In [1]:

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
```

In [2]:

```
from sklearn.datasets import load_boston

# Load the Boston housing dataset
boston_dataset = load_boston()

# Print the keys of the dataset
print(boston_dataset.keys())
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[2], line 1
----> 1 from sklearn.datasets import load_boston
      3 # Load the Boston housing dataset
      4 boston_dataset = load_boston()

File ~\anaconda3\lib\site-packages\sklearn\datasets\__init__.py:156, in __getattr__(nam
e)
    105 if name == "load_boston":
    106     msg = textwrap.dedent(
    107         """
    108         `load_boston` has been removed from scikit-learn since version 1.2.
   (...)
    154         """
    155     )
--> 156     raise ImportError(msg)
    157 try:
    158     return globals()[name]

ImportError: 
`load_boston` has been removed from scikit-learn since version 1.2.

The Boston housing prices dataset has an ethical problem: as
investigated in [1], the authors of this dataset engineered a
non-invertible variable "B" assuming that racial self-segregation had a
positive impact on house prices [2]. Furthermore the goal of the
research that led to the creation of this dataset was to study the
impact of air quality but it did not give adequate demonstration of the
validity of this assumption.

The scikit-learn maintainers therefore strongly discourage the use of
this dataset unless the purpose of the code is to study and educate
about ethical issues in data science and machine learning.

In this special case, you can fetch the dataset from the original
source::

    import pandas as pd
    import numpy as np

    data_url = "http://lib.stat.cmu.edu/datasets/boston"
    raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
    data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
    target = raw_df.values[1::2, 2]

Alternative datasets include the California housing dataset and the
Ames housing dataset. You can load the datasets as follows::

    from sklearn.datasets import fetch_california_housing
    housing = fetch_california_housing()

for the California housing dataset and::

    from sklearn.datasets import fetch_openml
    housing = fetch_openml(name="house_prices", as_frame=True)

for the Ames housing dataset.

[1] M Carlisle.
"Racist data destruction?"
<https://medium.com/@docintangible/racist-data-destruction-113e3eff54a8>

[2] Harrison Jr, David, and Daniel L. Rubinfeld.
```

```
"Hedonic housing prices and the demand for clean air."
Journal of environmental economics and management 5.1 (1978): 81-102.
<https://www.researchgate.net/publication/4974606_Hedonic_housing_prices_and_the_demand_
for_clean_air>
```

In [ ]:

```

```

In [3]:

```
dataset = pd.read_csv("C:\Users\omkar\Downloads\boston.csv")
```

```
  Cell In[3], line 1
    dataset = pd.read_csv("C:\Users\omkar\Downloads\boston.csv")
                                                               ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: t
runcated \UXXXXXXXX escape
```

In [4]:

```
dataset = pd.read_csv("C:/Users/omkar/Downloads/boston.csv")
```

In [5]:

```
dataset.keys()
```

Out[5]:

```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'MDEV'],
      dtype='object')
```

In [6]:

```
dataset.head()
```

Out[6]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MDEV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 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 | 24.0 |
| **1** | 0.02731 | 0.0 | 7.07 | 0.0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2.0 | 242.0 | 17.8 | 396.90 | 9.14 | 21.6 |
| **2** | 0.02729 | 0.0 | 7.07 | 0.0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2.0 | 242.0 | 17.8 | 392.83 | 4.03 | 34.7 |
| **3** | 0.03237 | 0.0 | 2.18 | 0.0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3.0 | 222.0 | 18.7 | 394.63 | 2.94 | 33.4 |
| **4** | 0.06905 | 0.0 | 2.18 | 0.0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3.0 | 222.0 | 18.7 | 396.90 | 5.33 | 36.2 |

In [7]:

```
dataset.describe()
```

Out[7]:

| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | R |
|---|---|---|---|---|---|---|---|---|---|
| **count** | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000 |
| **mean** | 3.593761 | 11.363636 | 11.136779 | 0.069170 | 0.554695 | 6.284634 | 68.574901 | 3.795043 | 9.549 |
| **std** | 8.596783 | 23.322453 | 6.860353 | 0.253994 | 0.115878 | 0.702617 | 28.148861 | 2.105710 | 8.707 |
| **min** | 0.006320 | 0.000000 | 0.460000 | 0.000000 | 0.385000 | 3.561000 | 2.900000 | 1.129600 | 1.000 |
| **25%** | 0.082045 | 0.000000 | 5.190000 | 0.000000 | 0.449000 | 5.885500 | 45.025000 | 2.100175 | 4.000 |
| **50%** | 0.256510 | 0.000000 | 9.690000 | 0.000000 | 0.538000 | 6.208500 | 77.500000 | 3.207450 | 5.000 |
| **75%** | 3.647423 | 12.500000 | 18.100000 | 0.000000 | 0.624000 | 6.623500 | 94.075000 | 5.188425 | 24.000 |
| **max** | 88.976200 | 100.000000 | 27.740000 | 1.000000 | 0.871000 | 8.780000 | 100.000000 | 12.126500 | 24.000 |

In [8]:

```
dataset.info()
```

Loading [MathJax]/extensions/Safe.js

```
<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  MDEV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [9]:
```
dataset.isnull().sum()
```

Out[9]:
```
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
MDEV       0
dtype: int64
```

In [10]:
```
dataset = dataset.fillna(dataset.mean())
```

In [11]:
```
dataset.isnull().sum()
```

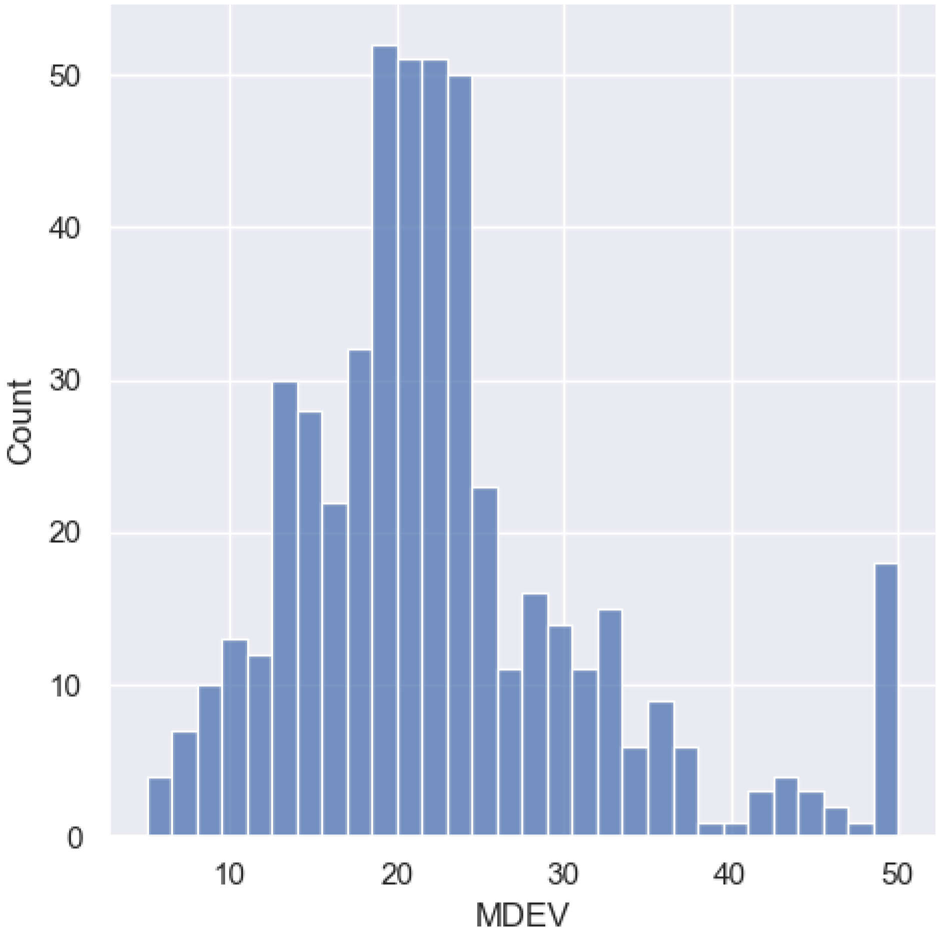
Out[11]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MDEV       0
dtype: int64
```

In [12]:
```
#Plot the distribution of 'MDEV' = median value of owner-occupied homes in thousands of
```

In [13]:
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.displot(dataset['MDEV'],bins=30);
```

Loading [MathJax]/extensions/Safe.js

```
plt.show()
```



In [14]:

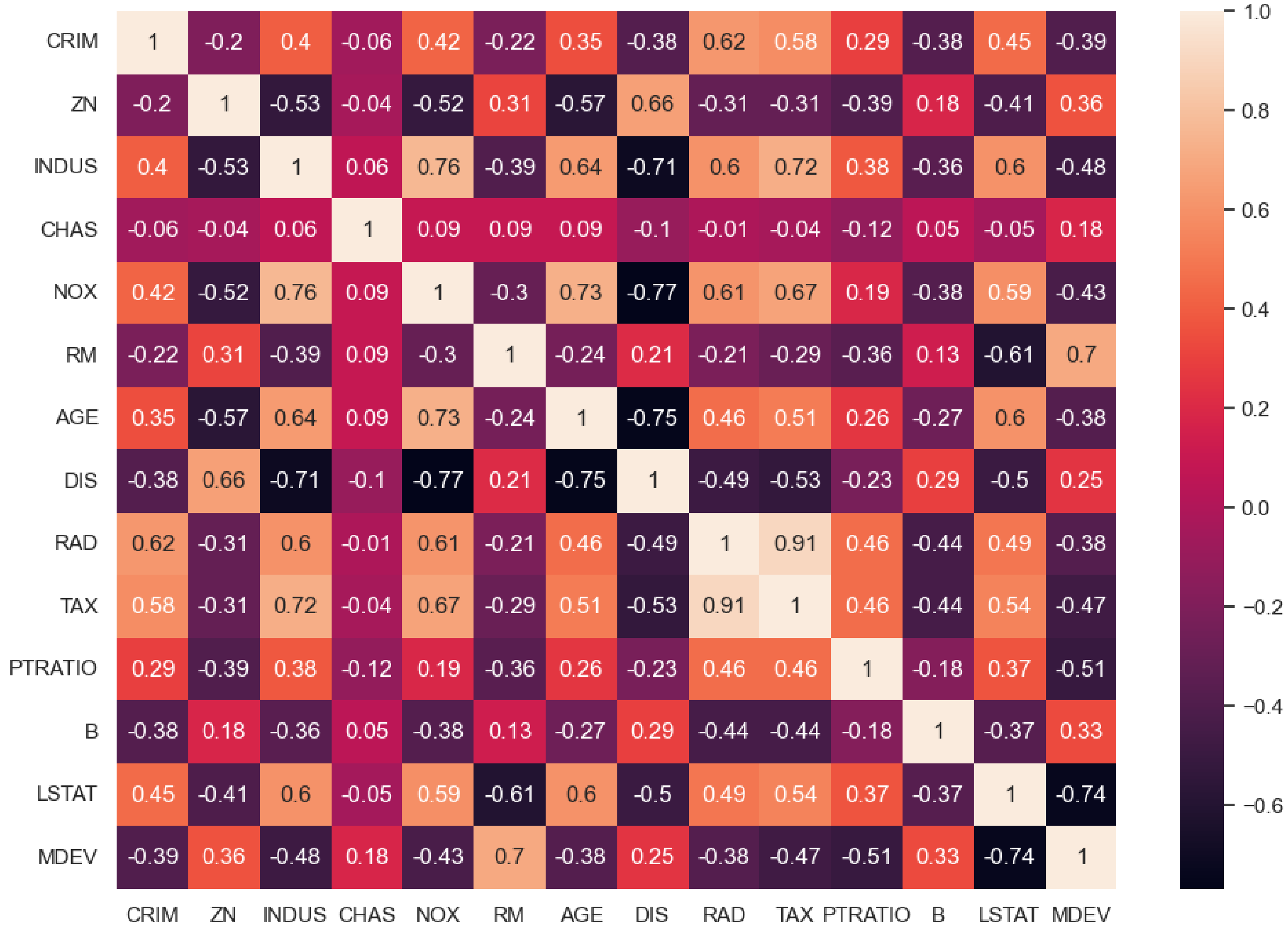
```
#correlation matrix
```

In [15]:

```
correlation_matrix = dataset.corr().round(2)
sns.heatmap(data=correlation_matrix, annot=True)
```

Out[15]:

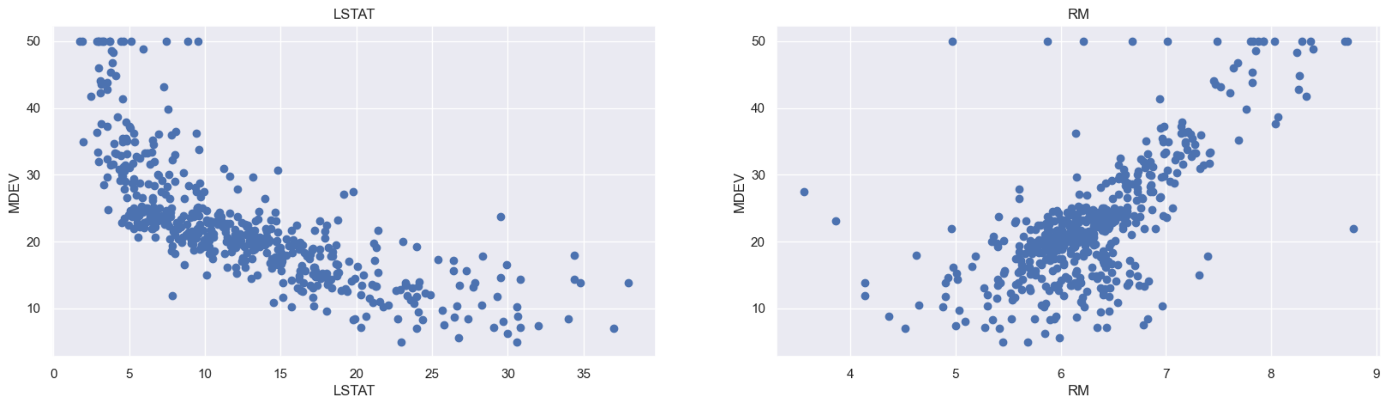
```
<Axes: >
```



| | CRIM | ZN | INDUS | CHAS | NOX | RM | AGE | DIS | RAD | TAX | PTRATIO | B | LSTAT | MDEV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRIM | 1 | -0.2 | 0.4 | -0.06 | 0.42 | -0.22 | 0.35 | -0.38 | 0.62 | 0.58 | 0.29 | -0.38 | 0.45 | -0.39 |
| ZN | -0.2 | 1 | -0.53 | -0.04 | -0.52 | 0.31 | -0.57 | 0.66 | -0.31 | -0.31 | -0.39 | 0.18 | -0.41 | 0.36 |
| INDUS | 0.4 | -0.53 | 1 | 0.06 | 0.76 | -0.39 | 0.64 | -0.71 | 0.6 | 0.72 | 0.38 | -0.36 | 0.6 | -0.48 |
| CHAS | -0.06 | -0.04 | 0.06 | 1 | 0.09 | 0.09 | 0.09 | -0.1 | -0.01 | -0.04 | -0.12 | 0.05 | -0.05 | 0.18 |
| NOX | 0.42 | -0.52 | 0.76 | 0.09 | 1 | -0.3 | 0.73 | -0.77 | 0.61 | 0.67 | 0.19 | -0.38 | 0.59 | -0.43 |
| RM | -0.22 | 0.31 | -0.39 | 0.09 | -0.3 | 1 | -0.24 | 0.21 | -0.21 | -0.29 | -0.36 | 0.13 | -0.61 | 0.7 |
| AGE | 0.35 | -0.57 | 0.64 | 0.09 | 0.73 | -0.24 | 1 | -0.75 | 0.46 | 0.51 | 0.26 | -0.27 | 0.6 | -0.38 |
| DIS | -0.38 | 0.66 | -0.71 | -0.1 | -0.77 | 0.21 | -0.75 | 1 | -0.49 | -0.53 | -0.23 | 0.29 | -0.5 | 0.25 |
| RAD | 0.62 | -0.31 | 0.6 | -0.01 | 0.61 | -0.21 | 0.46 | -0.49 | 1 | 0.91 | 0.46 | -0.44 | 0.49 | -0.38 |
| TAX | 0.58 | -0.31 | 0.72 | -0.04 | 0.67 | -0.29 | 0.51 | -0.53 | 0.91 | 1 | 0.46 | -0.44 | 0.54 | -0.47 |
| PTRATIO | 0.29 | -0.39 | 0.38 | -0.12 | 0.19 | -0.36 | 0.26 | -0.23 | 0.46 | 0.46 | 1 | -0.18 | 0.37 | -0.51 |
| B | -0.38 | 0.18 | -0.36 | 0.05 | -0.38 | 0.13 | -0.27 | 0.29 | -0.44 | -0.44 | -0.18 | 1 | -0.37 | 0.33 |
| LSTAT | 0.45 | -0.41 | 0.6 | -0.05 | 0.59 | -0.61 | 0.6 | -0.5 | 0.49 | 0.54 | 0.37 | -0.37 | 1 | -0.74 |
| MDEV | -0.39 | 0.36 | -0.48 | 0.18 | -0.43 | 0.7 | -0.38 | 0.25 | -0.38 | -0.47 | -0.51 | 0.33 | -0.74 | 1 |

In [16]:

```python
plt.figure(figsize=(20, 5))

# Define the features and the target variable
features = ['LSTAT', 'RM']
target = dataset['MDEV']

# Loop through each feature
for i, col in enumerate(features):
    # Create subplots
    plt.subplot(1, len(features), i + 1)

    # Define x and y values for the scatter plot
    x = dataset[col]
    y = target

    # Plot the scatter plot
    plt.scatter(x, y, marker='o')

    # Set title, xlabel, and ylabel for the subplot
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('MDEV')

plt.show()
```



In [17]:

```
import numpy as np
x = pd.DataFrame(np.c_[dataset['LSTAT'], dataset['RM']], columns=['LSTAT', 'RM'])
y = dataset['MDEV']
```

In [18]:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=5)

print("Training set shapes:")
print("x_train:", x_train.shape)
print("y_train:", y_train.shape)
print("Testing set shapes:")
print("x_test:", x_test.shape)
print("y_test:", y_test.shape)
```

```
Training set shapes:
x_train: (404, 2)
y_train: (404,)
Testing set shapes:
x_test: (102, 2)
y_test: (102,)
```

In [19]:

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()

model.fit(x_train,y_train)
```

Out[19]:

```
▼ LinearRegression
LinearRegression()
```

In [20]:

```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import numpy as np

y_pred = model.predict(x_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

r2 = r2_score(y_test, y_pred)

print("Model performance for testing set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
```

```
Model performance for testing set
--------------------------------------
RMSE is 5.137400784702911
                         .6628996975186953
```

Loading [MathJax]/extensions/Safe.js

In [21]:

```
# Predicting selling price

sample_data = [[6.89, 9.939]]

price = model.predict(sample_data)

print("Predicted selling price for house: ${:,.2f}".format(price[0]))
```

```
Predicted selling price for house: $43.41
```

```
C:\Users\omkar\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X does not
have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [22]:

```
sample_data_df = pd.DataFrame(sample_data, columns=['LSTAT', 'RM'])
price = model.predict(sample_data_df)
print("Predicted selling price for house: ${:,.2f}".format(price[0]))
```

```
Predicted selling price for house: $43.41
```

In [ ]: