```
# loading all libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
# When using the 'inline' backend, your matplotlib graphs will be included in your notebook, next to the code.
%matplotlib inline


# load the housing data from the scikit-learn library
from sklearn.datasets import load_boston


boston_dataset = load_boston()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be re

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    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 (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) 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.

  warnings.warn(msg, category=FutureWarning)
```

```
# We print the value of the boston_dataset to understand what it contains.
print(boston_dataset.keys())
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

```
# load dataset into pandas DataFrame and print dataset (first 5 values)
df = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
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
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
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
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
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

```
# as price column is missing need to create column of target values in dataframe
df['Price'] = boston_dataset.target
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
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

```
# describe the boston dataset
df.describe()
```

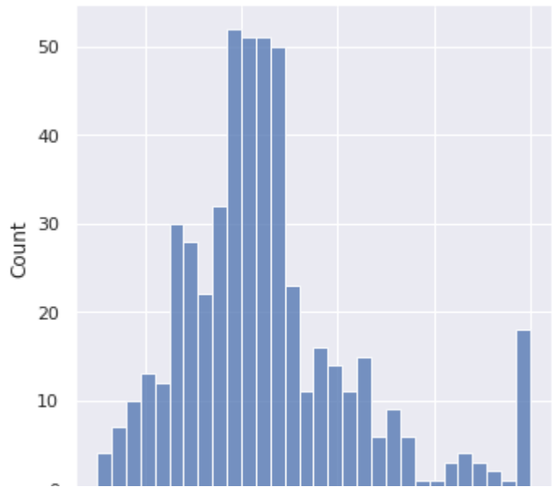
```
# info of boston dataset
df.info()
```

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

```
# checking the missing values using isnull()
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
Price      0
dtype: int64
```

```
# setting the output figure size
sns.set(rc={'figure.figsize':(11.7, 8.27)})
# plotting the target value Price for visualsing through histogram
sns.displot(df['Price'], bins=30)
plt.show()
```



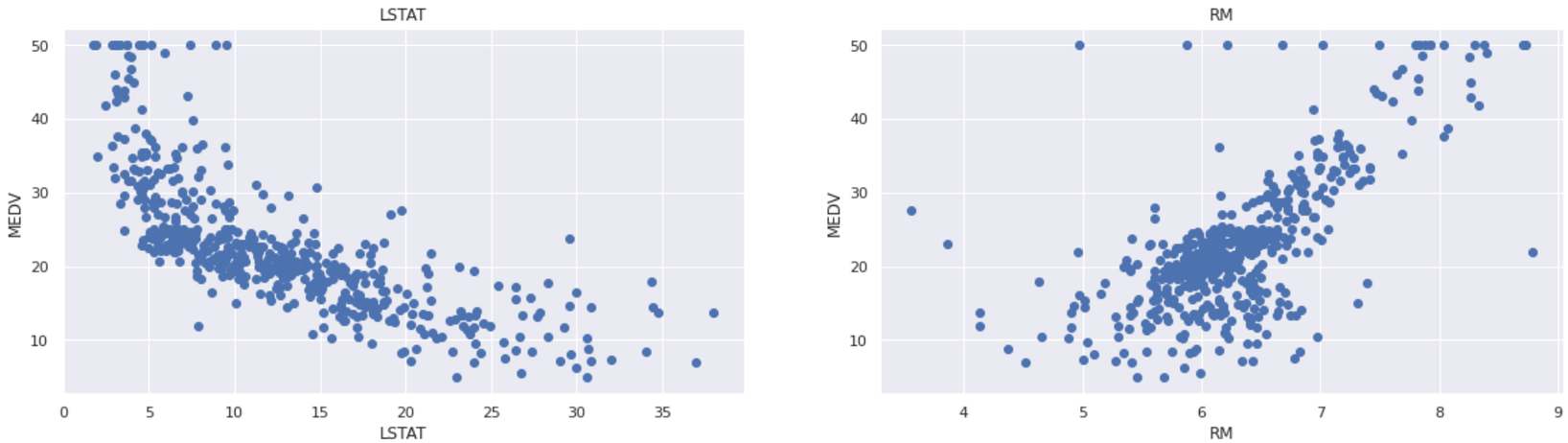

```
# correlation matrix to measure the linear relationships between the variables.
correlation_matrix = df.corr().round(2)
# annot - true to print value inside square
# use the heatmap function from the seaborn library to plot the matrix
sns.heatmap(data=correlation_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93623a4590>
```


```
# By observing correlation matrix we can see that RM has a strong positive correlation
# with Price (0.7) and LSTAT has a high negative correlation with Price (-0.74)
```

```
# RM and LSTAT are used as features
plt.figure(figsize=(20, 5))
features = ['LSTAT', 'RM']
target = df['Price']

for i, col in enumerate(features):
  plt.subplot(1, len(features), i+1)
  x = df[col]
  y = target
  plt.scatter(x, y, marker='o')
  plt.title(col)
  plt.xlabel(col)
  plt.ylabel('MEDV')
```




```
# We concatenate the LSTAT and RM columns using np.c_ provided by the numpy library
import numpy as np
X = pd.DataFrame(np.c_[df['LSTAT'], df['RM']], columns=['LSTAT','RM'])
Y = df['Price']

# We train the model with 80% of the samples and test with the remaining 20%
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 the sizes of our training and test set to verify if the splitting is proper
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(404, 2)
(102, 2)
(404,)
(102,)
```

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
```

```
LinearRegression()
```

```
# model evaluation
from sklearn.metrics import mean_squared_error, r2_score
y_pred = model.predict(X_test)
# root mean squared error
rmse = (np.sqrt(mean_squared_error(Y_test, y_pred)))
r2 = r2_score(Y_test, y_pred)
print('the model performance for testing set')
print('--------------------------------------')
print(f'RMSE is {rmse}')
print(f'R2 score is {r2}')
```

```
the model performance for testing set
--------------------------------------
RMSE is 5.137400784702911
R2 score is 0.6628996975186952
```

```
# produce matrix for sample data
sample_data = [[6.89, 9.939]]
price = model.predict(sample_data)
print(f"predicted selling price for house : {price[0]:.2f}")
```

```
predicted selling price for house : 43.41
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  "X does not have valid feature names, but"
```