```
import pandas as pd
import matplotlib.pyplot as plt
```

Load Dataset

```
df = pd.read_csv('Boston.csv')
df.head(10)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.90
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	386.63
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71

```
df.drop(columns=['Unnamed: 15','Unnamed: 16'],inplace=True)
```

```
df.drop(columns=['CAT. MEDV'],inplace=True)
```

Checking for null values

```
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
MEDV       0
dtype: int64
```

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

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.57
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.14
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.90
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.02
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.50
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.07
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.00

Checking correlation with target variable MEDV

```
df.corr()['MEDV'].sort_values()
```

```
LSTAT     -0.737663
PTRATIO   -0.507787
INDUS     -0.483725
TAX       -0.468536
NOX       -0.427321
CRIM      -0.388305
RAD       -0.381626
AGE       -0.376955
CHAS       0.175260
DIS        0.249929
B          0.333461
ZN         0.360445
RM         0.695360
MEDV       1.000000
Name: MEDV, dtype: float64
```

```
X = df.loc[:,['LSTAT','PTRATIO','RM']]
Y = df.loc[:,"MEDV"]
X.shape,Y.shape
```

```
((506, 3), (506,))
```

Preparing training and testing data set

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,random_state=10)
```

Normalizing training and testing dataset

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
scaler.fit(x_train)
```

```
▾ StandardScaler
StandardScaler()
```

```
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

Preparing model

```
from keras.models import Sequential
from keras.layers import Dense
```

```
model = Sequential()
```

```
model.add(Dense(128,input_shape=(3,),activation='relu',name='input'))
model.add(Dense(64,activation='relu',name='layer_1'))
model.add(Dense(1,activation='linear',name='output'))
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (Dense)               (None, 128)               512

 layer_1 (Dense)             (None, 64)                8256

 output (Dense)              (None, 1)                 65

=================================================================
Total params: 8,833
Trainable params: 8,833
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x_train,y_train,epochs=100,validation_split=0.05)
```

```
12/12 [==============================] - 0s 7ms/step - loss: 11.4140 - mae: 2.4701 - val_loss: 81.1654 - val_m
Epoch 87/100
12/12 [==============================] - 0s 8ms/step - loss: 11.3241 - mae: 2.4628 - val_loss: 79.9070 - val_m
Epoch 88/100
12/12 [==============================] - 0s 8ms/step - loss: 11.3244 - mae: 2.4665 - val_loss: 80.6700 - val_m
Epoch 89/100
12/12 [==============================] - 0s 8ms/step - loss: 11.2633 - mae: 2.4476 - val_loss: 80.9691 - val_m
Epoch 90/100
12/12 [==============================] - 0s 6ms/step - loss: 11.3179 - mae: 2.4568 - val_loss: 79.3354 - val_m
Epoch 91/100
12/12 [==============================] - 0s 7ms/step - loss: 11.2052 - mae: 2.4402 - val_loss: 81.1970 - val_m
Epoch 92/100
12/12 [==============================] - 0s 8ms/step - loss: 11.2065 - mae: 2.4538 - val_loss: 81.0578 - val_m
Epoch 93/100
12/12 [==============================] - 0s 7ms/step - loss: 11.1539 - mae: 2.4599 - val_loss: 80.4147 - val_m
Epoch 94/100
12/12 [==============================] - 0s 8ms/step - loss: 11.0334 - mae: 2.4449 - val_loss: 80.7812 - val_m
Epoch 95/100
12/12 [==============================] - 0s 8ms/step - loss: 11.0836 - mae: 2.4435 - val_loss: 80.6245 - val_m
Epoch 96/100
12/12 [==============================] - 0s 9ms/step - loss: 11.1159 - mae: 2.4439 - val_loss: 81.0934 - val_m
Epoch 97/100
12/12 [==============================] - 0s 6ms/step - loss: 11.0722 - mae: 2.4450 - val_loss: 83.7489 - val_m
Epoch 98/100
12/12 [==============================] - 0s 8ms/step - loss: 11.0852 - mae: 2.4527 - val_loss: 78.1914 - val_m
Epoch 99/100
12/12 [==============================] - 0s 7ms/step - loss: 10.9237 - mae: 2.4263 - val_loss: 82.1337 - val_m
Epoch 100/100
12/12 [==============================] - 0s 8ms/step - loss: 10.9013 - mae: 2.4176 - val_loss: 80.4514 - val_m
<keras.callbacks.History at 0x7fa1cca76bc0>
```

```
output = model.evaluate(x_test,y_test)
```

```
4/4 [==============================] - 0s 7ms/step - loss: 23.1388 - mae: 3.2114
```

```
print(f"Mean Squared Error: {output[0]}"
      ,f"Mean Absolute Error: {output[1]}",sep="\n")
```

```
Mean Squared Error: 23.138755798339844
Mean Absolute Error: 3.2113683223724365
```

```
y_pred = model.predict(x=x_test)
```

```
4/4 [==============================] - 0s 4ms/step
```

```
print(*zip(y_pred,y_test))
```

```
(array([25.836397], dtype=float32), 28.4) (array([30.462648], dtype=float32), 31.1) (array([25.88291], dtype=flo
```

✓ 0s completed at 08:58