	0	3	...	1939	0	122
3	6762812605	42491	4	2.50	3310	42998	2.0	0	0	3	...	2001	0	122
4	6762812919	42491	3	2.00	2710	4500	1.5	0	0	4	...	1929	0	122

5 rows x 23 columns



In [5]:

```
df.tail()
```

Out[5]:

	id	Date	number of bedrooms	number of bathrooms	living area	lot area	number of floors	waterfront present	number of views	condition of the house	...	Built Year	Renovation Year	Postal Code
--	----	------	-----------------------	------------------------	----------------	-------------	------------------------	-----------------------	-----------------------	------------------------------	-----	---------------	--------------------	----------------

			bedrooms	bathrooms	area	area	number of floors	waterfront present	number of views	condition of the house		Built Year	Renovation Year
14615	6762830250	42734	3	1.5	1556	20000	1.0	0	0	4 ...	1957	0	
14616	6762830339	42734	3	2.0	1680	7000	1.5	0	0	4 ...	1968	0	
14617	6762830618	42734	2	1.0	1070	6120	1.0	0	0	3 ...	1962	0	
14618	6762830709	42734	4	1.0	1030	6621	1.0	0	0	4 ...	1955	0	
14619	6762831463	42734	3	1.0	900	4770	1.0	0	0	3 ...	1969	2009	

5 rows x 23 columns



In [6]:

```
df.isnull().sum()
```

Out[6]:

```
id          0
Date        0
number of bedrooms      0
number of bathrooms     0
living area      0
lot area         0
number of floors        0
waterfront present      0
number of views         0
condition of the house  0
grade of the house      0
Area of the house(excluding basement)  0
Area of the basement    0
Built Year            0
Renovation Year        0
Postal Code           0
Lattitude             0
Longitude             0
living_area_renov      0
lot_area_renov         0
Number of schools nearby  0
Distance from the airport  0
Price                0
dtype: int64
```

In [7]:

```
dataset = df.values
X = dataset[:,0:23]
```

In [8]:

```
X
```

Out[8]:

```
array([[6.76281014e+09, 4.24910000e+04, 5.00000000e+00, ...,
        2.00000000e+00, 5.80000000e+01, 2.38000000e+06],
       [6.76281064e+09, 4.24910000e+04, 4.00000000e+00, ...,
        2.00000000e+00, 5.10000000e+01, 1.40000000e+06],
       [6.76281100e+09, 4.24910000e+04, 5.00000000e+00, ...,
        1.00000000e+00, 5.30000000e+01, 1.20000000e+06],
       ...,
       [6.76283062e+09, 4.27340000e+04, 2.00000000e+00, ...,
        2.00000000e+00, 6.40000000e+01, 2.09000000e+05],
       [6.76283071e+09, 4.27340000e+04, 4.00000000e+00, ...,
        3.00000000e+00, 5.40000000e+01, 2.05000000e+05],
       [6.76283146e+09, 4.27340000e+04, 3.00000000e+00, ...,
        2.00000000e+00, 5.50000000e+01, 1.46000000e+05]])
```

In [9]:

```
Y = dataset[:, 22]
```

```
In [10]:
```

```
Y
```

```
Out[10]:
```

```
array([2380000., 1400000., 1200000., ..., 209000., 205000., 146000.])
```

```
In [11]:
```

```
min_max_scaler = MinMaxScaler()  
X_scale = min_max_scaler.fit_transform(X)
```

```
In [12]:
```

```
X_scale
```

```
Out[12]:
```

```
array([[0.00578811, 0.          , 0.125      , ..., 0.5          , 0.26666667,  
        0.30202047],  
       [0.0284775 , 0.          , 0.09375    , ..., 0.5          , 0.03333333,  
        0.17344529],  
       [0.04528616, 0.          , 0.125      , ..., 0.          , 0.1          ,  
        0.14720546],  
       ...,  
       [0.95378774, 1.          , 0.03125    , ..., 0.5          , 0.46666667,  
        0.01718709],  
       [0.95800148, 1.          , 0.09375    , ..., 1.          , 0.13333333,  
        0.01666229],  
       [0.99291535, 1.          , 0.0625     , ..., 0.5          , 0.16666667,  
        0.00892154]])
```

```
In [13]:
```

```
X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)
```

```
In [14]:
```

```
X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)
```

```
In [15]:
```

```
print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)
```

```
(10234, 23) (2193, 23) (2193, 23) (10234,) (2193,) (2193,)
```

```
In [16]:
```

```
model = Sequential([  
    Dense(32, activation='relu', input_shape=(23,)),  
    Dense(32, activation='relu'),  
    Dense(1, activation='sigmoid'),  
])
```

```
In [17]:
```

```
model.compile(optimizer='sgd',  
              loss='binary_crossentropy',  
              metrics=['accuracy'])
```

```
In [18]:
```

```
hist = model.fit(X_train, Y_train,  
                 batch_size=32, epochs=100,  
                 validation_data=(X_val, Y_val))
```

[illegible]

[illegible]

[illegible]

[illegible]

```
Epoch 97/100
320/320 [=====] - 1s 2ms/step - loss: nan - accuracy: 0.0000e+00
- val_loss: nan - val_accuracy: 0.0000e+00
Epoch 98/100
320/320 [=====] - 1s 2ms/step - loss: nan - accuracy: 0.0000e+00
- val_loss: nan - val_accuracy: 0.0000e+00
Epoch 99/100
320/320 [=====] - 1s 2ms/step - loss: nan - accuracy: 0.0000e+00
- val_loss: nan - val_accuracy: 0.0000e+00
Epoch 100/100
320/320 [=====] - 1s 2ms/step - loss: nan - accuracy: 0.0000e+00
- val_loss: nan - val_accuracy: 0.0000e+00
```

```
model.evaluate(X_test, Y_test)[1]
```

Out[19]:

In [20]:

```
a[:10]
```