

AI ASSIGNMENT 1

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import Sequential
from sklearn.metrics import r2_score
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
```

/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

Reading the dataset

In [2]:

```
df = pd.read_csv('/kaggle/input/house-price-dataset-of-india/House Price India.csv')
df.drop(['id', 'Date'], axis=1, inplace=True)
df.head()
```

Out[2]:

	number of bedrooms	number of bathrooms	living area	lot area	number of floors	water front present	number of views	condition of the house	grade of the house	Area of the house(excluding base ment)		Build Year	Renovation Year	Postal Code	Latitude	Longitude	living_area_renov	lot_area_renov	Number of schools nearby	Distance from the airport	Price
0	5	2.50	3650	9050	2.0	0	4	5	10	3370	.	1921	0	122003	52.8645	-114.557	2880	5400	2	58	238000

	nu m b e r o f b e d r o o m s	nu m b e r o f b a t h r o o m s	li v i n g a r e a	l o t a r e a	n u m b e r o f f l o o r s	wa t e r f r o n t p r e s e n t	n u m b e r o f v i e w s	co n d i t i o n o f t h e h o u s e	g r a d e o f t h e h o u s e	A r e a o f t h e h o u s e (e x c l u d i n g b a s e m e n t)	.	.	B u i l t Y e a r	R e n o v a t i o n Y e a r	P o s t a l C o d e	L a t t i t u d e	L o n g i t u d e	l i v i n g _ a r e a _ r e n o v	l o t _ a r e a _ r e n o v	N u m b e r o f s c h o l s n e a r b y	D i s t a n c e f r o m t h e a i r p o r t	P r i c e
1	4	2.50	2920	4000	1.5	0	0	5	8	1910	.	.	1909	0	122004	52.8878	-114.470	2470	4000	2	51	140000
2	5	2.75	2910	9480	1.5	0	0	3	8	2910	.	.	1939	0	122004	52.8852	-114.468	2940	6600	1	53	120000
3	4	2.50	3310	42998	2.0	0	0	3	9	3310	.	.	2001	0	122005	52.9532	-114.321	3350	42847	3	76	838000
4	3	2.00	2710	4500	1.5	0	0	4	8	1880	.	.	1929	0	122006	52.9047	-114.485	2060	4500	1	51	805000

5 rows × 21 columns

The inference from the dataset is as follows,

- There are 21 features in the given dataset.
- The price column is the only column which depend on the rest of the 20 columns.

Getting the info of the dataframe

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 14620 entries, 0 to 14619
```

```
Data columns (total 21 columns):
```

#	Column	Non-Null Count	Dtype
0	number of bedrooms	14620 non-null	int64
1	number of bathrooms	14620 non-null	float64
2	living area	14620 non-null	int64
3	lot area	14620 non-null	int64
4	number of floors	14620 non-null	float64
5	waterfront present	14620 non-null	int64
6	number of views	14620 non-null	int64
7	condition of the house	14620 non-null	int64
8	grade of the house	14620 non-null	int64
9	Area of the house(excluding basement)	14620 non-null	int64
10	Area of the basement	14620 non-null	int64
11	Built Year	14620 non-null	int64
12	Renovation Year	14620 non-null	int64
13	Postal Code	14620 non-null	int64
14	Lattitude	14620 non-null	float64
15	Longitude	14620 non-null	float64
16	living_area_renov	14620 non-null	int64
17	lot_area_renov	14620 non-null	int64
18	Number of schools nearby	14620 non-null	int64
19	Distance from the airport	14620 non-null	int64
20	Price	14620 non-null	int64

```
dtypes: float64(4), int64(17)
```

```
memory usage: 2.3 MB
```

The inference from the above cell are discussed below,

- Only four columns "number of bathrooms", "number of floors", "Lattitude" and "Longitude" are float datatype.
- Rest of the columns have only int64 datatype.

Checking the NaN values from the dataframe

In [4]:

```
df.isna().sum()
```

Out[4]:

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
Latitude                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

```

From the above cell it is founded that,

- There are no NaN values in the given dataset

Splitting independent and dependent variables as X and y

```

In [5]:
X = df.drop('Price', axis=1)
y = df['Price']

```

Splitting the train test separately

```

In [6]:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12)

```

Getting the shape of X, y train and test

```

In [7]:
print(f'\n shape of X_train - {X_train.shape}\n')
print(f' shape of X_test - {X_test.shape}\n')
print(f' shape of y_train - {y_train.shape}\n')
print(f' shape of y_test - {y_test.shape}\n')
shape of X_train - (11696, 20)

shape of X_test - (2924, 20)

shape of y_train - (11696,)

shape of y_test - (2924,)

```

Model building

In [8]:

```
number_of_features = len(X.columns)

model = Sequential()

model.add(layer=Input(shape=number_of_features))

model.add(layer=Dense(units=32, activation='relu'))

model.add(layer=Dense(units=64, activation='relu'))

model.add(layer=Dense(units=128, activation='relu'))

model.add(layer=Dense(units=256, activation='relu'))

model.add(layer=Dense(units=512, activation='relu'))

model.add(layer=Dense(units=1024, activation='relu'))

model.add(layer=Dense(units=2048, activation='relu'))

model.add(layer=Dense(units=256, activation='relu'))

model.add(layer=Dense(units=128, activation='relu'))

model.add(layer=Dense(units=64, activation='relu'))

model.add(layer=Dense(units=32, activation='relu'))

model.add(layer=Dense(units=16, activation='relu'))

model.add(layer=Dense(units=1, activation='linear'))

model.summary()

Model: "sequential"
```

Layer (type)	Output Shape	Param #
=====	=====	=====
dense (Dense)	(None, 32)	672
dense_1 (Dense)	(None, 64)	2112
dense_2 (Dense)	(None, 128)	8320
dense_3 (Dense)	(None, 256)	33024
dense_4 (Dense)	(None, 512)	131584
dense_5 (Dense)	(None, 1024)	525312
dense_6 (Dense)	(None, 2048)	2099200

dense_7 (Dense)	(None, 256)	524544
dense_8 (Dense)	(None, 128)	32896
dense_9 (Dense)	(None, 64)	8256
dense_10 (Dense)	(None, 32)	2080
dense_11 (Dense)	(None, 16)	528
dense_12 (Dense)	(None, 1)	17

```
=====
Total params: 3,368,545
Trainable params: 3,368,545
Non-trainable params: 0
```

Compiling the model

```
In [9]:
model.compile(optimizer='adam', loss='mse', metrics=['mae', 'mape'])
```

Model Training

```
In [10]:
history = model.fit(X_train, y_train, epochs=100)

Epoch 1/100
366/366 [=====] - 11s 6ms/step - loss: 115373219840.0000 -
- mae: 223078.1875 - mape: 49.2277
Epoch 2/100
366/366 [=====] - 2s 5ms/step - loss: 69455994880.0000 -
- mae: 175933.2812 - mape: 37.2717
Epoch 3/100
366/366 [=====] - 2s 6ms/step - loss: 63709241344.0000 -
- mae: 169027.0156 - mape: 35.5546
Epoch 4/100
366/366 [=====] - 2s 6ms/step - loss: 61131841536.0000 -
- mae: 165759.4219 - mape: 34.9124
Epoch 5/100
366/366 [=====] - 2s 5ms/step - loss: 60745990144.0000 -
- mae: 167357.0938 - mape: 35.0202
Epoch 6/100
366/366 [=====] - 2s 5ms/step - loss: 63253823488.0000 -
- mae: 168348.6562 - mape: 35.1488
Epoch 7/100
366/366 [=====] - 2s 6ms/step - loss: 59147300864.0000 -
- mae: 164895.6562 - mape: 34.7841
Epoch 8/100
366/366 [=====] - 2s 6ms/step - loss: 63616413696.0000 -
- mae: 169971.0000 - mape: 35.2147
Epoch 9/100
366/366 [=====] - 2s 6ms/step - loss: 56888909824.0000 -
- mae: 161452.5312 - mape: 33.8793
```

Epoch 10/100
366/366 [=====] - 2s 6ms/step - loss: 56831733760.0000 -
mae: 160004.9688 - mape: 33.5284
Epoch 11/100
366/366 [=====] - 2s 6ms/step - loss: 59214237696.0000 -
mae: 162406.9219 - mape: 34.1977
Epoch 12/100
366/366 [=====] - 2s 6ms/step - loss: 59529773056.0000 -
mae: 163702.0625 - mape: 34.1877
Epoch 13/100
366/366 [=====] - 2s 7ms/step - loss: 58980405248.0000 -
mae: 161938.1562 - mape: 33.6812
Epoch 14/100
366/366 [=====] - 2s 6ms/step - loss: 53838446592.0000 -
mae: 155908.4219 - mape: 32.6511
Epoch 15/100
366/366 [=====] - 2s 6ms/step - loss: 54630879232.0000 -
mae: 156266.7656 - mape: 32.6601
Epoch 16/100
366/366 [=====] - 2s 6ms/step - loss: 55835422720.0000 -
mae: 158658.1875 - mape: 33.1131
Epoch 17/100
366/366 [=====] - 2s 6ms/step - loss: 54464913408.0000 -
mae: 157176.4844 - mape: 32.8886
Epoch 18/100
366/366 [=====] - 2s 6ms/step - loss: 56984547328.0000 -
mae: 161383.6719 - mape: 33.6896
Epoch 19/100
366/366 [=====] - 2s 6ms/step - loss: 54241468416.0000 -
mae: 155621.0000 - mape: 32.5170
Epoch 20/100
366/366 [=====] - 2s 6ms/step - loss: 59135946752.0000 -
mae: 163379.9688 - mape: 34.0004
Epoch 21/100
366/366 [=====] - 2s 6ms/step - loss: 52663656448.0000 -
mae: 154168.8594 - mape: 32.1560
Epoch 22/100
366/366 [=====] - 2s 6ms/step - loss: 55579299840.0000 -
mae: 157360.4844 - mape: 32.8772
Epoch 23/100
366/366 [=====] - 2s 6ms/step - loss: 54586580992.0000 -
mae: 156370.3594 - mape: 32.5105
Epoch 24/100
366/366 [=====] - 2s 6ms/step - loss: 53262577664.0000 -
mae: 155050.3906 - mape: 32.3967
Epoch 25/100
366/366 [=====] - 2s 6ms/step - loss: 54075871232.0000 -
mae: 157037.7812 - mape: 32.5757
Epoch 26/100
366/366 [=====] - 2s 6ms/step - loss: 51292758016.0000 -
mae: 151864.7812 - mape: 31.6146
Epoch 27/100

366/366 [=====] - 2s 6ms/step - loss: 55897804800.0000 -
mae: 157001.2969 - mape: 32.6494
Epoch 28/100
366/366 [=====] - 2s 7ms/step - loss: 52807000064.0000 -
mae: 154402.5312 - mape: 32.1374
Epoch 29/100
366/366 [=====] - 2s 5ms/step - loss: 50559614976.0000 -
mae: 150681.7656 - mape: 31.3239
Epoch 30/100
366/366 [=====] - 2s 6ms/step - loss: 54030680064.0000 -
mae: 155396.6719 - mape: 32.1251
Epoch 31/100
366/366 [=====] - 2s 6ms/step - loss: 52283559936.0000 -
mae: 153384.2031 - mape: 32.0258
Epoch 32/100
366/366 [=====] - 2s 6ms/step - loss: 52832710656.0000 -
mae: 154543.4375 - mape: 31.9707
Epoch 33/100
366/366 [=====] - 2s 6ms/step - loss: 52489314304.0000 -
mae: 153925.0781 - mape: 31.7906
Epoch 34/100
366/366 [=====] - 2s 6ms/step - loss: 59632648192.0000 -
mae: 161197.2188 - mape: 33.5867
Epoch 35/100
366/366 [=====] - 2s 5ms/step - loss: 52995551232.0000 -
mae: 152778.0312 - mape: 31.5439
Epoch 36/100
366/366 [=====] - 2s 6ms/step - loss: 51760218112.0000 -
mae: 152372.9219 - mape: 31.5417
Epoch 37/100
366/366 [=====] - 2s 6ms/step - loss: 51361501184.0000 -
mae: 152362.2969 - mape: 31.6092
Epoch 38/100
366/366 [=====] - 2s 6ms/step - loss: 53745012736.0000 -
mae: 153568.5156 - mape: 31.8070
Epoch 39/100
366/366 [=====] - 2s 6ms/step - loss: 52044754944.0000 -
mae: 152404.7344 - mape: 31.6283
Epoch 40/100
366/366 [=====] - 2s 6ms/step - loss: 52324454400.0000 -
mae: 151717.5000 - mape: 31.5201
Epoch 41/100
366/366 [=====] - 2s 6ms/step - loss: 51012214784.0000 -
mae: 150878.4219 - mape: 31.4002
Epoch 42/100
366/366 [=====] - 2s 6ms/step - loss: 51013070848.0000 -
mae: 151318.3750 - mape: 31.3557
Epoch 43/100
366/366 [=====] - 3s 7ms/step - loss: 52125188096.0000 -
mae: 151897.7812 - mape: 31.5532
Epoch 44/100

366/366 [=====] - 2s 6ms/step - loss: 49911029760.0000 -
mae: 149106.8281 - mape: 30.9280
Epoch 45/100
366/366 [=====] - 2s 6ms/step - loss: 51189428224.0000 -
mae: 152566.5625 - mape: 31.6698
Epoch 46/100
366/366 [=====] - 2s 6ms/step - loss: 50706468864.0000 -
mae: 149443.4844 - mape: 30.8647
Epoch 47/100
366/366 [=====] - 2s 6ms/step - loss: 51435798528.0000 -
mae: 151232.4531 - mape: 31.4542
Epoch 48/100
366/366 [=====] - 2s 6ms/step - loss: 48945655808.0000 -
mae: 148143.1250 - mape: 30.8189
Epoch 49/100
366/366 [=====] - 2s 6ms/step - loss: 53101506560.0000 -
mae: 153325.5469 - mape: 31.6500
Epoch 50/100
366/366 [=====] - 2s 6ms/step - loss: 52147609600.0000 -
mae: 151818.1719 - mape: 31.4515
Epoch 51/100
366/366 [=====] - 2s 6ms/step - loss: 50240663552.0000 -
mae: 150436.9844 - mape: 31.1419
Epoch 52/100
366/366 [=====] - 2s 6ms/step - loss: 49564909568.0000 -
mae: 148995.1562 - mape: 31.0548
Epoch 53/100
366/366 [=====] - 2s 6ms/step - loss: 51030122496.0000 -
mae: 151829.7031 - mape: 31.3057
Epoch 54/100
366/366 [=====] - 2s 6ms/step - loss: 50005434368.0000 -
mae: 149201.5156 - mape: 31.0676
Epoch 55/100
366/366 [=====] - 2s 6ms/step - loss: 48938049536.0000 -
mae: 146918.7188 - mape: 30.4790
Epoch 56/100
366/366 [=====] - 2s 6ms/step - loss: 49289723904.0000 -
mae: 148141.9688 - mape: 30.6173
Epoch 57/100
366/366 [=====] - 2s 6ms/step - loss: 49386569728.0000 -
mae: 148052.7969 - mape: 30.6916
Epoch 58/100
366/366 [=====] - 2s 7ms/step - loss: 49865011200.0000 -
mae: 148512.9375 - mape: 30.8196
Epoch 59/100
366/366 [=====] - 2s 6ms/step - loss: 48806412288.0000 -
mae: 147634.5938 - mape: 30.6775
Epoch 60/100
366/366 [=====] - 2s 6ms/step - loss: 48990040064.0000 -
mae: 147891.5469 - mape: 30.9335
Epoch 61/100

366/366 [=====] - 2s 6ms/step - loss: 49532538880.0000 -
mae: 148435.2031 - mape: 30.7960
Epoch 62/100
366/366 [=====] - 2s 6ms/step - loss: 49416626176.0000 -
mae: 147605.6875 - mape: 30.6762
Epoch 63/100
366/366 [=====] - 2s 6ms/step - loss: 48503390208.0000 -
mae: 147891.4375 - mape: 30.7656
Epoch 64/100
366/366 [=====] - 2s 6ms/step - loss: 48167800832.0000 -
mae: 145974.1562 - mape: 30.3071
Epoch 65/100
366/366 [=====] - 2s 6ms/step - loss: 48472039424.0000 -
mae: 146392.1562 - mape: 30.4809
Epoch 66/100
366/366 [=====] - 2s 6ms/step - loss: 47465766912.0000 -
mae: 145686.1250 - mape: 30.3407
Epoch 67/100
366/366 [=====] - 2s 6ms/step - loss: 48080560128.0000 -
mae: 145239.3281 - mape: 30.2406
Epoch 68/100
366/366 [=====] - 2s 6ms/step - loss: 47816654848.0000 -
mae: 145906.5781 - mape: 30.3515
Epoch 69/100
366/366 [=====] - 2s 6ms/step - loss: 48326881280.0000 -
mae: 147244.9219 - mape: 30.5693
Epoch 70/100
366/366 [=====] - 2s 6ms/step - loss: 48167899136.0000 -
mae: 145695.2812 - mape: 30.3434
Epoch 71/100
366/366 [=====] - 2s 6ms/step - loss: 46709411840.0000 -
mae: 144538.7656 - mape: 30.0584
Epoch 72/100
366/366 [=====] - 2s 5ms/step - loss: 47881138176.0000 -
mae: 145494.6250 - mape: 30.2558
Epoch 73/100
366/366 [=====] - 2s 6ms/step - loss: 47831670784.0000 -
mae: 146193.8750 - mape: 30.4547
Epoch 74/100
366/366 [=====] - 2s 6ms/step - loss: 48283926528.0000 -
mae: 146402.7344 - mape: 30.5587
Epoch 75/100
366/366 [=====] - 2s 6ms/step - loss: 49335169024.0000 -
mae: 147990.4531 - mape: 30.8068
Epoch 76/100
366/366 [=====] - 2s 6ms/step - loss: 46650712064.0000 -
mae: 144099.5156 - mape: 30.0486
Epoch 77/100
366/366 [=====] - 2s 6ms/step - loss: 48079314944.0000 -
mae: 145554.9688 - mape: 30.2738
Epoch 78/100

366/366 [=====] - 2s 5ms/step - loss: 46424326144.0000 -
mae: 144024.5938 - mape: 30.0820
Epoch 79/100
366/366 [=====] - 2s 6ms/step - loss: 45502382080.0000 -
mae: 142833.1094 - mape: 29.9501
Epoch 80/100
366/366 [=====] - 2s 6ms/step - loss: 46820442112.0000 -
mae: 144880.7188 - mape: 30.0728
Epoch 81/100
366/366 [=====] - 2s 6ms/step - loss: 49424752640.0000 -
mae: 148025.7500 - mape: 30.7913
Epoch 82/100
366/366 [=====] - 2s 6ms/step - loss: 48581087232.0000 -
mae: 147064.2969 - mape: 30.5191
Epoch 83/100
366/366 [=====] - 2s 6ms/step - loss: 45965008896.0000 -
mae: 143562.7031 - mape: 30.1378
Epoch 84/100
366/366 [=====] - 2s 6ms/step - loss: 47132475392.0000 -
mae: 144201.1406 - mape: 30.0132
Epoch 85/100
366/366 [=====] - 2s 6ms/step - loss: 46012600320.0000 -
mae: 143252.1094 - mape: 29.8453
Epoch 86/100
366/366 [=====] - 2s 6ms/step - loss: 47465938944.0000 -
mae: 145318.7031 - mape: 30.3195
Epoch 87/100
366/366 [=====] - 2s 6ms/step - loss: 46110588928.0000 -
mae: 143106.4375 - mape: 29.8871
Epoch 88/100
366/366 [=====] - 2s 7ms/step - loss: 46045933568.0000 -
mae: 143356.4062 - mape: 29.9401
Epoch 89/100
366/366 [=====] - 2s 6ms/step - loss: 47192424448.0000 -
mae: 144077.8906 - mape: 30.0460
Epoch 90/100
366/366 [=====] - 2s 6ms/step - loss: 46030782464.0000 -
mae: 143928.2812 - mape: 30.1497
Epoch 91/100
366/366 [=====] - 2s 6ms/step - loss: 47286464512.0000 -
mae: 144318.2188 - mape: 30.0864
Epoch 92/100
366/366 [=====] - 2s 6ms/step - loss: 46418128896.0000 -
mae: 144305.5469 - mape: 30.2092
Epoch 93/100
366/366 [=====] - 2s 6ms/step - loss: 45890568192.0000 -
mae: 143217.1562 - mape: 29.9014
Epoch 94/100
366/366 [=====] - 2s 6ms/step - loss: 44947415040.0000 -
mae: 141036.0000 - mape: 29.5444
Epoch 95/100

```

366/366 [=====] - 2s 6ms/step - loss: 44814655488.0000 -
    mae: 142566.5625 - mape: 29.8249
Epoch 96/100
366/366 [=====] - 2s 5ms/step - loss: 45107769344.0000 -
    mae: 141834.5000 - mape: 29.7779
Epoch 97/100
366/366 [=====] - 2s 6ms/step - loss: 44985761792.0000 -
    mae: 142067.0000 - mape: 29.6291
Epoch 98/100
366/366 [=====] - 2s 6ms/step - loss: 44453998592.0000 -
    mae: 141057.8750 - mape: 29.5080
Epoch 99/100
366/366 [=====] - 2s 6ms/step - loss: 45318406144.0000 -
    mae: 142222.6250 - mape: 29.7655
Epoch 100/100
366/366 [=====] - 2s 6ms/step - loss: 45620596736.0000 -
    mae: 143098.0312 - mape: 29.9300

```

Predicting the unseen dataset

```

In [11]:
y_pred = model.predict(X_test)
92/92 [=====] - 0s 2ms/step

```

```

In [12]:
y_pred

Out[12]:
array([[360362.7 ],
       [394695.16],
       [402347.7 ],
       ...,
       [415337.1 ],
       [324866.47],
       [523623.03]], dtype=float32)

```

```

In [13]:
y_test

```

```

Out[13]:
12149    640000
13581    650000
11595    325000
2769     373000
7393     355000
...
7362     497000
11132    400000
142      366750
1405     276000
6184     569000

```

```
Name: Price, Length: 2924, dtype: int64
```

Visual representation of the actual value vs the predicted value

```

In [14]:

```

```
plt.scatter(y_test, y_pred, c=y_pred)
plt.show()
```

```
In [15]:
pd.DataFrame({'Actual Value':y_test.values.flatten(), 'Predicted Value':y_pred.flatte
n()})
```

Out[15]:

	Actual Value	Predicted Value
0	640000	360362.68750
1	650000	394695.15625
2	325000	402347.68750
3	373000	499798.25000
4	355000	514653.06250
...
2919	497000	497558.09375
2920	400000	374115.21875
2921	366750	415337.09375
2922	276000	324866.46875

	Actual Value	Predicted Value
2923	569000	523623.03125

2924 rows × 2 columns

The above cell maps the actual value and the predicted values

```
In [16]:  
r2_score(y_pred=y_pred, y_true=y_test) * 100
```

```
Out[16]:  
66.53464784299932
```

Thus the model is trained with ANN and got an accuracy of 66.5%

```
In [ ]:
```