Lab Assignment 1: Linear regression by using Deep Neural network

Implement Boston housing price prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.

In [24]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
import math

#libraries required to implement Deep Neural Netwrok
import tensorflow as tf
from tensorflow import keras
from keras import metrics
from keras.models import Sequential
from keras.layers import Dropout,Dense
```

In [3]:
```
housing = pd.read_csv("boston_housing.csv")
housing.head(10)
```

Out[3]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	price
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.10	18.9

In [4]:
```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
```

```
 12  lstat    506 non-null    float64
 13  price    506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [5]:
```
housing.corr()
```

Out[5]:

	crim	zn	indus	chas	nox	rm	age	dis	
crim	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625
zn	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311
indus	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595
chas	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007
nox	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611
rm	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209
age	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456
dis	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494
rad	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000
tax	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910
ptratio	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464
black	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444
lstat	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488
price	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381

In [6]:
```
independent = housing.drop('price',axis=1)
independent.head()
```

Out[6]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33

In [7]:
```
dependent = housing['price'].values
print(dependent)
```

```
[24.  21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.  18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.  12.7 14.5 13.2 13.1 13.5 18.9 20.  21.  24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.  16.6 14.4 19.4 19.7 20.5 25.  23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.  22.2 25.  33.  23.5 19.4 22.  17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.  20.8 21.2 20.3 28.  23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.  22.9 25.  20.6 28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8 19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.  20.3 20.5 17.3 18.8 21.4
```

```
15.7 16.2 18.  14.3 19.2 19.6 23.  18.4 15.6 18.1 17.4 17.1 13.3 17.8
14.  14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5 19.6 15.3 19.4
17.  15.6 13.1 41.3 24.3 23.3 27.  50.  50.  50.  22.7 25.  50.  23.8
23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9 37.2 39.8 36.2
37.9 32.5 26.4 29.6 50.  32.  29.8 34.9 37.  30.5 36.4 31.1 29.1 50.
33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.  22.6 24.4 22.5 24.4 20.
21.7 19.3 22.4 28.1 23.7 25.  23.3 28.7 21.5 23.  26.7 21.7 27.5 30.1
44.8 50.  37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.  24.  25.1 31.5
23.7 23.3 22.  20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8
29.6 42.8 21.9 20.9 44.  50.  36.  30.1 33.8 43.1 48.8 31.  36.5 22.8
30.7 50.  43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.  33.2 33.1 29.1 35.1
45.4 35.4 46.  50.  32.2 22.  20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9
21.7 28.6 27.1 20.3 22.5 29.  24.8 22.  26.4 33.1 36.1 28.4 33.4 28.2
22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1 21.  23.8 23.1
20.4 18.5 25.  24.6 23.  22.2 19.3 22.6 19.8 17.1 19.4 22.2 20.7 21.1
19.5 18.5 20.6 19.  18.7 32.7 16.5 23.9 31.2 17.5 17.2 23.1 24.5 26.6
22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.  19.9 20.8 16.8
21.9 27.5 21.9 23.1 50.  50.  50.  50.  50.  13.8 13.8 15.  13.9 13.3
13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2 11.5 15.1 23.2
 9.7 13.8 12.7 13.1 12.5  8.5  5.   6.3  5.6  7.2 12.1  8.3  8.5  5.
11.9 27.9 17.2 27.5 15.  17.2 17.9 16.3  7.   7.2  7.5 10.4  8.8  8.4
16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.   9.5 14.5 14.1 16.1 14.3
11.7 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8 11.8 14.9 12.6
14.1 13.  13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9 20.  16.4 17.7
19.5 20.2 21.4 19.9 19.  19.1 19.1 20.1 19.9 19.6 23.2 29.8 13.8 13.3
16.7 12.  14.6 21.4 23.  23.7 25.  21.8 20.6 21.2 19.1 20.6 15.2  7.
 8.1 13.6 20.1 21.8 24.5 23.1 19.7 18.3 21.2 17.5 16.8 22.4 20.6 23.9
22.  11.9]
```

Splitting the dataframe into two parts for training test purpose

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(independent, dependent, test_siz
```

Standardize features by removing the mean and scaling to unit variance

In [10]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Creating the neural network

Building the model

In [11]:
```
model = Sequential()
model.add(Dense(128, input_dim=13, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

#Output Layer
model.add(Dense(1, activation='linear'))
```

Compiling the model

In [12]:
```
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1792

 dense_1 (Dense)             (None, 64)                8256

 dropout (Dropout)           (None, 64)                0

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 10,113
Trainable params: 10,113
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model with dataset

In [14]:
```
history = model.fit(X_train, y_train, epochs =100, validation_data=(X_test, y_test))
```

```
Epoch 1/100
11/11 [==============================] - 2s 42ms/step - loss: 594.8780 - mae: 22.463
8 - val_loss: 501.5819 - val_mae: 20.5589
Epoch 2/100
11/11 [==============================] - 0s 8ms/step - loss: 544.2582 - mae: 21.2813
- val_loss: 450.8507 - val_mae: 19.2638
Epoch 3/100
11/11 [==============================] - 0s 8ms/step - loss: 480.6098 - mae: 19.7262
- val_loss: 382.4235 - val_mae: 17.4581
Epoch 4/100
11/11 [==============================] - 0s 8ms/step - loss: 401.9467 - mae: 17.5952
- val_loss: 293.4428 - val_mae: 14.9450
Epoch 5/100
11/11 [==============================] - 0s 8ms/step - loss: 288.9069 - mae: 14.5259
- val_loss: 193.9512 - val_mae: 11.6721
Epoch 6/100
11/11 [==============================] - 0s 10ms/step - loss: 191.7813 - mae: 11.105
3 - val_loss: 109.8338 - val_mae: 8.3761
Epoch 7/100
11/11 [==============================] - 0s 13ms/step - loss: 116.6904 - mae: 8.4382
- val_loss: 66.0906 - val_mae: 6.2935
Epoch 8/100
11/11 [==============================] - 0s 9ms/step - loss: 80.8698 - mae: 6.8195 -
val_loss: 50.0294 - val_mae: 5.3563
Epoch 9/100
11/11 [==============================] - 0s 8ms/step - loss: 67.1568 - mae: 6.2021 -
val_loss: 38.5601 - val_mae: 4.6348
Epoch 10/100
11/11 [==============================] - 0s 8ms/step - loss: 57.6914 - mae: 5.8040 -
val_loss: 31.6592 - val_mae: 4.0974
Epoch 11/100
11/11 [==============================] - 0s 10ms/step - loss: 53.6463 - mae: 5.4813
- val_loss: 27.8740 - val_mae: 3.7698
Epoch 12/100
11/11 [==============================] - 0s 12ms/step - loss: 55.0314 - mae: 5.5308
- val_loss: 25.1481 - val_mae: 3.4832
Epoch 13/100
11/11 [==============================] - 0s 9ms/step - loss: 47.7335 - mae: 5.1655 -
val_loss: 23.1004 - val_mae: 3.3053
Epoch 14/100
11/11 [==============================] - 0s 7ms/step - loss: 37.2234 - mae: 4.6874 -
val_loss: 21.6427 - val_mae: 3.1635
Epoch 15/100
11/11 [==============================] - 0s 8ms/step - loss: 42.1442 - mae: 4.9029 -
val_loss: 20.3925 - val_mae: 3.0601
```

```
Epoch 16/100
11/11 [==============================] - 0s 8ms/step - loss: 39.2747 - mae: 4.6361 -
val_loss: 19.6018 - val_mae: 2.9895
Epoch 17/100
11/11 [==============================] - 0s 8ms/step - loss: 43.8243 - mae: 4.9924 -
val_loss: 18.9098 - val_mae: 2.9237
Epoch 18/100
11/11 [==============================] - 0s 11ms/step - loss: 30.9494 - mae: 4.3278
- val_loss: 18.3561 - val_mae: 2.8787
Epoch 19/100
11/11 [==============================] - 0s 9ms/step - loss: 36.5611 - mae: 4.5882 -
val_loss: 17.8959 - val_mae: 2.8327
Epoch 20/100
11/11 [==============================] - 0s 10ms/step - loss: 36.7107 - mae: 4.5298
- val_loss: 16.7390 - val_mae: 2.7144
Epoch 21/100
11/11 [==============================] - 0s 10ms/step - loss: 41.0262 - mae: 4.7853
- val_loss: 16.4061 - val_mae: 2.7031
Epoch 22/100
11/11 [==============================] - 0s 9ms/step - loss: 33.1524 - mae: 4.3537 -
val_loss: 15.9431 - val_mae: 2.6422
Epoch 23/100
11/11 [==============================] - 0s 9ms/step - loss: 41.9565 - mae: 4.8762 -
val_loss: 16.1249 - val_mae: 2.6658
Epoch 24/100
11/11 [==============================] - 0s 9ms/step - loss: 34.3149 - mae: 4.3147 -
val_loss: 15.7943 - val_mae: 2.6578
Epoch 25/100
11/11 [==============================] - 0s 12ms/step - loss: 32.1147 - mae: 4.1484
- val_loss: 15.5908 - val_mae: 2.6311
Epoch 26/100
11/11 [==============================] - 0s 11ms/step - loss: 33.4529 - mae: 4.3877
- val_loss: 15.1347 - val_mae: 2.5838
Epoch 27/100
11/11 [==============================] - 0s 9ms/step - loss: 34.1410 - mae: 4.3368 -
val_loss: 14.4514 - val_mae: 2.5406
Epoch 28/100
11/11 [==============================] - 0s 8ms/step - loss: 33.2252 - mae: 4.3455 -
val_loss: 15.1017 - val_mae: 2.5818
Epoch 29/100
11/11 [==============================] - 0s 8ms/step - loss: 34.6353 - mae: 4.3231 -
val_loss: 15.4216 - val_mae: 2.6348
Epoch 30/100
11/11 [==============================] - 0s 9ms/step - loss: 32.8708 - mae: 4.3677 -
val_loss: 14.9542 - val_mae: 2.6370
Epoch 31/100
11/11 [==============================] - 0s 8ms/step - loss: 30.1581 - mae: 4.1980 -
val_loss: 14.4279 - val_mae: 2.5522
Epoch 32/100
11/11 [==============================] - 0s 8ms/step - loss: 34.0690 - mae: 4.3133 -
val_loss: 14.2370 - val_mae: 2.5307
Epoch 33/100
11/11 [==============================] - 0s 8ms/step - loss: 33.8557 - mae: 4.3754 -
val_loss: 14.9207 - val_mae: 2.6386
Epoch 34/100
11/11 [==============================] - 0s 8ms/step - loss: 33.2381 - mae: 4.3595 -
val_loss: 15.0067 - val_mae: 2.6238
Epoch 35/100
11/11 [==============================] - 0s 8ms/step - loss: 31.6980 - mae: 4.2479 -
val_loss: 14.1079 - val_mae: 2.5562
Epoch 36/100
11/11 [==============================] - 0s 10ms/step - loss: 34.5846 - mae: 4.2782
- val_loss: 14.4266 - val_mae: 2.5683
Epoch 37/100
11/11 [==============================] - 0s 12ms/step - loss: 27.5506 - mae: 3.9550
- val_loss: 14.0097 - val_mae: 2.5153
Epoch 38/100
11/11 [==============================] - 0s 9ms/step - loss: 29.7051 - mae: 4.1834 -
val_loss: 13.6098 - val_mae: 2.5045
```

```
Epoch 39/100
11/11 [==============================] - 0s 9ms/step - loss: 28.1436 - mae: 3.9303 -
val_loss: 13.1325 - val_mae: 2.4699
Epoch 40/100
11/11 [==============================] - 0s 8ms/step - loss: 34.2349 - mae: 4.3714 -
val_loss: 13.0460 - val_mae: 2.4550
Epoch 41/100
11/11 [==============================] - 0s 8ms/step - loss: 29.4799 - mae: 3.9773 -
val_loss: 13.5800 - val_mae: 2.5003
Epoch 42/100
11/11 [==============================] - 0s 8ms/step - loss: 27.6785 - mae: 3.9298 -
val_loss: 12.7931 - val_mae: 2.4029
Epoch 43/100
11/11 [==============================] - 0s 8ms/step - loss: 32.6485 - mae: 4.1609 -
val_loss: 12.7849 - val_mae: 2.4088
Epoch 44/100
11/11 [==============================] - 0s 8ms/step - loss: 32.5446 - mae: 4.2877 -
val_loss: 13.3112 - val_mae: 2.4558
Epoch 45/100
11/11 [==============================] - 0s 8ms/step - loss: 27.3966 - mae: 3.8649 -
val_loss: 13.5872 - val_mae: 2.4882
Epoch 46/100
11/11 [==============================] - 0s 8ms/step - loss: 29.2429 - mae: 4.1869 -
val_loss: 12.5736 - val_mae: 2.4151
Epoch 47/100
11/11 [==============================] - 0s 8ms/step - loss: 29.1843 - mae: 4.0763 -
val_loss: 13.0630 - val_mae: 2.4340
Epoch 48/100
11/11 [==============================] - 0s 9ms/step - loss: 30.4442 - mae: 4.3000 -
val_loss: 12.9947 - val_mae: 2.4427
Epoch 49/100
11/11 [==============================] - 0s 7ms/step - loss: 31.2358 - mae: 4.3237 -
val_loss: 12.6250 - val_mae: 2.4031
Epoch 50/100
11/11 [==============================] - 0s 8ms/step - loss: 32.0350 - mae: 4.3174 -
val_loss: 12.7361 - val_mae: 2.3713
Epoch 51/100
11/11 [==============================] - 0s 8ms/step - loss: 29.7161 - mae: 3.9424 -
val_loss: 12.1410 - val_mae: 2.3453
Epoch 52/100
11/11 [==============================] - 0s 8ms/step - loss: 30.2601 - mae: 4.1618 -
val_loss: 12.1046 - val_mae: 2.2845
Epoch 53/100
11/11 [==============================] - 0s 7ms/step - loss: 29.6655 - mae: 4.0094 -
val_loss: 13.7399 - val_mae: 2.5033
Epoch 54/100
11/11 [==============================] - 0s 8ms/step - loss: 28.7741 - mae: 3.8988 -
val_loss: 12.5506 - val_mae: 2.3735
Epoch 55/100
11/11 [==============================] - 0s 8ms/step - loss: 23.7701 - mae: 3.5679 -
val_loss: 11.8593 - val_mae: 2.3202
Epoch 56/100
11/11 [==============================] - 0s 8ms/step - loss: 34.0934 - mae: 4.4160 -
val_loss: 12.2576 - val_mae: 2.3038
Epoch 57/100
11/11 [==============================] - 0s 7ms/step - loss: 28.6253 - mae: 4.0399 -
val_loss: 12.0595 - val_mae: 2.2651
Epoch 58/100
11/11 [==============================] - 0s 8ms/step - loss: 28.5515 - mae: 4.0721 -
val_loss: 11.7722 - val_mae: 2.2827
Epoch 59/100
11/11 [==============================] - 0s 8ms/step - loss: 29.3942 - mae: 4.0599 -
val_loss: 12.0372 - val_mae: 2.2931
Epoch 60/100
11/11 [==============================] - 0s 8ms/step - loss: 28.0857 - mae: 3.9726 -
val_loss: 11.7334 - val_mae: 2.3075
Epoch 61/100
11/11 [==============================] - 0s 9ms/step - loss: 26.5669 - mae: 3.8502 -
val_loss: 12.4096 - val_mae: 2.3715
```

```
Epoch 62/100
11/11 [==============================] - 0s 12ms/step - loss: 31.0314 - mae: 4.2524
- val_loss: 11.4896 - val_mae: 2.2649
Epoch 63/100
11/11 [==============================] - 0s 9ms/step - loss: 24.9976 - mae: 3.8415 -
val_loss: 12.0763 - val_mae: 2.3211
Epoch 64/100
11/11 [==============================] - 0s 8ms/step - loss: 31.4533 - mae: 4.1935 -
val_loss: 12.9626 - val_mae: 2.4517
Epoch 65/100
11/11 [==============================] - 0s 11ms/step - loss: 29.6848 - mae: 4.0366
- val_loss: 12.8057 - val_mae: 2.4295
Epoch 66/100
11/11 [==============================] - 0s 9ms/step - loss: 31.0766 - mae: 4.1030 -
val_loss: 11.7680 - val_mae: 2.2943
Epoch 67/100
11/11 [==============================] - 0s 9ms/step - loss: 25.3113 - mae: 3.7607 -
val_loss: 11.4072 - val_mae: 2.2627
Epoch 68/100
11/11 [==============================] - 0s 8ms/step - loss: 27.2083 - mae: 3.9658 -
val_loss: 11.8338 - val_mae: 2.3060
Epoch 69/100
11/11 [==============================] - 0s 8ms/step - loss: 27.7070 - mae: 3.9319 -
val_loss: 12.2777 - val_mae: 2.3501
Epoch 70/100
11/11 [==============================] - 0s 7ms/step - loss: 27.7673 - mae: 4.1368 -
val_loss: 12.4269 - val_mae: 2.3863
Epoch 71/100
11/11 [==============================] - 0s 8ms/step - loss: 26.2247 - mae: 3.8185 -
val_loss: 12.7703 - val_mae: 2.4469
Epoch 72/100
11/11 [==============================] - 0s 8ms/step - loss: 28.6457 - mae: 3.9100 -
val_loss: 12.2995 - val_mae: 2.3922
Epoch 73/100
11/11 [==============================] - 0s 8ms/step - loss: 26.7181 - mae: 3.8108 -
val_loss: 12.3056 - val_mae: 2.3785
Epoch 74/100
11/11 [==============================] - 0s 8ms/step - loss: 25.9396 - mae: 3.8535 -
val_loss: 12.1410 - val_mae: 2.3715
Epoch 75/100
11/11 [==============================] - 0s 8ms/step - loss: 24.4448 - mae: 3.6945 -
val_loss: 12.6699 - val_mae: 2.4196
Epoch 76/100
11/11 [==============================] - 0s 8ms/step - loss: 27.3243 - mae: 3.9372 -
val_loss: 11.7946 - val_mae: 2.3145
Epoch 77/100
11/11 [==============================] - 0s 8ms/step - loss: 30.0221 - mae: 4.0814 -
val_loss: 11.5706 - val_mae: 2.3039
Epoch 78/100
11/11 [==============================] - 0s 11ms/step - loss: 29.7178 - mae: 4.1449
- val_loss: 11.7025 - val_mae: 2.3328
Epoch 79/100
11/11 [==============================] - 0s 9ms/step - loss: 28.1257 - mae: 4.0784 -
val_loss: 11.8114 - val_mae: 2.3179
Epoch 80/100
11/11 [==============================] - 0s 9ms/step - loss: 26.6568 - mae: 3.9636 -
val_loss: 12.1877 - val_mae: 2.3474
Epoch 81/100
11/11 [==============================] - 0s 8ms/step - loss: 26.3173 - mae: 3.8428 -
val_loss: 11.8078 - val_mae: 2.3201
Epoch 82/100
11/11 [==============================] - 0s 8ms/step - loss: 28.5098 - mae: 4.0987 -
val_loss: 11.9378 - val_mae: 2.3800
Epoch 83/100
11/11 [==============================] - 0s 8ms/step - loss: 28.2484 - mae: 4.1497 -
val_loss: 12.2785 - val_mae: 2.3804
Epoch 84/100
11/11 [==============================] - 0s 8ms/step - loss: 25.9026 - mae: 3.7604 -
val_loss: 12.2218 - val_mae: 2.3623
```

```
Epoch 85/100
11/11 [==============================] - 0s 8ms/step - loss: 22.9306 - mae: 3.7259 -
val_loss: 12.1636 - val_mae: 2.3505
Epoch 86/100
11/11 [==============================] - 0s 9ms/step - loss: 29.6360 - mae: 3.9759 -
val_loss: 11.9656 - val_mae: 2.3306
Epoch 87/100
11/11 [==============================] - 0s 7ms/step - loss: 25.9415 - mae: 3.8867 -
val_loss: 12.6801 - val_mae: 2.4209
Epoch 88/100
11/11 [==============================] - 0s 9ms/step - loss: 29.3300 - mae: 3.9551 -
val_loss: 12.2482 - val_mae: 2.3724
Epoch 89/100
11/11 [==============================] - 0s 8ms/step - loss: 27.4408 - mae: 3.6758 -
val_loss: 11.5763 - val_mae: 2.2904
Epoch 90/100
11/11 [==============================] - 0s 8ms/step - loss: 26.2976 - mae: 3.7874 -
val_loss: 12.0017 - val_mae: 2.3264
Epoch 91/100
11/11 [==============================] - 0s 8ms/step - loss: 27.0847 - mae: 4.0028 -
val_loss: 12.1870 - val_mae: 2.3389
Epoch 92/100
11/11 [==============================] - 0s 8ms/step - loss: 24.7617 - mae: 3.6715 -
val_loss: 11.5055 - val_mae: 2.2647
Epoch 93/100
11/11 [==============================] - 0s 8ms/step - loss: 23.5859 - mae: 3.7647 -
val_loss: 12.0525 - val_mae: 2.3340
Epoch 94/100
11/11 [==============================] - 0s 8ms/step - loss: 23.5501 - mae: 3.6372 -
val_loss: 12.1854 - val_mae: 2.3492
Epoch 95/100
11/11 [==============================] - 0s 8ms/step - loss: 28.6185 - mae: 4.0752 -
val_loss: 11.7613 - val_mae: 2.3027
Epoch 96/100
11/11 [==============================] - 0s 8ms/step - loss: 27.8427 - mae: 4.0300 -
val_loss: 11.9862 - val_mae: 2.3297
Epoch 97/100
11/11 [==============================] - 0s 19ms/step - loss: 27.1506 - mae: 3.8961
- val_loss: 11.9213 - val_mae: 2.3053
Epoch 98/100
11/11 [==============================] - 0s 8ms/step - loss: 22.2612 - mae: 3.5872 -
val_loss: 12.5827 - val_mae: 2.3599
Epoch 99/100
11/11 [==============================] - 0s 8ms/step - loss: 27.0519 - mae: 3.9136 -
val_loss: 12.0597 - val_mae: 2.3271
Epoch 100/100
11/11 [==============================] - 0s 8ms/step - loss: 30.7245 - mae: 4.0694 -
val_loss: 13.1034 - val_mae: 2.5188
```

In [21]:

```
#Predict on test data
predictions = model.predict(X_test)
print("Predicted values are: ", predictions[:5])
print("Real values are: ", y_test[:5])
```

```
6/6 [==============================] - 0s 2ms/step
Predicted values are:  [[24.777115]
 [33.65618 ]
 [17.774288]
 [25.399376]
 [16.290585]]
Real values are:  [23.6 32.4 13.6 22.8 16.1]
```

In [25]:

```
r2_score(y_test,predictions)
```

Out[25]:

```
0.8268546096136964
```

In [ ]: