In [1]:
```python
import numpy as np
import pandas as pd

```

In [2]:
```python
data = pd.read_csv("boston.csv")
```

In [3]:
```python
data.head()

```

Out[3]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIC
0	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3
1	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8
2	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8
3	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7
4	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7

In [4]:
```python
data.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'D
IS',
       'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'Price'],
      dtype='object')
```

In [5]:
```python
data.head(n=10)
```

Out[5]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATI
0	0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.
1	1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.
2	2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.
3	3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.
4	4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.
5	5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.
6	6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.
7	7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.
8	8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.
9	9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.

In [6]:
```python
data.shape
```

Out[6]:
```
(506, 15)
```

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
Unnamed: 0    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
Price         0
dtype: int64
```

In [8]:
```
data.describe()
```

Out[8]:

	Unnamed: 0	CRIM	ZN	INDUS	CHAS	NOX	RM
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	252.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617
min	0.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000
25%	126.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500
50%	252.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500
75%	378.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500
max	505.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000

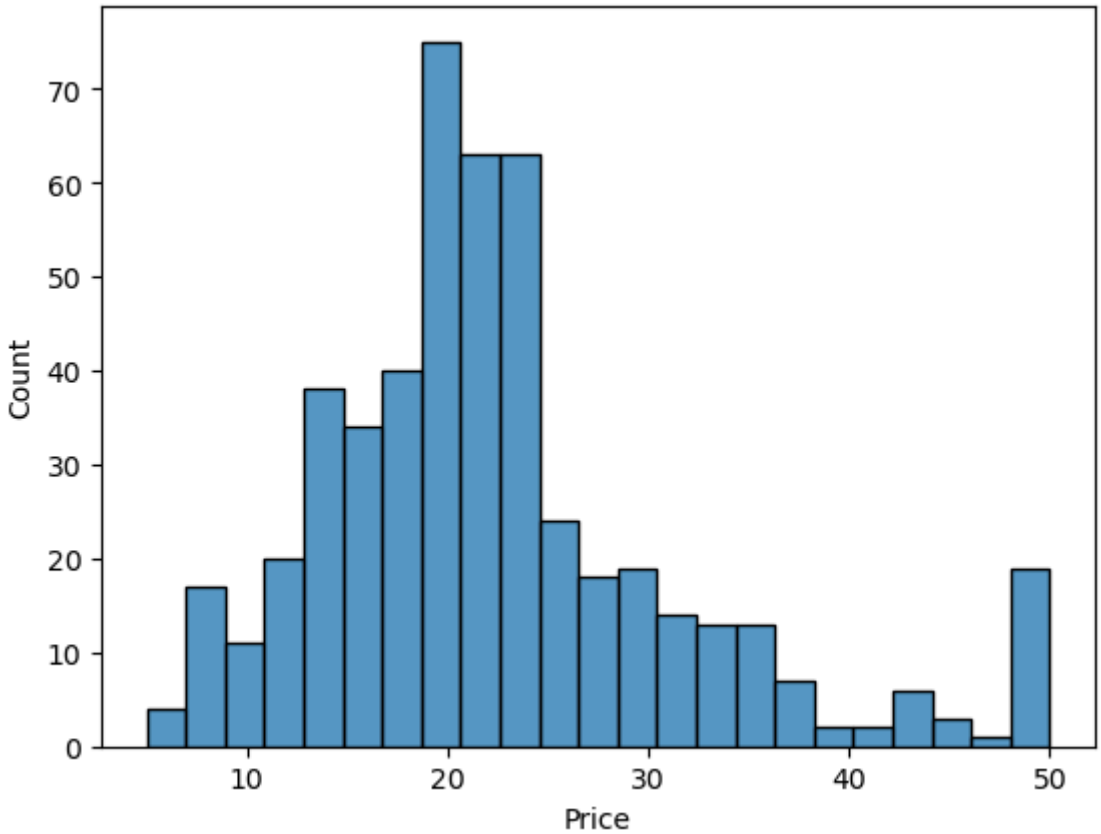
In [9]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 15 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  506 non-null    int64
 1   CRIM        506 non-null    float64
 2   ZN          506 non-null    float64
 3   INDUS       506 non-null    float64
 4   CHAS        506 non-null    float64
 5   NOX         506 non-null    float64
 6   RM          506 non-null    float64
 7   AGE         506 non-null    float64
 8   DIS         506 non-null    float64
 9   RAD         506 non-null    float64
 10  TAX         506 non-null    float64
 11  PTRATIO     506 non-null    float64
 12  B           506 non-null    float64
 13  LSTAT       506 non-null    float64
 14  Price       506 non-null    float64
dtypes: float64(14), int64(1)
memory usage: 59.4 KB
```

In [10]:
```
import seaborn as sns
```

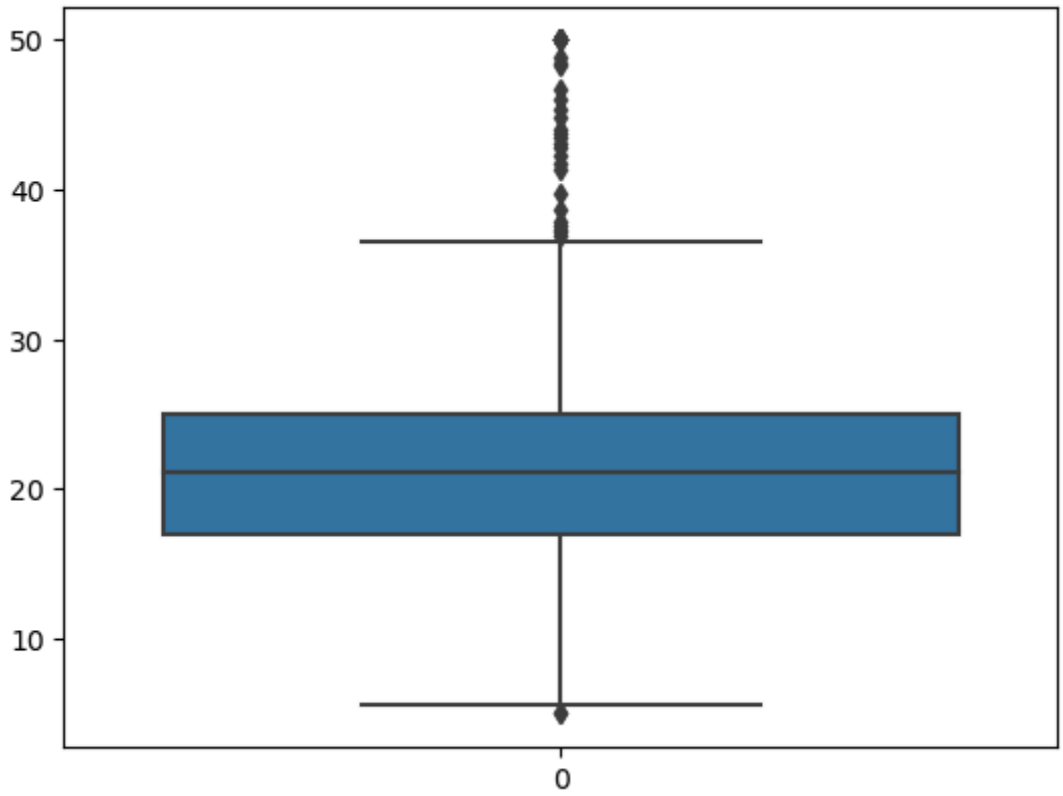
In [11]:
```
sns.histplot(data["Price"])

```

Out[11]:
```
<Axes: xlabel='Price', ylabel='Count'>
```



In [12]:
```
sns.boxplot(data["Price"])
```

Out[12]: <Axes: >



In [13]:
```
from sklearn.preprocessing import StandardScaler
# Split the data into input and output variables
X = data.drop('Price', axis=1)
y =data['Price']
# Scale the input features
scaler =StandardScaler()
X =scaler.fit_transform(X)
```

In [14]:
```
from sklearn.model_selection import train_test_split
# Splitthe data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
```

In [15]:
```
print('Training set shape:', X_train.shape, y_train.shape)
print('Testing set shape:', X_test.shape, y_test.shape)
```

```
Training set shape: (354, 14) (354,)
Testing set shape: (152, 14) (152,)
```

In [16]:
```
from keras.models import Sequential
from keras.layers import Dense, Dropout

```

In [17]:

```
# Define the model architecture
model = Sequential()
model.add(Dense(128,activation = 'relu',input_dim =14))
model.add(Dense(64,activation = 'relu'))
model.add(Dense(32,activation = 'relu'))
model.add(Dense(16,activation = 'relu'))
model.add(Dense(1))
```

```
C:\Users\STES\anaconda3\Lib\site-packages\keras\src\layers\core\dense.py:8
8: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a lay
er. When using Sequential models, prefer using an `Input(shape)` object as
the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [18]:

```
# Display the model summary
print(model.summary())
```

Model: "sequential"

Layer (type)	Output Shape
dense (Dense)	(None, 128)
dense_1 (Dense)	(None, 64)
dense_2 (Dense)	(None, 32)
dense_3 (Dense)	(None, 16)
dense_4 (Dense)	(None, 1)

Total params: 12,801 (50.00 KB)

Trainable params: 12,801 (50.00 KB)

Non-trainable params: 0 (0.00 B)

None

In [19]:

```

# Compile the model
model.compile(optimizer = 'adam',loss ='mean_squared_error',metrics=['m

x_val=X_train
y_val=y_train

```

In [20]:

```
history = model.fit(X_train,
                    y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - loss: 624.6559 - mae: 23.1586 - val_
loss: 622.1426 - val_mae: 23.1068
Epoch 2/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 622.1426 - mae: 23.1068 - va
l_loss: 620.0562 - val_mae: 23.0625
Epoch 3/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 620.0562 - mae: 23.0625 - va
l_loss: 618.4172 - val_mae: 23.0253
Epoch 4/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 618.4172 - mae: 23.0253 - va
l_loss: 617.0788 - val_mae: 22.9923
Epoch 5/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 617.0788 - mae: 22.9923 - va
l_loss: 615.6612 - val_mae: 22.9577
Epoch 6/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step - loss: 615.6611 - mae: 22.9577 - va
l_loss: 614.0786 - val_mae: 22.9198
Epoch 7/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 614.0786 - mae: 22.9198 - va
l_loss: 612.3089 - val_mae: 22.8784
Epoch 8/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 612.3089 - mae: 22.8784 - va
l_loss: 610.3096 - val_mae: 22.8320
Epoch 9/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 610.3096 - mae: 22.8320 - va
l_loss: 608.0159 - val_mae: 22.7791
Epoch 10/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 608.0159 - mae: 22.7791 - va
l_loss: 605.3791 - val_mae: 22.7185
Epoch 11/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 605.3790 - mae: 22.7185 - va
l_loss: 602.3395 - val_mae: 22.6489
Epoch 12/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 602.3395 - mae: 22.6489 - va
l_loss: 598.8791 - val_mae: 22.5700
Epoch 13/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 598.8791 - mae: 22.5700 - va
l_loss: 595.0938 - val_mae: 22.4838
Epoch 14/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 595.0938 - mae: 22.4838 - va
l_loss: 591.0748 - val_mae: 22.3923
Epoch 15/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 591.0748 - mae: 22.3923 - va
l_loss: 586.8221 - val_mae: 22.2952
Epoch 16/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 586.8221 - mae: 22.2952 - va
l_loss: 582.3458 - val_mae: 22.1927
Epoch 17/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 582.3458 - mae: 22.1927 - va
l_loss: 577.6431 - val_mae: 22.0842
Epoch 18/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 577.6431 - mae: 22.0842 - va
l_loss: 572.6659 - val_mae: 21.9685
Epoch 19/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - loss: 572.6659 - mae: 21.9685 - va
l_loss: 567.3625 - val_mae: 21.8447
Epoch 20/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - loss: 567.3625 - mae: 21.8447 - va
l_loss: 561.7023 - val_mae: 21.7116
```

In [22]:

```
results = model.evaluate(X_test, y_test)
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 471.2321 - mae: 20.0160
```

In [ ]:

```
```