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

Load Dataset

```
url = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"
df = pd.read_csv(url)
df.head(10)
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
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

Next steps: Generate code with df | View recommended plots | New interactive sheet

Checking for null values

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

	0
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	age	dis	rad	tax	ptratio	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	50
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	35
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	9
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	37
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	39
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	39
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	39

Checking correlation with target variable MEDV

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

	medv
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

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()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input (Dense)                   │ (None, 128)            │           512 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ layer_1 (Dense)                 │ (None, 64)             │         8,256 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ output (Dense)                  │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 8,833 (34.50 KB)
 Trainable params: 8,833 (34.50 KB)
 Non-trainable params: 0 (0.00 B)
```

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

```
Epoch 84/100
Epoch 84/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 9.9835 - mae: 2.3671 - val_loss: 80.5598 - val_mae: 5.1366
Epoch 85/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 10.3276 - mae: 2.4002 - val_loss: 84.2075 - val_mae: 5.2334
Epoch 86/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - loss: 11.8153 - mae: 2.4825 - val_loss: 79.3377 - val_mae: 5.1011
Epoch 87/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.9645 - mae: 2.5138 - val_loss: 83.2999 - val_mae: 5.2307
Epoch 88/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 12.5619 - mae: 2.4674 - val_loss: 84.3078 - val_mae: 5.2289
Epoch 89/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 10.0256 - mae: 2.3766 - val_loss: 83.0850 - val_mae: 5.1253
Epoch 90/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 9.1409 - mae: 2.3161 - val_loss: 80.8863 - val_mae: 5.0894
Epoch 91/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 12.1685 - mae: 2.5031 - val_loss: 83.0917 - val_mae: 5.2055
Epoch 92/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 8.9529 - mae: 2.2443 - val_loss: 83.1287 - val_mae: 5.1559
Epoch 93/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 10.1395 - mae: 2.4177 - val_loss: 81.3497 - val_mae: 5.1094
Epoch 94/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - loss: 9.9061 - mae: 2.3541 - val_loss: 84.2678 - val_mae: 5.2567
Epoch 95/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step - loss: 11.4230 - mae: 2.4551 - val_loss: 81.8559 - val_mae: 5.1166
Epoch 96/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - loss: 10.4983 - mae: 2.4857 - val_loss: 84.9311 - val_mae: 5.1525
Epoch 97/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - loss: 9.1775 - mae: 2.2612 - val_loss: 82.4787 - val_mae: 5.1379
Epoch 98/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 9.1371 - mae: 2.1630 - val_loss: 80.8819 - val_mae: 5.1033
Epoch 99/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 10.4211 - mae: 2.3409 - val_loss: 82.4397 - val_mae: 5.1492
Epoch 100/100
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 10.6222 - mae: 2.4133 - val_loss: 85.2638 - val_mae: 5.1754
<keras.src.callbacks.history.History at 0x7eb3262015d0>
```

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

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - loss: 20.8151 - mae: 3.0742
```

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

```
Mean Squared Error: 22.936630249023438
Mean Absolute Error: 3.112030506134033
```

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

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```

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

```
(array([25.099472], dtype=float32), 28.4) (array([30.49332], dtype=float32), 31.1) (array([25.702652], dtype=float32), 23.5) (array([26.
```