Deep Learning Lab Assignment 01

Name of Student: Aniket Yashvant Sandbhor
Branch: Computer Engineering
Exam number: B190424404

0.1 Problem Statement: 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.

[1]:
```python
# Import Libraries
from sklearn.datasets import fetch_openml
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

[2]:
```python
#ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```python
# read data from sklearn data set
data = fetch_openml(name='boston', version=1)
df=pd.DataFrame(data.data,columns=data.feature_names)
df['price']=data.target
df
```

[3]:
```
        CRIM    ZN  INDUS CHAS    NOX     RM   AGE     DIS RAD    TAX  
↪ \
0    0.00632  18.0   2.31    0  0.538  6.575  65.2  4.0900   1  296.0
1    0.02731   0.0   7.07    0  0.469  6.421  78.9  4.9671   2  242.0
2    0.02729   0.0   7.07    0  0.469  7.185  61.1  4.9671   2  242.0
3    0.03237   0.0   2.18    0  0.458  6.998  45.8  6.0622   3  222.0
4    0.06905   0.0   2.18    0  0.458  7.147  54.2  6.0622   3  222.0
..       ...   ...    ...  ...    ...    ...   ...     ...  ..    ...
501  0.06263   0.0  11.93    0  0.573  6.593  69.1  2.4786   1  273.0
502  0.04527   0.0  11.93    0  0.573  6.120  76.7  2.2875   1  273.0
```

```
503  0.06076   0.0  11.93    0  0.573  6.976  91.0  2.1675   1  273.0
504  0.10959   0.0  11.93    0  0.573  6.794  89.3  2.3889   1  273.0
505  0.04741   0.0  11.93    0  0.573  6.030  80.8  2.5050   1  273.0

     PTRATIO       B  LSTAT  price
0       15.3  396.90   4.98   24.0
1       17.8  396.90   9.14   21.6
2       17.8  392.83   4.03   34.7
3       18.7  394.63   2.94   33.4
4       18.7  396.90   5.33   36.2
..       ...     ...    ...    ...
501     21.0  391.99   9.67   22.4
502     21.0  396.90   9.08   20.6
503     21.0  396.90   5.64   23.9
504     21.0  393.45   6.48   22.0
505     21.0  396.90   7.88   11.9

[506 rows x 14 columns]
```

[4]:
```
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    category
 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    category
 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: category(2), float64(12)
memory usage: 49.0 KB
```

[5]:
```
df.isnull().sum()
```

[5]:
```
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
```

[6]:
```
df.describe()
```

[6]:
```
             CRIM          ZN       INDUS         NOX          RM  \
↪     AGE  \
count  506.000000  506.000000  506.000000  506.000000  506.000000
↪506.000000
mean     3.613524   11.363636   11.136779    0.554695    6.284634
↪68.574901
std      8.601545   23.322453    6.860353    0.115878    0.702617
↪28.148861
min      0.006320    0.000000    0.460000    0.385000    3.561000
↪2.900000
25%      0.082045    0.000000    5.190000    0.449000    5.885500
↪45.025000
50%      0.256510    0.000000    9.690000    0.538000    6.208500
↪77.500000
75%      3.677083   12.500000   18.100000    0.624000    6.623500
↪94.075000
max     88.976200  100.000000   27.740000    0.871000    8.780000
↪100.000000

              DIS         TAX     PTRATIO           B       LSTAT  \
↪   price
count  506.000000  506.000000  506.000000  506.000000  506.000000
↪506.000000
mean     3.795043  408.237154   18.455534  356.674032   12.653063
↪22.532806
std      2.105710  168.537116    2.164946   91.294864    7.141062
↪9.197104
min      1.129600  187.000000   12.600000    0.320000    1.730000
↪5.000000
25%      2.100175  279.000000   17.400000  375.377500    6.950000
↪17.025000
```

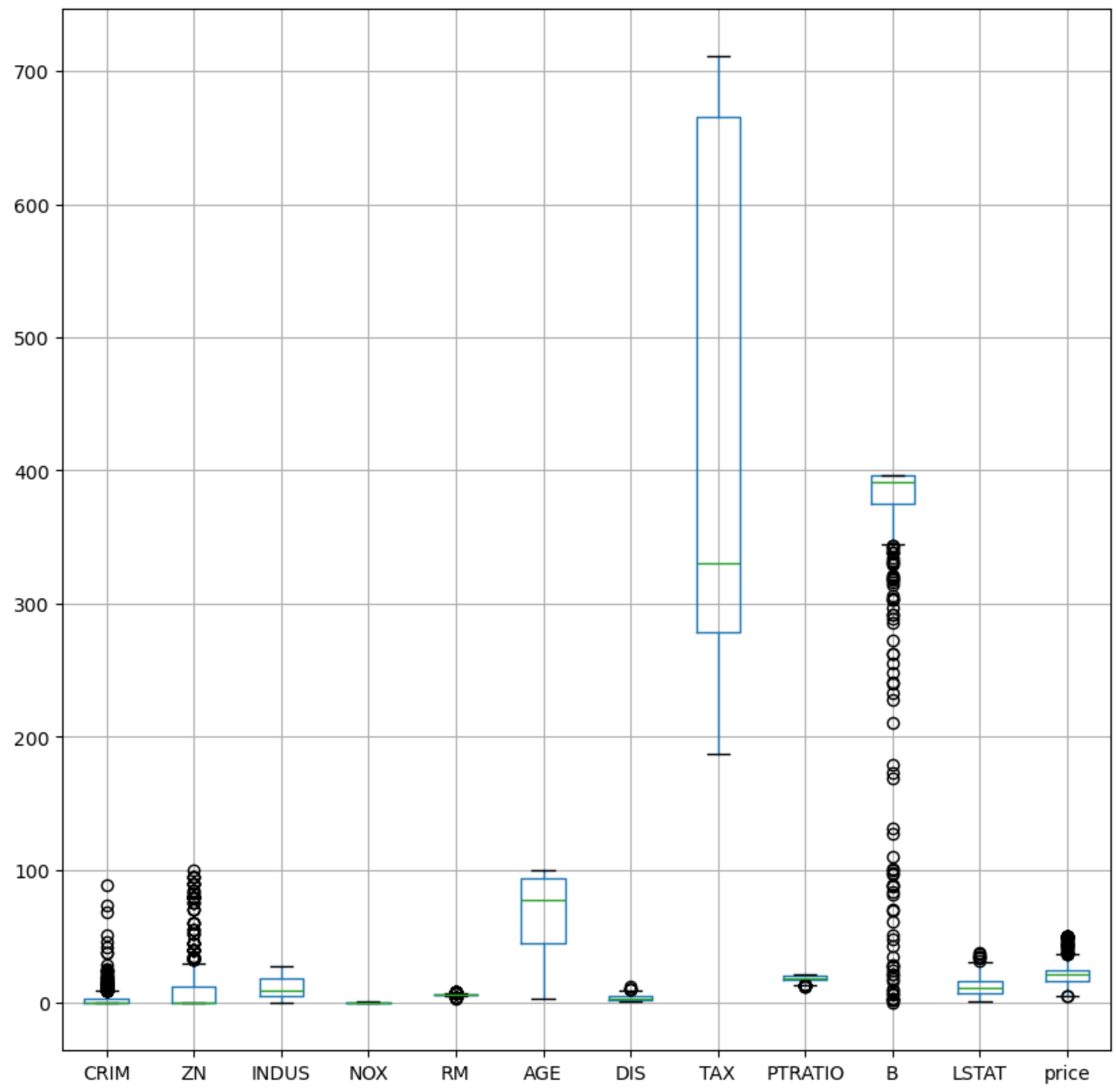
```
50%       3.207450  330.000000   19.050000  391.440000   11.360000
 ↪21.200000
75%       5.188425  666.000000   20.200000  396.225000   16.955000
 ↪25.000000
max      12.126500  711.000000   22.000000  396.900000   37.970000
 ↪50.000000
```

[7]:
```
df.shape
```

[7]:
```
(506, 14)
```

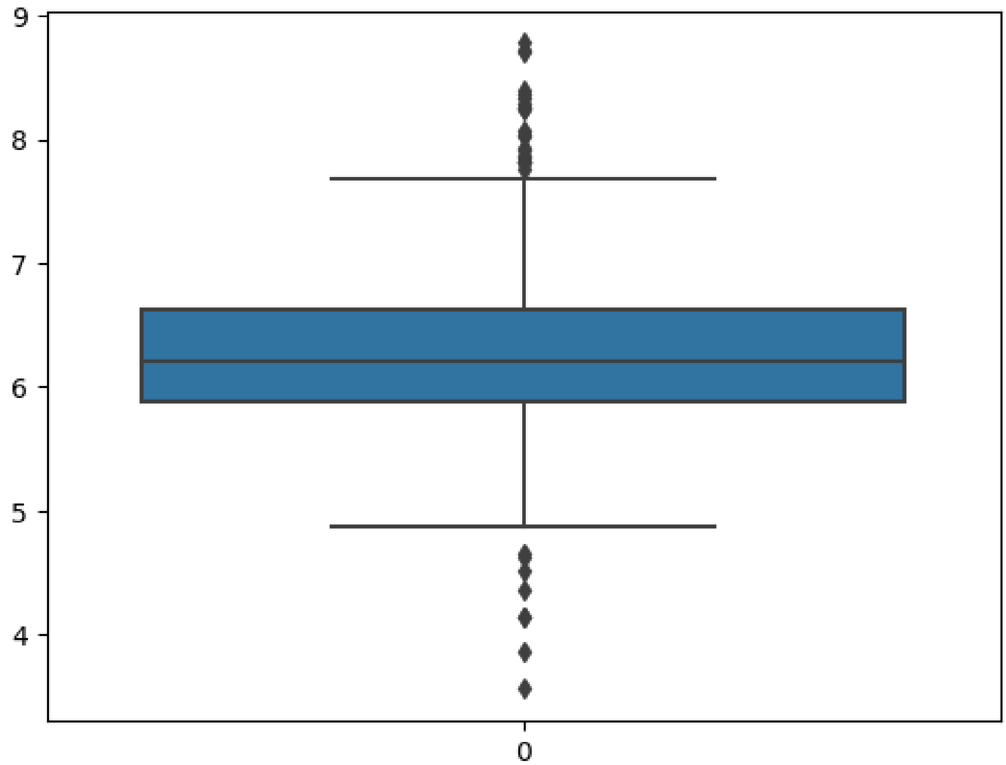
[8]:
```
#univariate EDA
fig = plt.figure(figsize = (10,10))
df.boxplot()
```

[8]:
```
<Axes: >
```



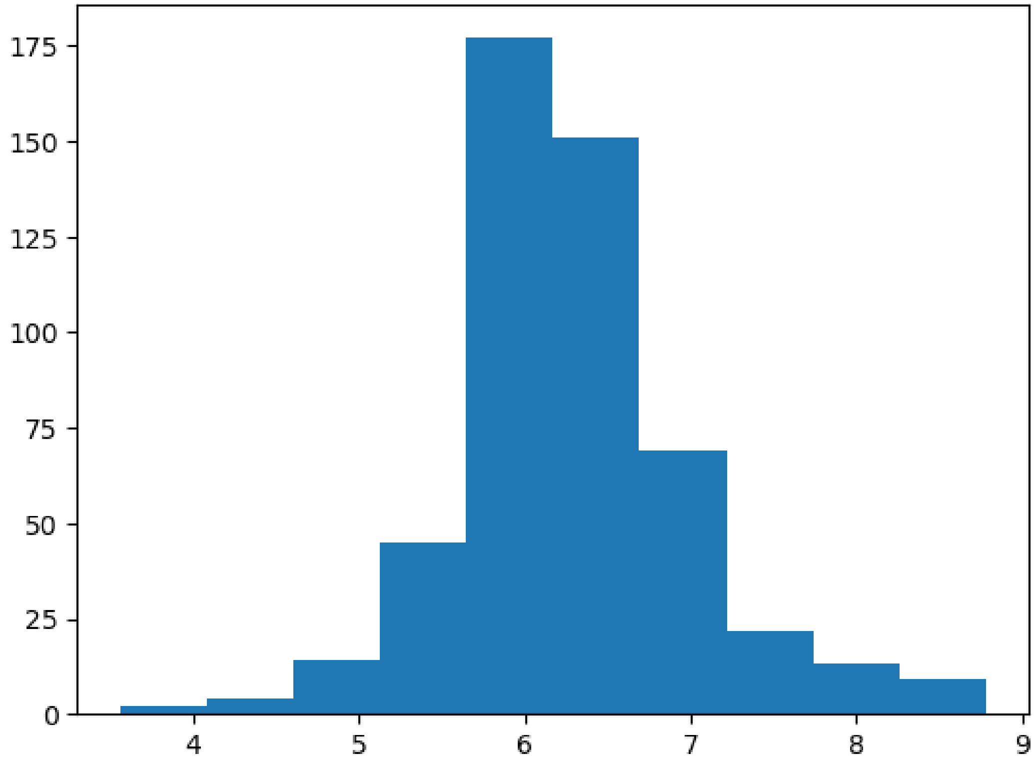
[9]:
```
sns.boxplot(df["RM"])
```

[9]:
```
<Axes: >
```



[10]:
```
plt.hist(df["RM"])
```

[10]:
```
(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>)
```

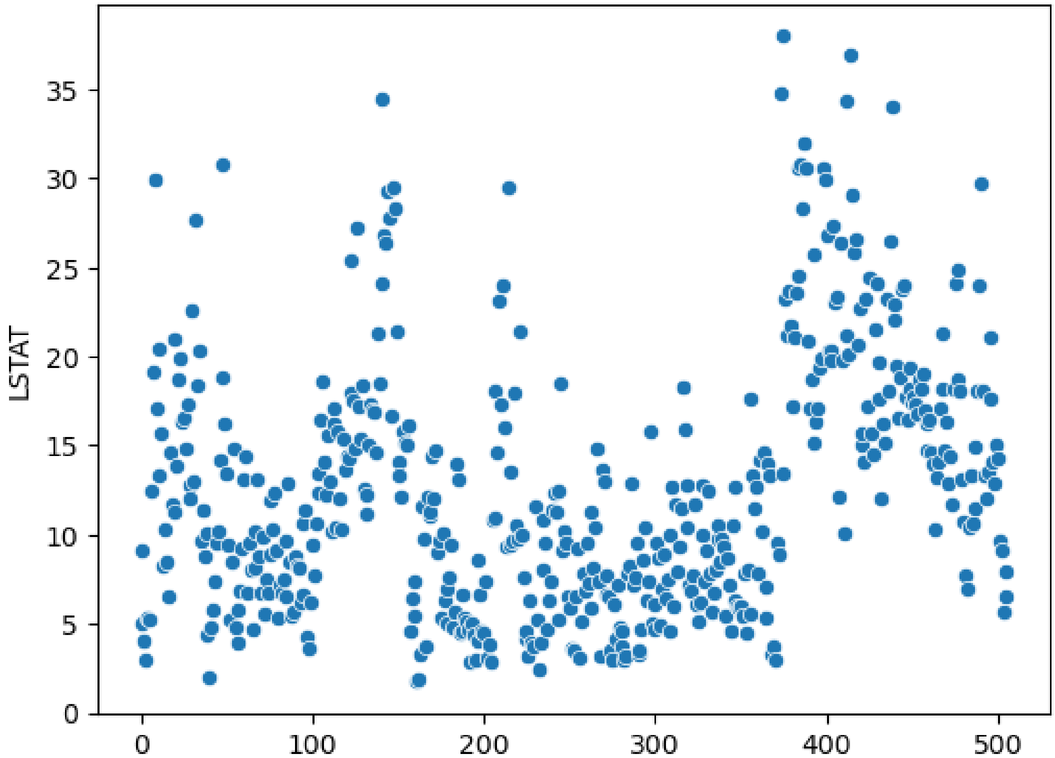


[11]:
```
df["RM"].value_counts()
```

[11]:
```
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
```

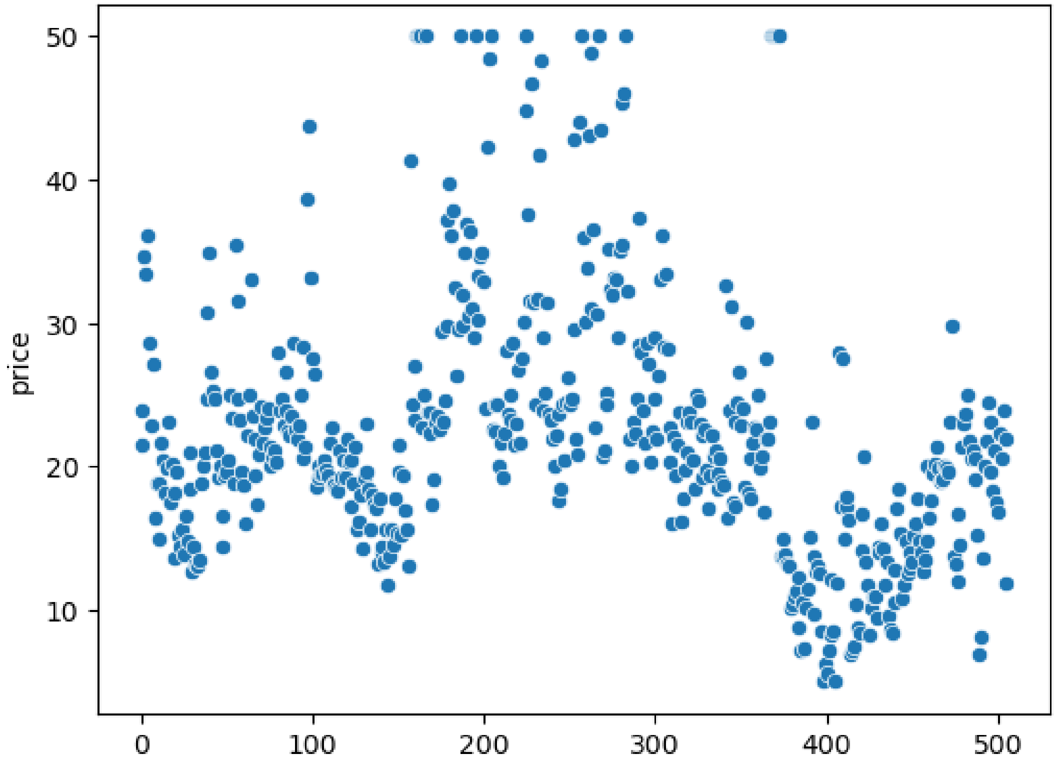
[12]:
```
#bivariate EDA
sns.scatterplot(df["LSTAT"])
```

[12]:
```
<Axes: ylabel='LSTAT'>
```



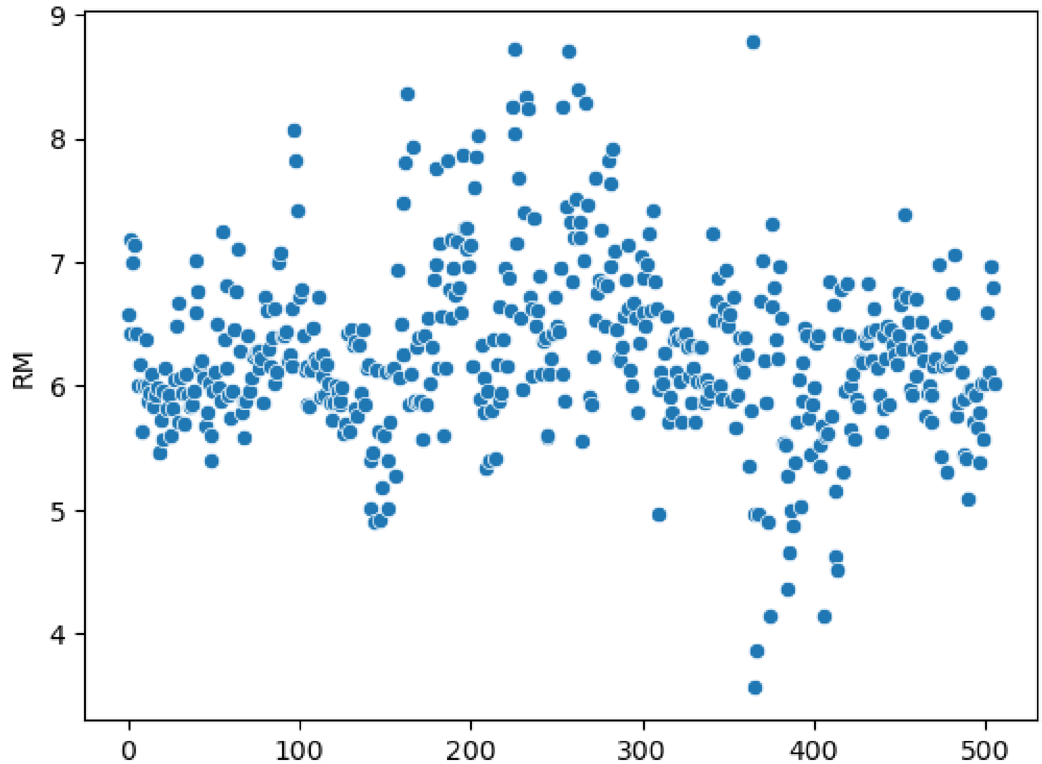
[13]:
```
sns.scatterplot(df["price"])
```

[13]: <Axes: ylabel='price'>



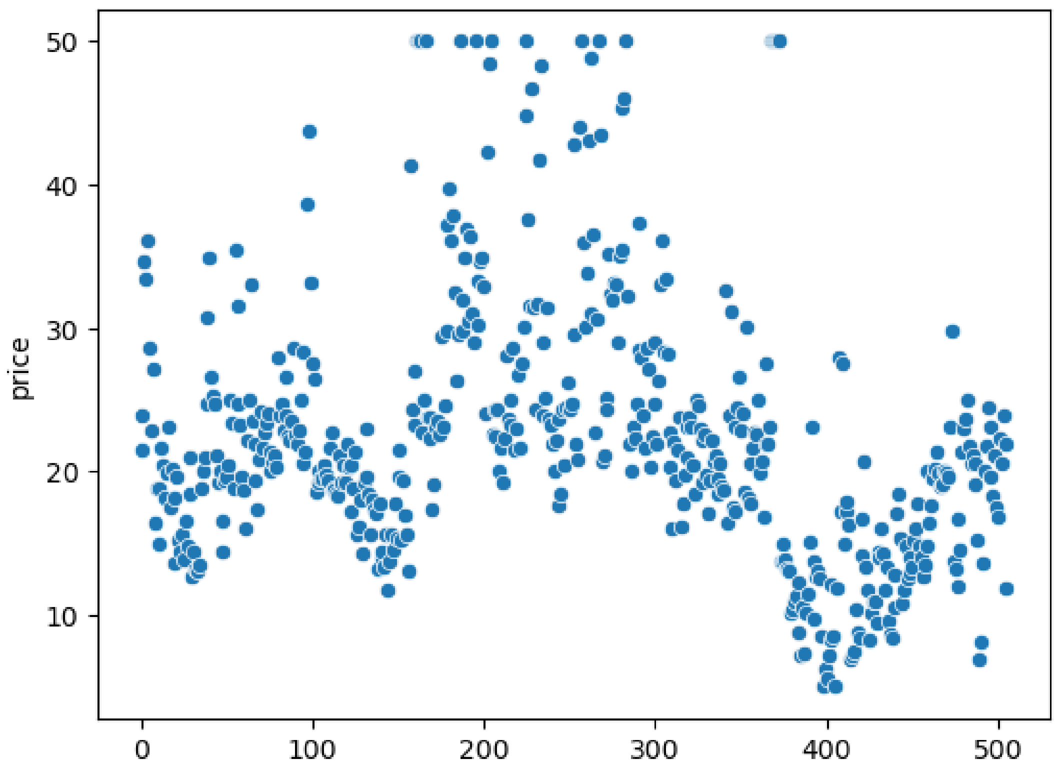
[14]:
```
sns.scatterplot(df["RM"])
```

[14]: <Axes: ylabel='RM'>



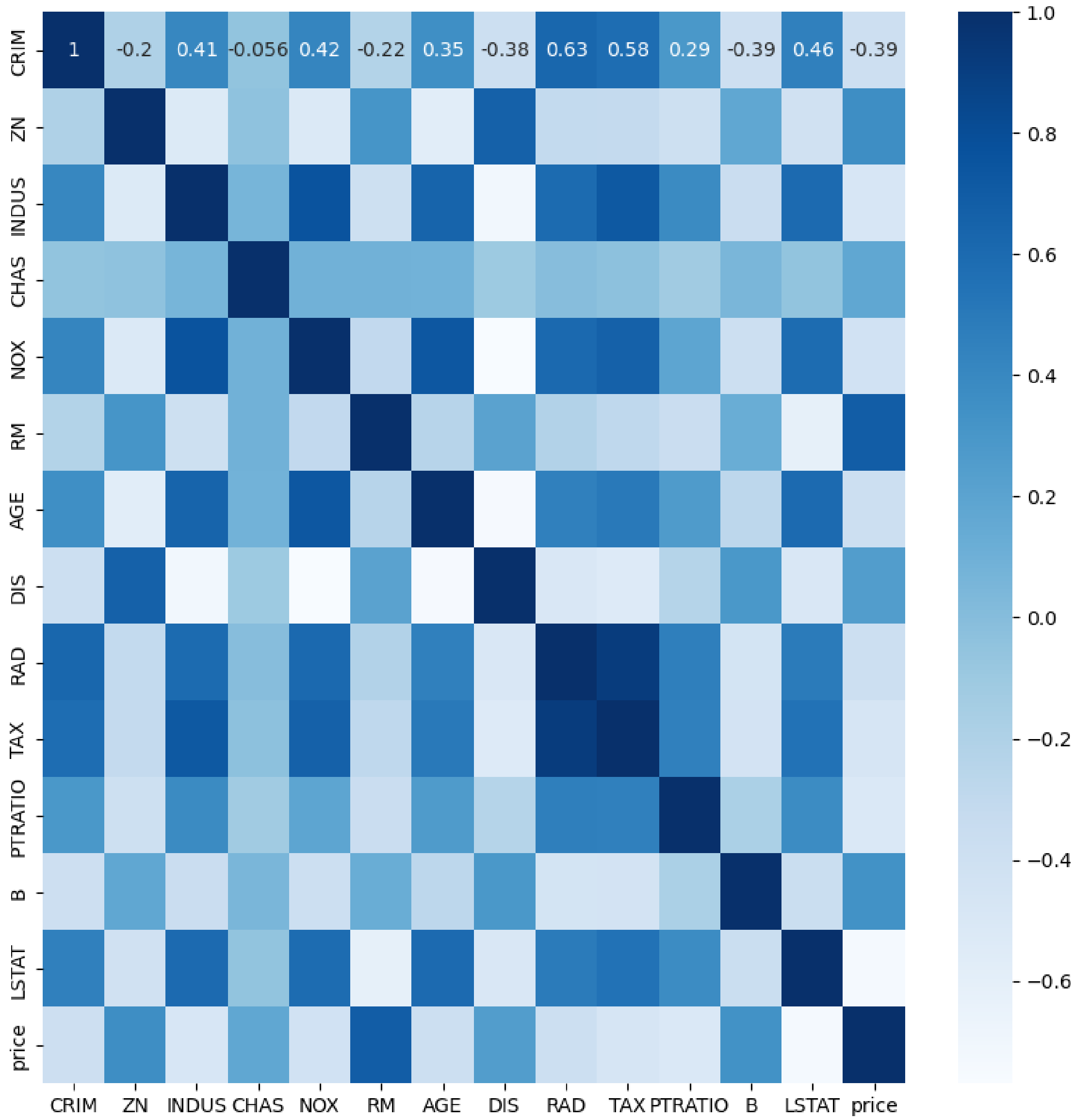

```
[15]: sns.scatterplot(df["price"])
```

```
[15]: <Axes: ylabel='price'>
```



[16]:
```
#Multivariate EDA
fig = plt.subplots(figsize = (10,10))
sns.heatmap(df.corr(),annot = True, cmap = "Blues")
```

[16]: <Axes: >



[17]:
```
import tensorflow.keras as tk
```

[18]:
```
model = tk.Sequential()
```

[19]:
```
#adding input layer

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

[20]:
```
#adding first hidden layer
model.add(tk.layers.Dense(units=6, activation="relu",
 ↪kernel_initializer="he_uniform"))
```

[21]:
```
#adding second hidden layer
model.add(tk.layers.Dense(units=6, activation="relu",
 ↪kernel_initializer="he_uniform"))
```

[22]:
```
#adding output layer
model.add(tk.layers.Dense(units=1, activation="relu",
 ↪kernel_initializer="he_uniform"))
```

[23]:
```
#compiling the model
model.compile(optimizer="adam", loss="mean_absolute_error")
```

[24]:
```
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

=================================================================
Total params: 133
Trainable params: 133
Non-trainable params: 0
_________________________________________________________________
```

[25]:
```
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
 ↪\
0     0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1     0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2     0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3     0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4     0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0
..        ...   ...    ...   ...    ...    ...   ...     ...  ..    ...
501   0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273.0
502   0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273.0
503   0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273.0
504   0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273.0
505   0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273.0
```

```
     PTRATIO       B  LSTAT
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 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
Name: price, Length: 506, dtype: float64
```

[26]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.2,
 ↪random_state=10)
```

[27]:
```
#training the model
import time
start = time.time()
```

[28]:
```
xtrain = np.asarray(xtrain).astype(np.int)
ytrain = np.asarray(ytrain).astype(np.int)
xtest = np.asarray(xtest).astype(np.int)
ytest = np.asarray(ytest).astype(np.int)
```

[34]:
```
obj1 = model.fit(
    x=xtrain,
    y=ytrain,
    epochs=50,
    batch_size=64,
    validation_data=(xtest,ytest))
```

```
Epoch 1/50
```

```
7/7 [==============================] - 0s 6ms/step - loss: 5.0830 -
 ↪val_loss:
6.7354
Epoch 2/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0733 -
 ↪val_loss:
6.7919
Epoch 3/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0686 -
 ↪val_loss:
6.7067
Epoch 4/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0782 -
 ↪val_loss:
6.6325
Epoch 5/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0707 -
 ↪val_loss:
6.7508
Epoch 6/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0561 -
 ↪val_loss:
6.7306
Epoch 7/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0620 -
 ↪val_loss:
6.6313
Epoch 8/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0552 -
 ↪val_loss:
6.6990
Epoch 9/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0499 -
 ↪val_loss:
6.7977
Epoch 10/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0503 -
 ↪val_loss:
6.6881
Epoch 11/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0370 -
 ↪val_loss:
6.6896
Epoch 12/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0333 -
 ↪val_loss:
6.7339
```

```
Epoch 13/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0302 -
 ↪val_loss:
6.7115
Epoch 14/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0274 -
 ↪val_loss:
6.7117
Epoch 15/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0306 -
 ↪val_loss:
6.6701
Epoch 16/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0173 -
 ↪val_loss:
6.7728
Epoch 17/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0513 -
 ↪val_loss:
6.7743
Epoch 18/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0141 -
 ↪val_loss:
6.6130
Epoch 19/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0256 -
 ↪val_loss:
6.6556
Epoch 20/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0174 -
 ↪val_loss:
6.7011
Epoch 21/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0173 -
 ↪val_loss:
6.6496
Epoch 22/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0009 -
 ↪val_loss:
6.7282
Epoch 23/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0067 -
 ↪val_loss:
6.7249
Epoch 24/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0082 -
 ↪val_loss:
```

```
6.6563
Epoch 25/50
7/7 [==============================] - 0s 2ms/step - loss: 5.0019 -
↪val_loss:
6.6628
Epoch 26/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9956 -
↪val_loss:
6.7347
Epoch 27/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9930 -
↪val_loss:
6.6936
Epoch 28/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9909 -
↪val_loss:
6.6724
Epoch 29/50
7/7 [==============================] - 0s 5ms/step - loss: 4.9847 -
↪val_loss:
6.7199
Epoch 30/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9870 -
↪val_loss:
6.6280
Epoch 31/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9832 -
↪val_loss:
6.6773
Epoch 32/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9764 -
↪val_loss:
6.6701
Epoch 33/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9739 -
↪val_loss:
6.6586
Epoch 34/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9703 -
↪val_loss:
6.6681
Epoch 35/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9674 -
↪val_loss:
6.7015
Epoch 36/50
```

```
7/7 [==============================] - 0s 2ms/step - loss: 4.9664 -
 ↪val_loss:
6.6575
Epoch 37/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9591 -
 ↪val_loss:
6.6909
Epoch 38/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9593 -
 ↪val_loss:
6.7107
Epoch 39/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9582 -
 ↪val_loss:
6.6839
Epoch 40/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9743 -
 ↪val_loss:
6.5945
Epoch 41/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9524 -
 ↪val_loss:
6.6900
Epoch 42/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9473 -
 ↪val_loss:
6.6846
Epoch 43/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9459 -
 ↪val_loss:
6.6782
Epoch 44/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9461 -
 ↪val_loss:
6.5616
Epoch 45/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9502 -
 ↪val_loss:
6.6170
Epoch 46/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9398 -
 ↪val_loss:
6.6692
Epoch 47/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9279 -
 ↪val_loss:
6.5818
```

```
Epoch 48/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9493 -
↪val_loss:
6.5832
Epoch 49/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9279 -
↪val_loss:
6.7665
Epoch 50/50
7/7 [==============================] - 0s 2ms/step - loss: 4.9292 -
↪val_loss:
6.6342
```

[35]:
```
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 27ms/step
```

[35]:
```
array([[22.94993]], dtype=float32)
```

[36]:
```
ypred1 = model.predict(xtest)
display(ypred1,ytest)
ypred1.shape, ytest.shape
```

```
4/4 [==============================] - 0s 888us/step
```

```
array([[25.702642 ],
       [28.444447 ],
       [29.357456 ],
       [24.524887 ],
       [15.102709 ],
       [13.014451 ],
       [17.893679 ],
       [18.427303 ],
       [23.910418 ],
       [20.499844 ],
       [19.007633 ],
       [20.855028 ],
       [23.029997 ],
       [26.344334 ],
       [26.583637 ],
       [22.481096 ],
       [22.999012 ],
       [21.77428  ],
       [21.911695 ],
       [13.86223  ],
       [16.847582 ],
       [15.415434 ],
```

```
[18.554    ],
[22.433075 ],
[24.853453 ],
[23.048958 ],
[21.288006 ],
[15.178105 ],
[23.710459 ],
[11.5643635],
[25.822113 ],
[19.607464 ],
[28.905306 ],
[23.938332 ],
[17.647133 ],
[20.03867  ],
[12.418465 ],
[30.430744 ],
[21.826927 ],
[21.266056 ],
[21.893162 ],
[21.19336  ],
[16.616415 ],
[26.998022 ],
[21.258476 ],
[25.5597   ],
[15.997657 ],
[17.092764 ],
[23.973526 ],
[20.108427 ],
[19.09886  ],
[17.565575 ],
[20.237057 ],
[22.400234 ],
[20.651041 ],
[23.127792 ],
[19.27963  ],
[17.704473 ],
[18.934395 ],
[17.282207 ],
[18.38596  ],
[19.250677 ],
[18.759264 ],
[22.879692 ],
[22.183954 ],
[18.372986 ],
[22.661    ],
[14.923089 ],
[22.053057 ],
[24.852676 ],
```

```
       [17.178604 ],
       [22.577019 ],
       [17.797623 ],
       [16.092999 ],
       [19.727102 ],
       [19.897713 ],
       [20.272139 ],
       [17.312191 ],
       [23.825518 ],
       [24.038198 ],
       [22.211403 ],
       [24.702425 ],
       [25.042713 ],
       [24.86498  ],
       [21.141356 ],
       [20.80736  ],
       [21.673819 ],
       [26.757322 ],
       [22.850698 ],
       [11.911356 ],
       [22.840862 ],
       [20.951946 ],
       [21.666296 ],
       [22.766096 ],
       [21.841616 ],
       [21.025793 ],
       [23.607447 ],
       [20.80088  ],
       [21.27327  ],
       [19.212612 ],
       [22.801687 ],
       [ 6.646275 ]], dtype=float32)
array([28, 31, 23, 26, 19, 14, 50, 14, 20, 37, 20, 27, 36, 32, 33, 48,
 ↪24,
       26, 23, 17, 41, 14, 18, 25, 36, 19, 27, 14, 46, 17, 30, 31, 23,
 ↪24,
       16, 18,  8, 37, 22, 22, 46, 30, 12, 29, 16, 23, 19, 21, 45, 15,
 ↪22,
       12, 24, 43, 22, 33, 19, 22, 16, 15, 19, 21, 50, 50, 29, 17, 22,
 ↪ 8,
       32, 42, 12, 28, 19, 50, 27, 23, 50,  7, 18, 37, 22, 22, 17, 22,
 ↪23,
       23, 27, 29, 22,  7, 20, 18, 21, 23, 16, 15, 23, 24, 22, 19, 22,
 ↪18])
```

```
[36]: ((102, 1), (102,))
```

[37]:
```
from sklearn.metrics import mean_absolute_error
error = mean_absolute_error(ytest, ypred1)
error
```

[37]: 6.634203176872403

[38]:
```
plt.figure(figsize=(10,10))
ax1 = sns.distplot(ytest, hist=False, color="r", label="Actual Value")
sns.distplot(ypred1, hist=False, color="b", label="Predicted Values",
 ↪ax=ax1)
```

[38]: <Axes: ylabel='Density'>