```
print("Name : Bhagvat Nivrutti Mutthe ")
print("Roll No : BCB-76")
print("Assignment no.1")
print("LINEAR REGRESSION USING
                                DEEP NEURAL NETWORK")
```

```
Roll
Name : Bhagvat Nivrutti Mutthe
     No : BCB-76
   Assignment no.1
   LINEAR REGRESSION  USING  DEEP  NEURAL  NETWORK
```

```
from sklearn import datasets
   import warnings
   import numpy as np
   import pandas as pd    as plt
   import matplotlib.pyplot
   import seaborn as sns
   %matplotlib inline
   #ignore warnings
   warnings.filterwarnings('ignore')
```

```
# read data from sklearn data set
data=datasets.load_boston()
df=pd.DataFrame(data.data,columns=data.feature_names)
df['price']=data.target
df
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7
...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0

506 rows × 14 columns

Next steps: Generate code with df View recommended plots

```
df.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    float64
 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    float64
 9   TAX       506 non-null    float64
 10  PTRATIO   506 non-null    float64
 11  B         506 non-null    float64
 12  LSTAT     506 non-null    float64
 13  price     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

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

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
price      0
dtype: int64
```

```
df.describe()
```

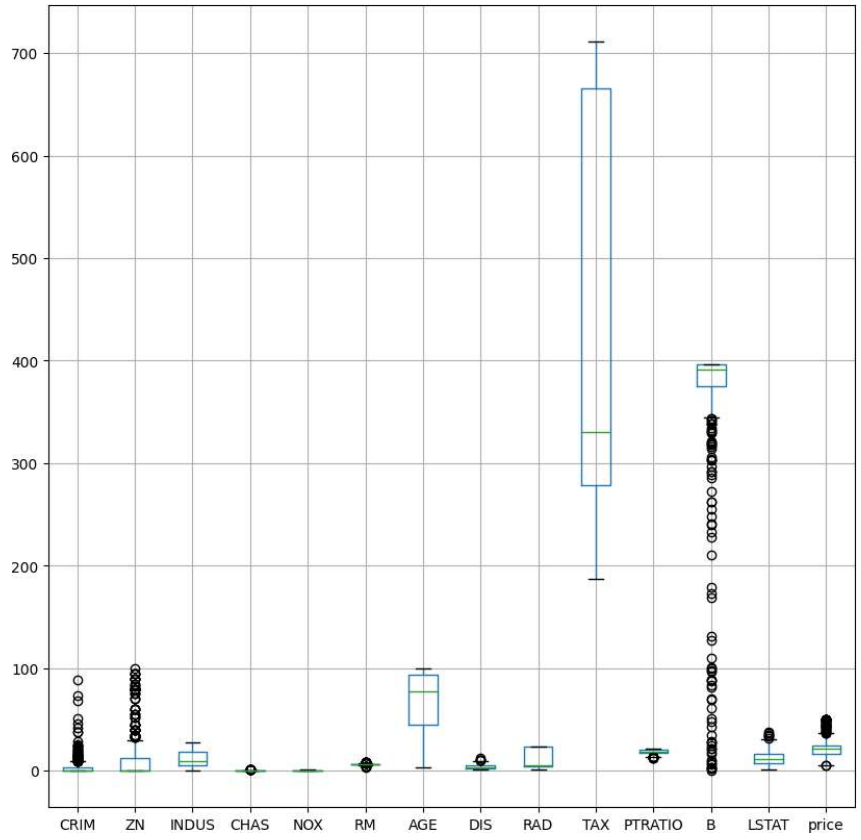
	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

```
df.shape
```

```
(506, 14)
```

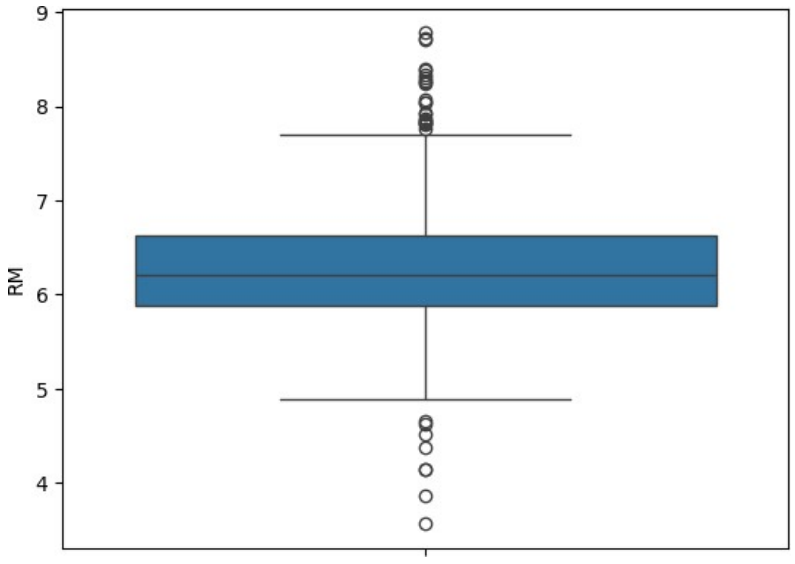
```
#univariate EDA
fig=plt.figure(figsize=(10,10))
df.boxplot()
```

```
<Axes: >
```



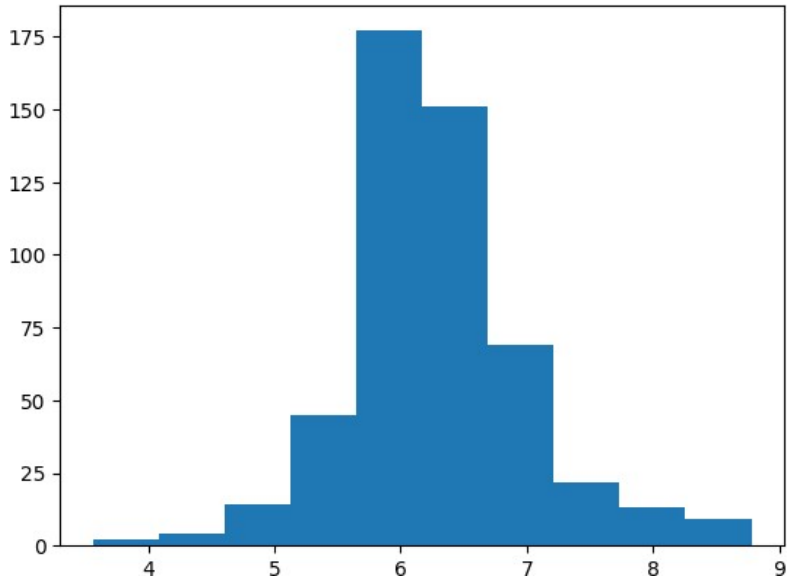

```
sns.boxplot(df["RM"])
```

```
<Axes: ylabel='RM'>
```




```
plt.hist(df["RM"])
```

```
(array([  2.,   4.,  14.,  45., 177., 151.,  69.,  22.,  13.,   9.]),
 array([3.561 , 4.0829, 4.6048, 5.1267, 5.6486, 6.1705, 6.6924, 7.2143,
        7.7362, 8.2581, 8.78  ]),
 <BarContainer object of 10 artists>)
```

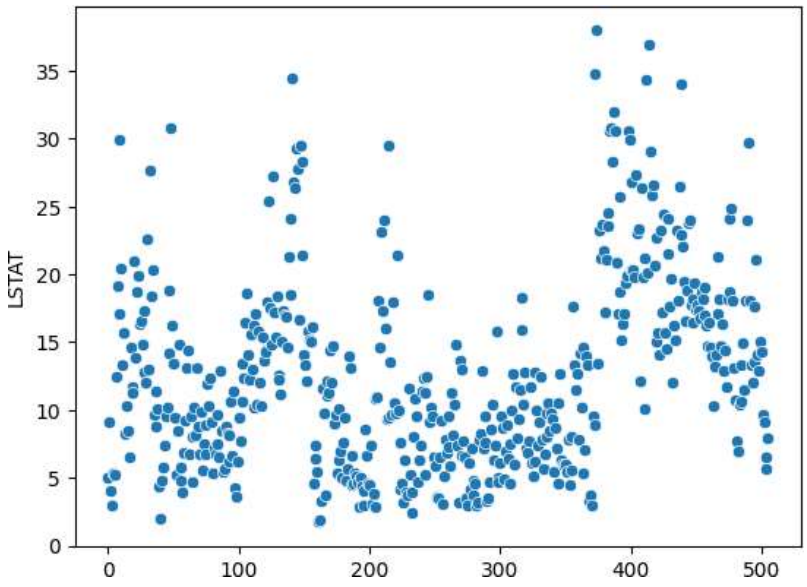



```
df["RM"].value_counts()
```

```
RM
5.713    3
6.167    3
6.127    3
6.229    3
6.405    3
        ..
5.859    1
6.416    1
5.572    1
5.880    1
6.976    1
Name: count, Length: 446, dtype: int64
```

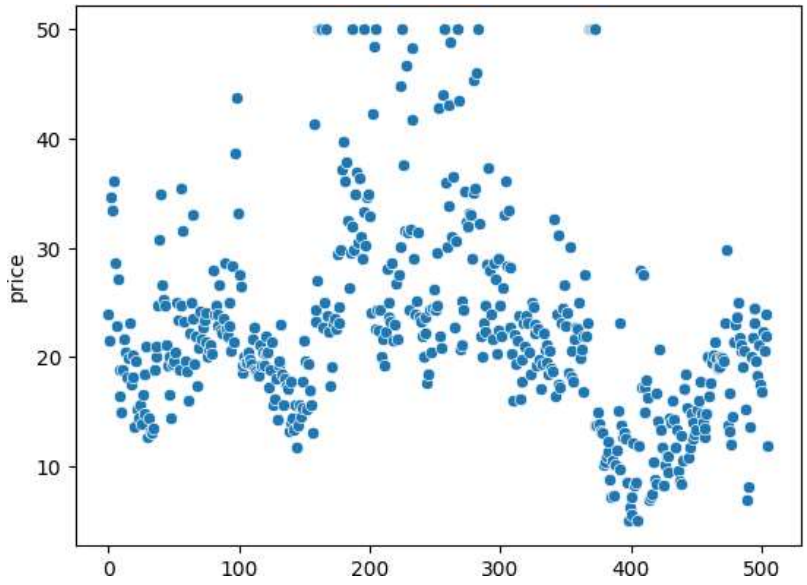
```
#bivariate EDA
sns.scatterplot(df["LSTAT"])
```

```
<Axes: ylabel='LSTAT'>
```



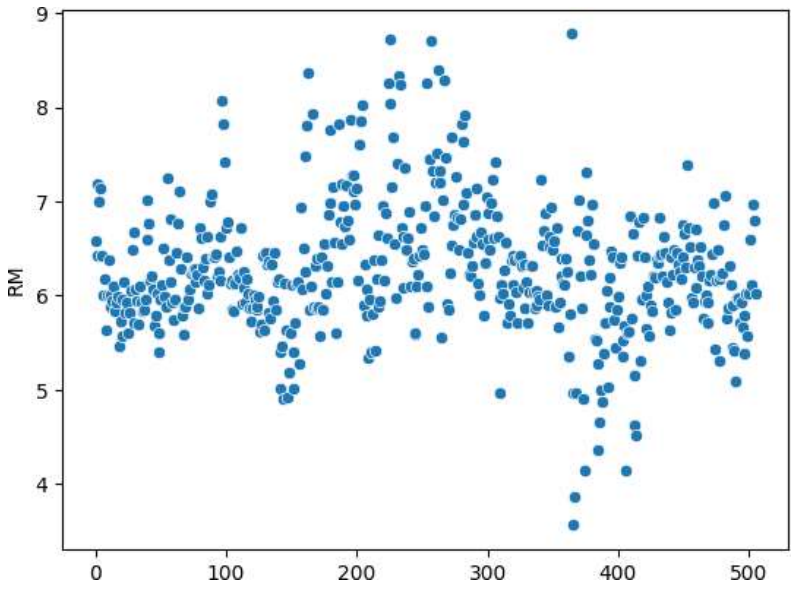

```
sns.scatterplot(df["price"])
```

```
<Axes: ylabel='price'>
```



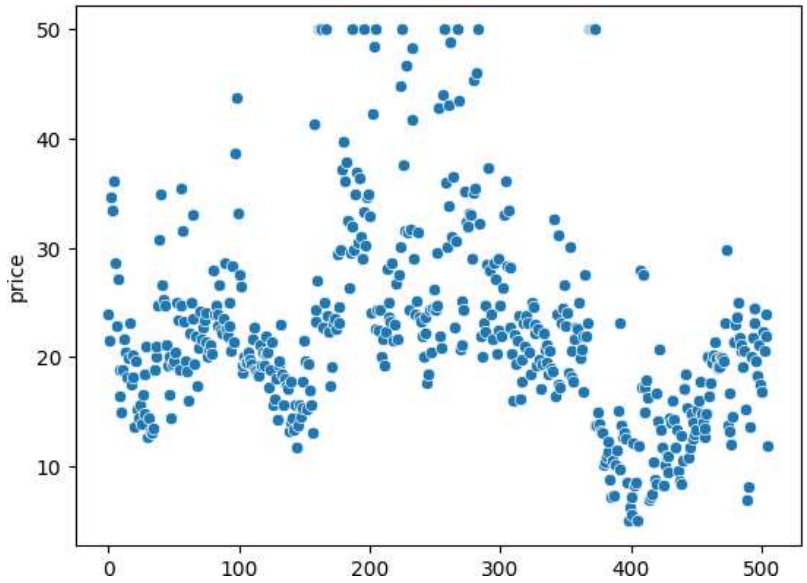

```
sns.scatterplot(df["RM"])
```

```
<Axes: ylabel='RM'>
```



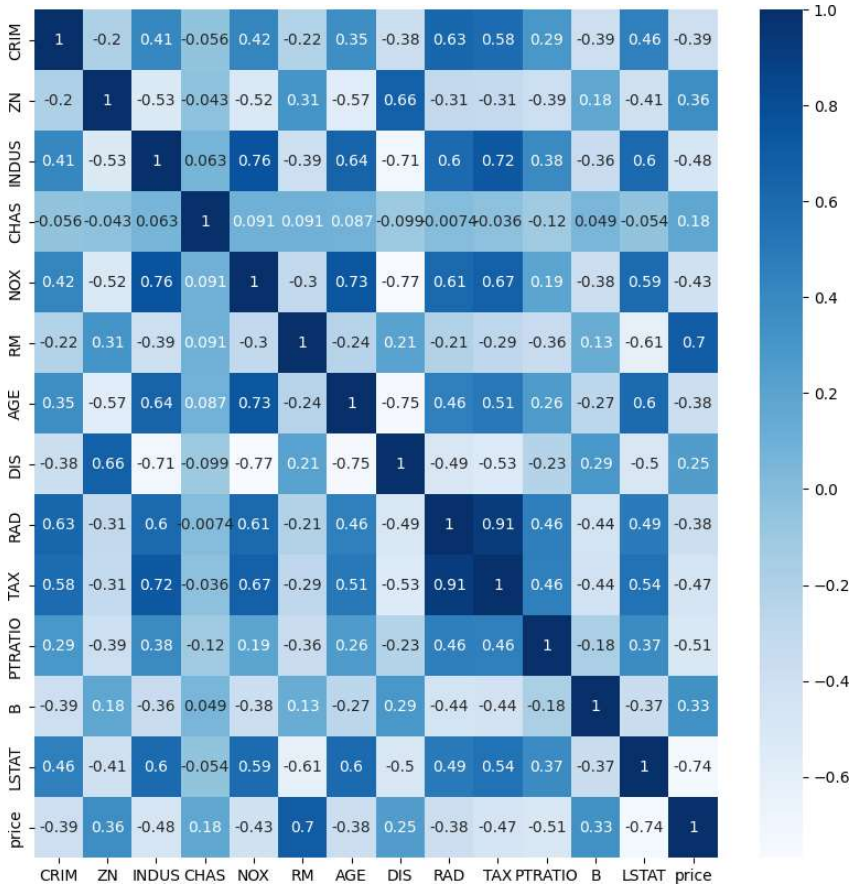

```
sns.scatterplot(df["price"])
```

```
<Axes: ylabel='price'>
```




```
#Multivariate EDA
fig= plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(),annot=True,cmap="Blues")
```

```
<Axes: >
```




```
pip install keras_tuner
```

```
Collecting keras_tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.1/129.1 kB 3.2 MB/s eta 0:00:00
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (from keras_tuner) (2.15.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from keras_tuner) (24.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from keras_tuner) (2.31.0)
Collecting kt-legacy (from keras_tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->ker
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->keras_tuner) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->keras_tun
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->keras_tun
Installing collected packages: kt-legacy, keras_tuner
Successfully installed keras_tuner-1.4.7 kt-legacy-1.0.5
```

```
import tensorflow.keras as tk


model=tk.Sequential()


#adding input layer

model.add(tk.layers.Input(shape=(13,)))


#adding first hidden layer
model.add(tk.layers.Dense(units=6,activation="relu",kernel_initializer="he_uniform"))



#adding second hidden layer
model.add(tk.layers.Dense(units=6,activation="relu",kernel_initializer="he_uniform"))


#adding output layer
model.add(tk.layers.Dense(units=1,activation="relu",kernel_initializer="he_uniform"))


#compiling the model
#model.compile(optimizer="adam",loss="mean_squared_error")


#compiling the model
model.compile(optimizer="adam",loss="mean_absolute_error")


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 6)                 84

 dense_1 (Dense)             (None, 6)                 42

 dense_2 (Dense)             (None, 1)                 7

 dense_3 (Dense)             (None, 1)                 2

=================================================================
Total params: 135 (540.00 Byte)
Trainable params: 135 (540.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
df.head()
x=df.iloc[:,:-1] #independent
display(x)
y=df['price'] #dependent
display(y)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7
...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0

506 rows × 13 columns

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
```

Next steps: Generate code with x View recommended plots

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=10)


#training the model
import time
start=time.time()
obj1=model.fit(x=xtrain,y=ytrain,epochs=50,batch_size=64,validation_data=(xtest,ytest))
```

```
Epoch 1/50
7/7 [==============================]  - 2s 39ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 2/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 3/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 4/50
7/7 [==============================]  - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 5/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 6/50
7/7 [==============================]  - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 7/50
7/7 [==============================]  - 0s 6ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 8/50
7/7 [==============================]  - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 9/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 10/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 11/50
7/7 [==============================]  - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 12/50
7/7 [==============================]  - 0s 8ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 13/50
7/7 [==============================]  - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 14/50
7/7 [==============================]  - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 15/50
7/7 [==============================]  - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 16/50
7/7 [==============================]  - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
```

```
Epoch 17/50
7/7 [==============================] - 0s 8ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 18/50
7/7 [==============================] - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 19/50
7/7 [==============================] - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 20/50
7/7 [==============================] - 0s 11ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 21/50
7/7 [==============================] - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 22/50
7/7 [==============================] - 0s 9ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 23/50
7/7 [==============================] - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 24/50
7/7 [==============================] - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 25/50
7/7 [==============================] - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 26/50
7/7 [==============================] - 0s 10ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 27/50
7/7 [==============================] - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 28/50
7/7 [==============================] - 0s 7ms/step - loss: 21.8418 - val_loss: 25.2696
Epoch 29/50
7/7 [==============================] - 0s 10ms/step - loss: 21.8418 - val loss: 25.2696
```

```
ypred=model.predict([[0.00632,18.0,2.31,0.0,0.538,6.575,65.2,4.0900,1.0,296.0,15.3,396.90,4.98]])
ypred
```

```
1/1 [==============================] - 0s 147ms/step
array([[0.]], dtype=float32)
```

```
ypred1=model.predict(xtest)
display(ypred1,ytest)
ypred1.shape,ytest.shape
```

```
4/4 [==============================] - 0s 3ms/step
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
```

```
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]], dtype=float32)
```