

```
import numpy as np
import matplotlib.pyplot as plt
```

```
import pandas as pd
import seaborn as sns
```

```
%matplotlib inline
```

+ Code

+ Text

```
boston_dataset = pd.read_csv("/content/BostonHousing.csv")
```

```
boston_dataset.head()
```

	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

```
boston_dataset.isnull().sum()
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-1-c8d9d07afc9a> in <cell line: 1>()
----> 1 boston_dataset.isnull().sum()

NameError: name 'boston_dataset' is not defined
```

```
correlation_matrix = boston_dataset.corr().round(2)
# annot = True to print the values inside the square
sns.heatmap(data=correlation_matrix, annot=True)
```

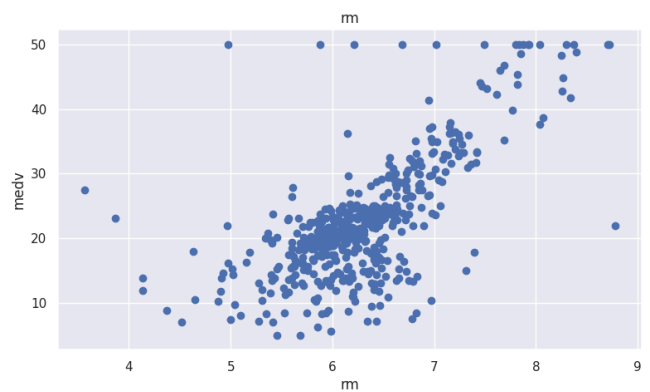
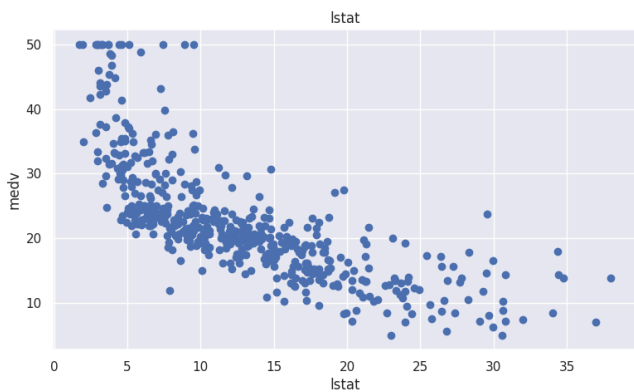
```
-----
NameError                                Traceback (most recent call last)
<ipython-input-2-d9dd4820afee> in <cell line: 1>()
----> 1 correlation_matrix = boston_dataset.corr().round(2)
      2 # annot = True to print the values inside the square
      3 sns.heatmap(data=correlation_matrix, annot=True)

NameError: name 'boston_dataset' is not defined
```

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

```
features = ['lstat', 'rm']
target = boston_dataset['medv']
```

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



```
X = pd.DataFrame(np.c_[boston_dataset['lstat'], boston_dataset['rm']], columns = ['lstat','rm'])
Y = boston_dataset['medv']
```

```
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(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
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
```

```
lin_model = LinearRegression()
lin_model.fit(X_train, Y_train)
```

```
▼ LinearRegression
LinearRegression()
```

```
model = sm.OLS(Y_train, X_train).fit()
```

```
print(model.summary())
```

```

=====
                        OLS Regression Results
=====
Dep. Variable:          medv      R-squared (uncentered):      0.947
Model:                  OLS      Adj. R-squared (uncentered):    0.947
Method:                 Least Squares      F-statistic:          3581.
Date:                   Fri, 21 Apr 2023      Prob (F-statistic):    6.67e-257
Time:                   04:07:20      Log-Likelihood:       -1272.2
No. Observations:       404      AIC:                  2548.
Df Residuals:           402      BIC:                  2556.
Df Model:                2
Covariance Type:        nonrobust
=====

```

	coef	std err	t	P> t	[0.025	0.975]
lstat	-0.6911	0.036	-19.367	0.000	-0.761	-0.621
rm	4.9699	0.081	61.521	0.000	4.811	5.129

```

=====
Omnibus:                 121.894      Durbin-Watson:           2.063
Prob(Omnibus):            0.000      Jarque-Bera (JB):         389.671
Skew:                     1.370      Prob(JB):                 2.42e-85
Kurtosis:                 6.954      Cond. No.                  4.70
=====

```

Notes:

- [1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
- [2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

