Regresao LInear

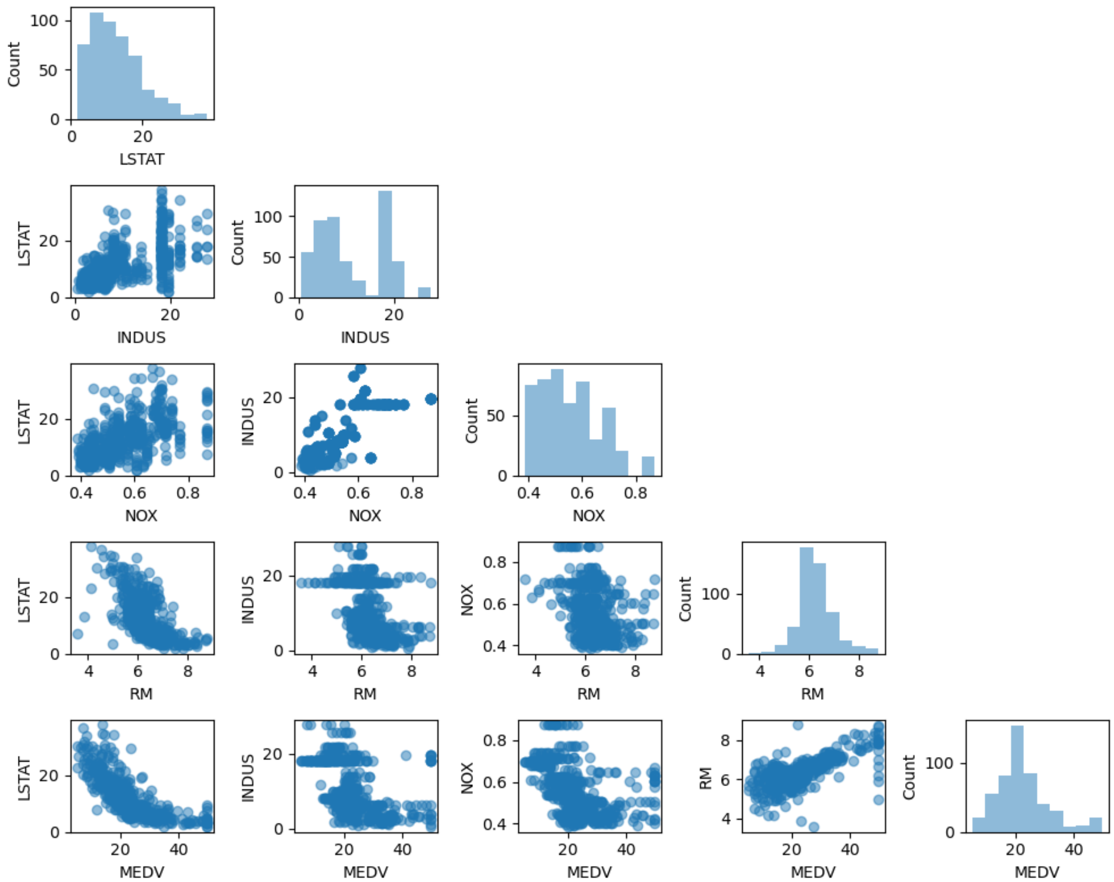
September 28, 2022

[5]:
```python
import pandas as pd

df = pd.read_csv('housing.data.txt',header=None, sep='\s+')

df.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS',
              'NOX', 'RM', 'AGE', 'DIS', 'RAD',
              'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
df.head()
```

[5]:
```
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
```

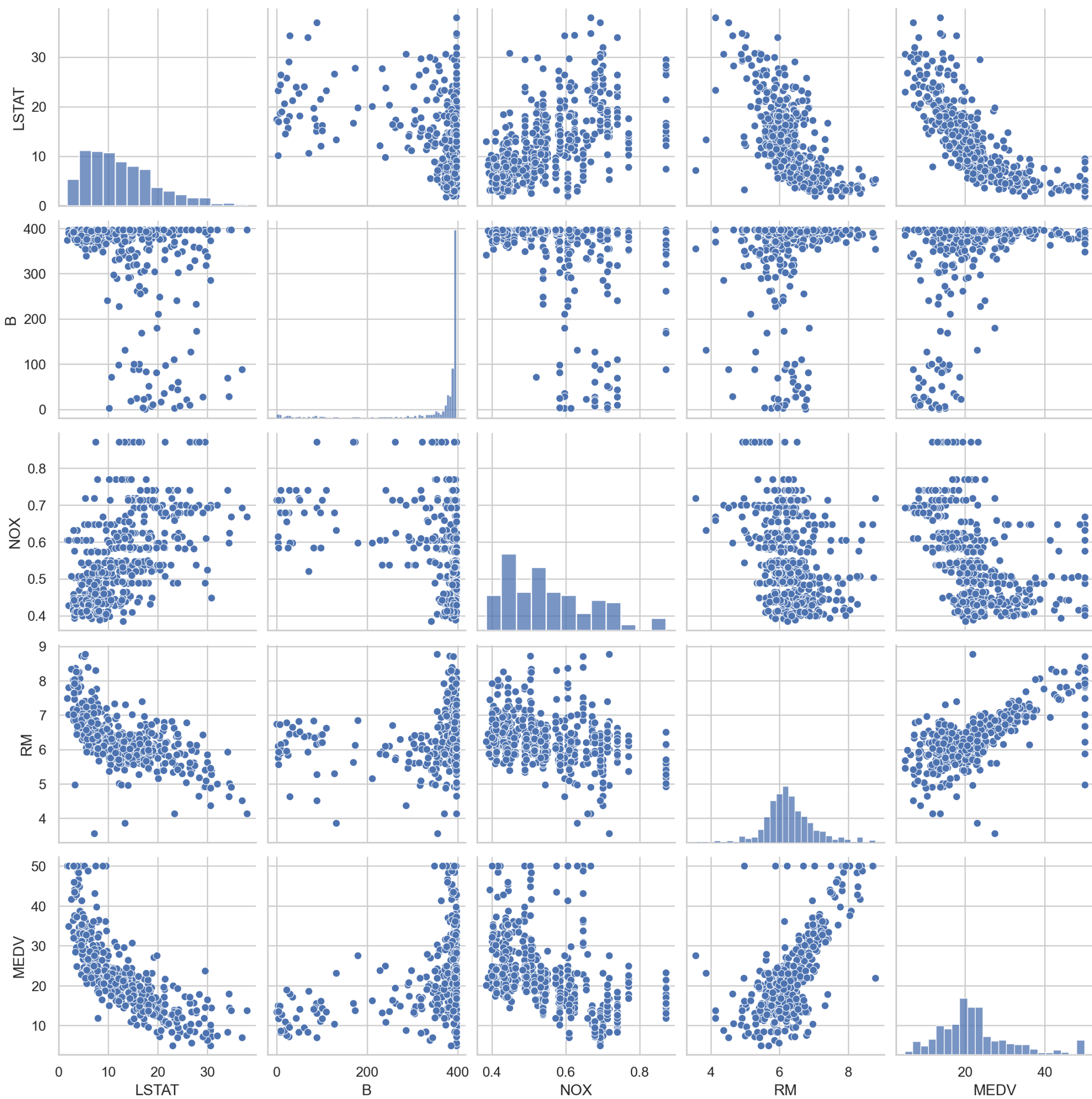
[6]:
```python
import matplotlib.pyplot as plt
from mlxtend.plotting import scatterplotmatrix
```

[7]:
```python
cols = ['LSTAT', 'INDUS', 'NOX', 'RM', 'MEDV']

scatterplotmatrix(df[cols].values, figsize=(10, 8),
                  names=cols, alpha=0.5)
plt.tight_layout()
#plt.savefig('images/10_03.png', dpi=300)
plt.show()
```

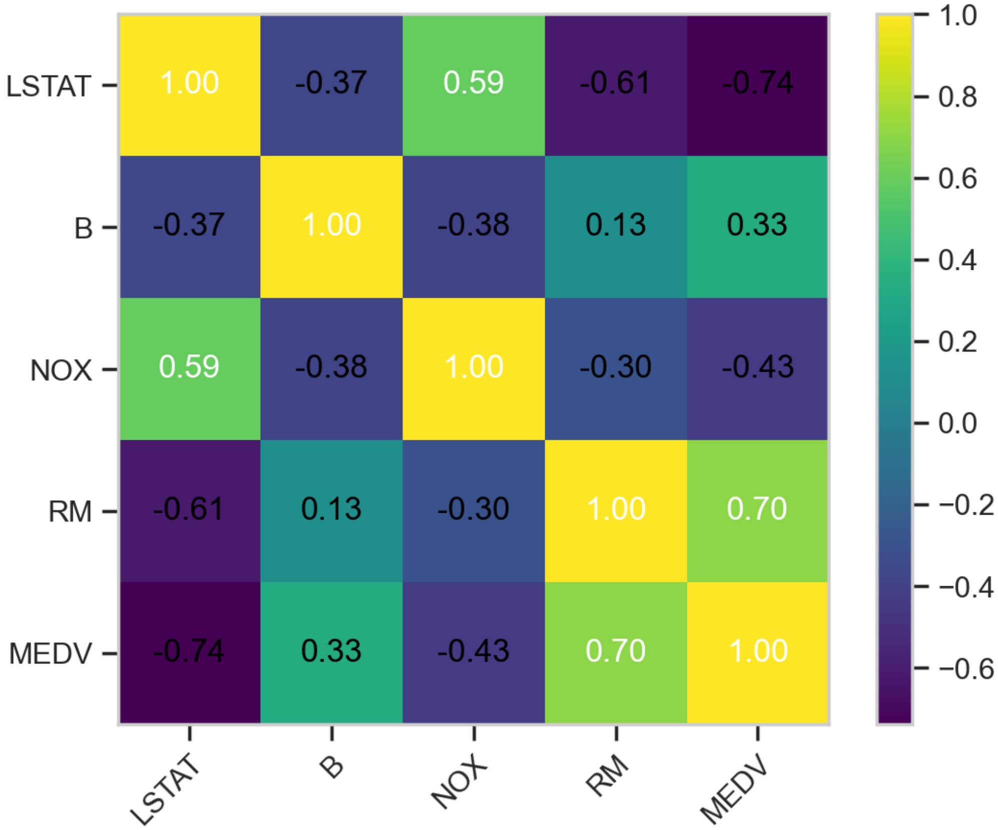


[8]:
```
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
%config InlineBackend.figure_format = 'retina'

sns.set(style='whitegrid', context='notebook')

cols = ['LSTAT', 'B', 'NOX', 'RM', 'MEDV']
sns.pairplot(df[cols], size=2.5)
plt.savefig('scatter.png', dpi=300)

plt.show()
```

```
/usr/lib/python3/dist-packages/seaborn/axisgrid.py:2089: UserWarning: The `size`
parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



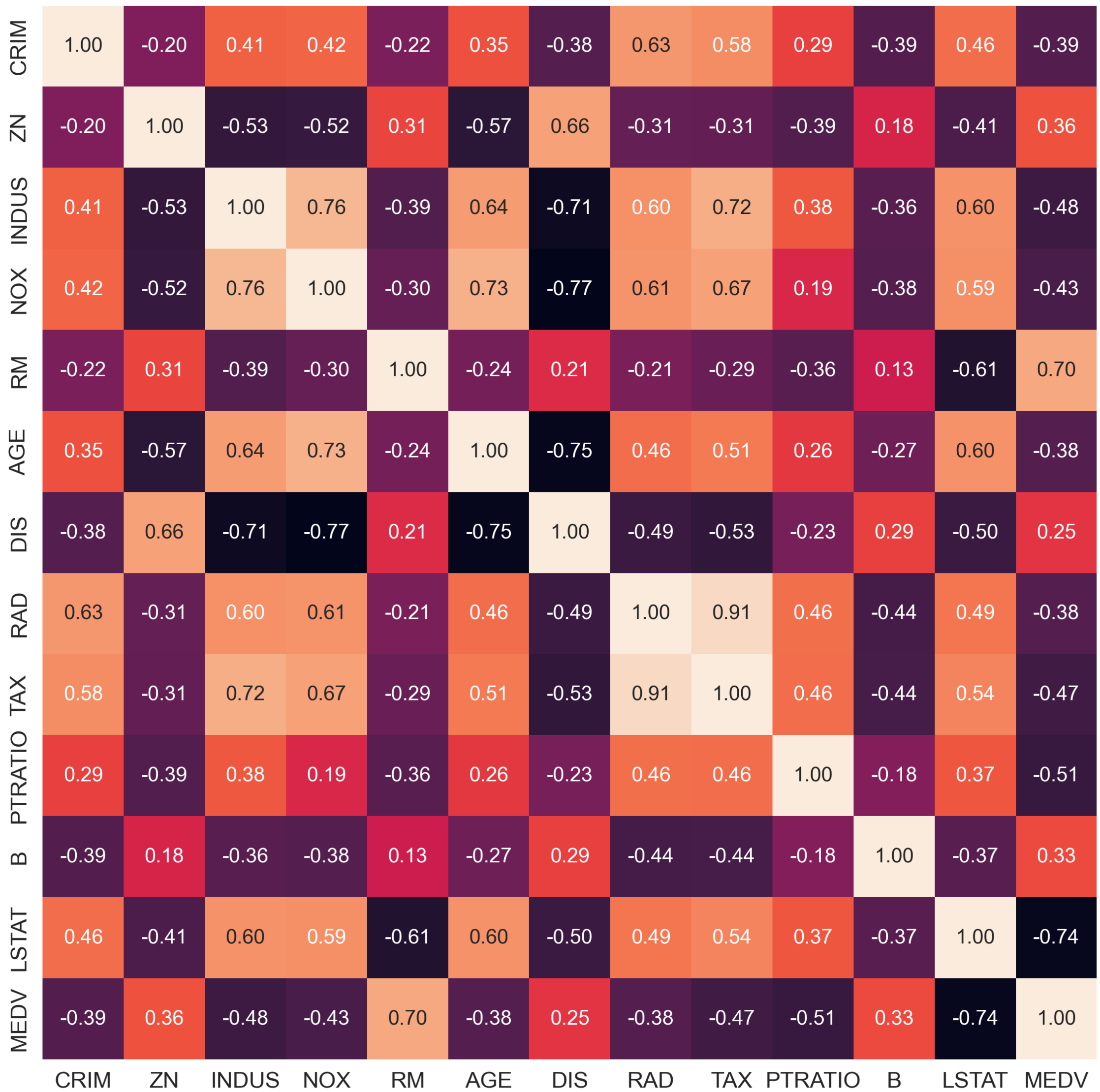
[9]:
```
import numpy as np
from mlxtend.plotting import heatmap


cm = np.corrcoef(df[cols].values.T)
hm = heatmap(cm, row_names=cols, column_names=cols)

# plt.savefig('images/10_04.png', dpi=300)
plt.show()
```



[10]:

```
import numpy as np

corr_cols = ['CRIM', 'ZN', 'INDUS',
             'NOX', 'RM', 'AGE', 'DIS', 'RAD',
             'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

cm = np.corrcoef(df[corr_cols].values.T)
sns.set(font_scale=1.5)

fig, ax = plt.subplots(figsize=(14, 14))

hm = sns.heatmap(cm,
                 ax=ax,
                 cbar=False,
                 annot=True,
                 square=True,
                 fmt='.2f',
                 annot_kws={'size': 15},
                 yticklabels=corr_cols,
                 xticklabels=corr_cols)
```

```
# plt.savefig('corr_mat.png', dpi=300)

plt.show()
```



0.1 Implementando um modelo de regressão linear de mínimos quadrados comum

…

0.1.1 Resolvendo a regressão para parâmetros de regressão com gradiente descendente

[11]:
```python
import numpy as np
```

[12]:
```python
class LinearRegressionGD(object):

    def __init__(self, eta=0.001, n_iter=20):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self, X, y):
        self.w_ = np.zeros(1 + X.shape[1])
        self.cost_ = []

        for i in range(self.n_iter):
            output = self.net_input(X)
            errors = (y - output)
            self.w_[1:] += self.eta * X.T.dot(errors)
            self.w_[0] += self.eta * errors.sum()
            cost = (errors**2).sum() / 2.0
            self.cost_.append(cost)
        return self

    def net_input(self, X):
        return np.dot(X, self.w_[1:]) + self.w_[0]

    def predict(self, X):
        return self.net_input(X)
```

[13]:
```python
X = df[['RM']].values
y = df['MEDV'].values
```

[14]:
```python
X
```

[14]:
```
array([[6.575],
       [6.421],
       [7.185],
       [6.998],
       [7.147],
       [6.43 ],
       [6.012],
       [6.172],
       [5.631],
       [6.004],
       [6.377],
       [6.009],
       [5.889],
```

```
[5.949],
[6.096],
[5.834],
[5.935],
[5.99 ],
[5.456],
[5.727],
[5.57 ],
[5.965],
[6.142],
[5.813],
[5.924],
[5.599],
[5.813],
[6.047],
[6.495],
[6.674],
[5.713],
[6.072],
[5.95 ],
[5.701],
[6.096],
[5.933],
[5.841],
[5.85 ],
[5.966],
[6.595],
[7.024],
[6.77 ],
[6.169],
[6.211],
[6.069],
[5.682],
[5.786],
[6.03 ],
[5.399],
[5.602],
[5.963],
[6.115],
[6.511],
[5.998],
[5.888],
[7.249],
[6.383],
[6.816],
[6.145],
[5.927],
```

```
[5.741],
[5.966],
[6.456],
[6.762],
[7.104],
[6.29 ],
[5.787],
[5.878],
[5.594],
[5.885],
[6.417],
[5.961],
[6.065],
[6.245],
[6.273],
[6.286],
[6.279],
[6.14 ],
[6.232],
[5.874],
[6.727],
[6.619],
[6.302],
[6.167],
[6.389],
[6.63 ],
[6.015],
[6.121],
[7.007],
[7.079],
[6.417],
[6.405],
[6.442],
[6.211],
[6.249],
[6.625],
[6.163],
[8.069],
[7.82 ],
[7.416],
[6.727],
[6.781],
[6.405],
[6.137],
[6.167],
[5.851],
[5.836],
```

```
[6.127],
[6.474],
[6.229],
[6.195],
[6.715],
[5.913],
[6.092],
[6.254],
[5.928],
[6.176],
[6.021],
[5.872],
[5.731],
[5.87 ],
[6.004],
[5.961],
[5.856],
[5.879],
[5.986],
[5.613],
[5.693],
[6.431],
[5.637],
[6.458],
[6.326],
[6.372],
[5.822],
[5.757],
[6.335],
[5.942],
[6.454],
[5.857],
[6.151],
[6.174],
[5.019],
[5.403],
[5.468],
[4.903],
[6.13 ],
[5.628],
[4.926],
[5.186],
[5.597],
[6.122],
[5.404],
[5.012],
[5.709],
```

```
[6.129],
[6.152],
[5.272],
[6.943],
[6.066],
[6.51 ],
[6.25 ],
[7.489],
[7.802],
[8.375],
[5.854],
[6.101],
[7.929],
[5.877],
[6.319],
[6.402],
[5.875],
[5.88 ],
[5.572],
[6.416],
[5.859],
[6.546],
[6.02 ],
[6.315],
[6.86 ],
[6.98 ],
[7.765],
[6.144],
[7.155],
[6.563],
[5.604],
[6.153],
[7.831],
[6.782],
[6.556],
[7.185],
[6.951],
[6.739],
[7.178],
[6.8  ],
[6.604],
[7.875],
[7.287],
[7.107],
[7.274],
[6.975],
[7.135],
```

```
[6.162],
[7.61 ],
[7.853],
[8.034],
[5.891],
[6.326],
[5.783],
[6.064],
[5.344],
[5.96 ],
[5.404],
[5.807],
[6.375],
[5.412],
[6.182],
[5.888],
[6.642],
[5.951],
[6.373],
[6.951],
[6.164],
[6.879],
[6.618],
[8.266],
[8.725],
[8.04 ],
[7.163],
[7.686],
[6.552],
[5.981],
[7.412],
[8.337],
[8.247],
[6.726],
[6.086],
[6.631],
[7.358],
[6.481],
[6.606],
[6.897],
[6.095],
[6.358],
[6.393],
[5.593],
[5.605],
[6.108],
[6.226],
```

```
[6.433],
[6.718],
[6.487],
[6.438],
[6.957],
[8.259],
[6.108],
[5.876],
[7.454],
[8.704],
[7.333],
[6.842],
[7.203],
[7.52 ],
[8.398],
[7.327],
[7.206],
[5.56 ],
[7.014],
[8.297],
[7.47 ],
[5.92 ],
[5.856],
[6.24 ],
[6.538],
[7.691],
[6.758],
[6.854],
[7.267],
[6.826],
[6.482],
[6.812],
[7.82 ],
[6.968],
[7.645],
[7.923],
[7.088],
[6.453],
[6.23 ],
[6.209],
[6.315],
[6.565],
[6.861],
[7.148],
[6.63 ],
[6.127],
[6.009],
```

```
[6.678],
[6.549],
[5.79 ],
[6.345],
[7.041],
[6.871],
[6.59 ],
[6.495],
[6.982],
[7.236],
[6.616],
[7.42 ],
[6.849],
[6.635],
[5.972],
[4.973],
[6.122],
[6.023],
[6.266],
[6.567],
[5.705],
[5.914],
[5.782],
[6.382],
[6.113],
[6.426],
[6.376],
[6.041],
[5.708],
[6.415],
[6.431],
[6.312],
[6.083],
[5.868],
[6.333],
[6.144],
[5.706],
[6.031],
[6.316],
[6.31 ],
[6.037],
[5.869],
[5.895],
[6.059],
[5.985],
[5.968],
[7.241],
```

```
[6.54 ],
[6.696],
[6.874],
[6.014],
[5.898],
[6.516],
[6.635],
[6.939],
[6.49 ],
[6.579],
[5.884],
[6.728],
[5.663],
[5.936],
[6.212],
[6.395],
[6.127],
[6.112],
[6.398],
[6.251],
[5.362],
[5.803],
[8.78 ],
[3.561],
[4.963],
[3.863],
[4.97 ],
[6.683],
[7.016],
[6.216],
[5.875],
[4.906],
[4.138],
[7.313],
[6.649],
[6.794],
[6.38 ],
[6.223],
[6.968],
[6.545],
[5.536],
[5.52 ],
[4.368],
[5.277],
[4.652],
[5.   ],
[4.88 ],
```

```
[5.39 ],
[5.713],
[6.051],
[5.036],
[6.193],
[5.887],
[6.471],
[6.405],
[5.747],
[5.453],
[5.852],
[5.987],
[6.343],
[6.404],
[5.349],
[5.531],
[5.683],
[4.138],
[5.608],
[5.617],
[6.852],
[5.757],
[6.657],
[4.628],
[5.155],
[4.519],
[6.434],
[6.782],
[5.304],
[5.957],
[6.824],
[6.411],
[6.006],
[5.648],
[6.103],
[5.565],
[5.896],
[5.837],
[6.202],
[6.193],
[6.38 ],
[6.348],
[6.833],
[6.425],
[6.436],
[6.208],
[6.629],
```

```
[6.461],
[6.152],
[5.935],
[5.627],
[5.818],
[6.406],
[6.219],
[6.485],
[5.854],
[6.459],
[6.341],
[6.251],
[6.185],
[6.417],
[6.749],
[6.655],
[6.297],
[7.393],
[6.728],
[6.525],
[5.976],
[5.936],
[6.301],
[6.081],
[6.701],
[6.376],
[6.317],
[6.513],
[6.209],
[5.759],
[5.952],
[6.003],
[5.926],
[5.713],
[6.167],
[6.229],
[6.437],
[6.98 ],
[5.427],
[6.162],
[6.484],
[5.304],
[6.185],
[6.229],
[6.242],
[6.75 ],
[7.061],
```

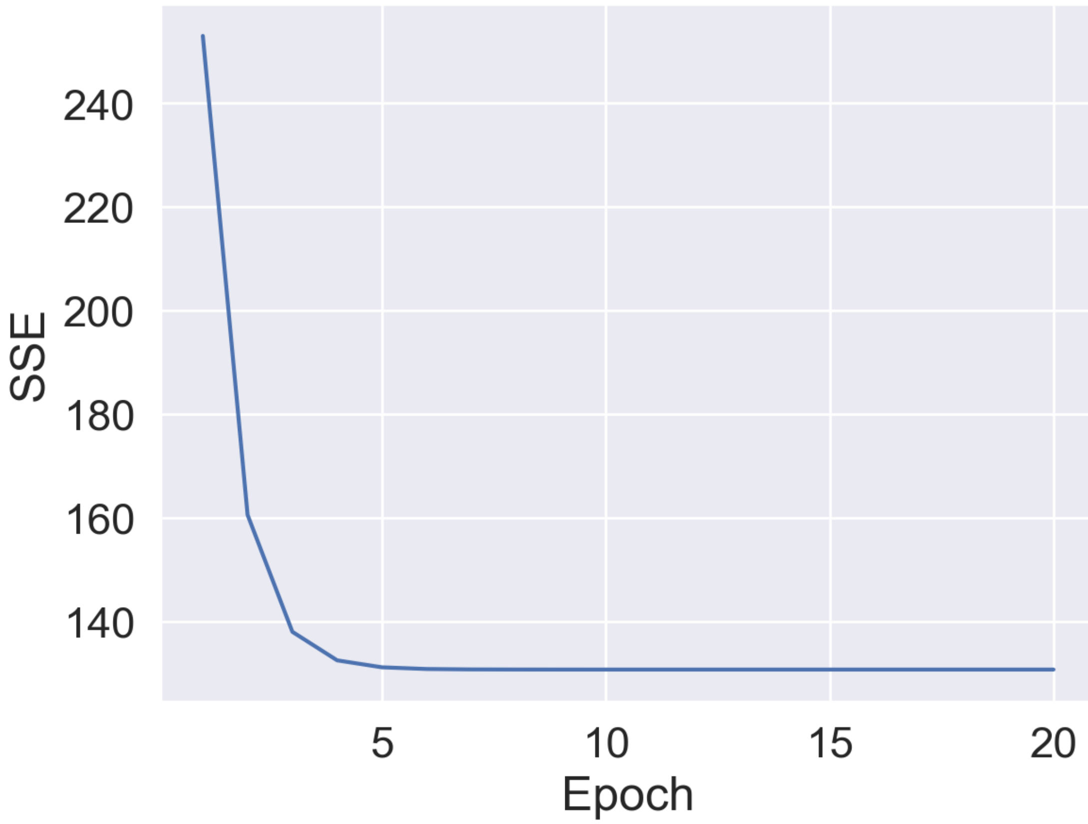
```
       [5.762],
       [5.871],
       [6.312],
       [6.114],
       [5.905],
       [5.454],
       [5.414],
       [5.093],
       [5.983],
       [5.983],
       [5.707],
       [5.926],
       [5.67 ],
       [5.39 ],
       [5.794],
       [6.019],
       [5.569],
       [6.027],
       [6.593],
       [6.12 ],
       [6.976],
       [6.794],
       [6.03 ]])
```

[15]:
```
from sklearn.preprocessing import StandardScaler


sc_x = StandardScaler()
sc_y = StandardScaler()
X_std = sc_x.fit_transform(X)
y_std = sc_y.fit_transform(y[:,np.newaxis]).flatten()
```

[16]:
```
lr = LinearRegressionGD()
lr.fit(X_std, y_std)
```

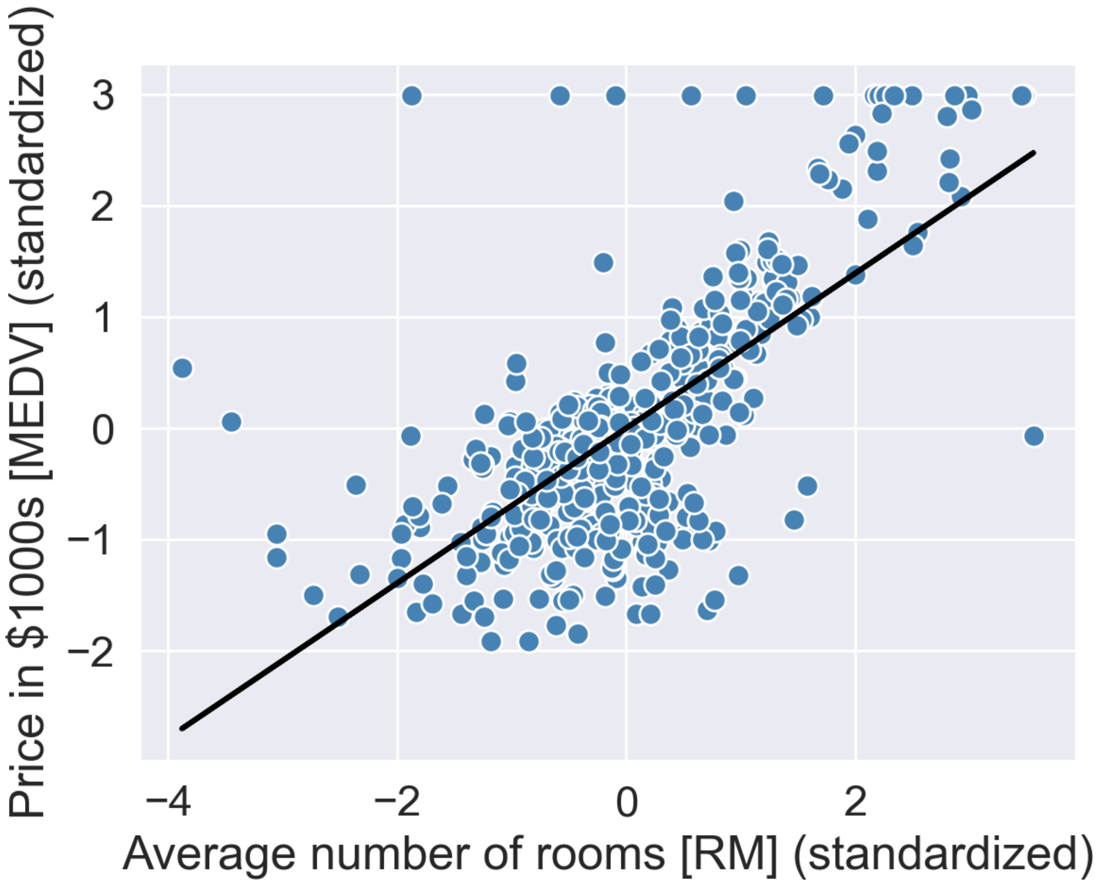
[16]:
```
<__main__.LinearRegressionGD at 0x7fe46acd9ab0>
```

[17]:
```
plt.plot(range(1, lr.n_iter+1), lr.cost_)
plt.ylabel('SSE')
plt.xlabel('Epoch')
#plt.tight_layout()
#plt.savefig('images/10_05.png', dpi=300)
plt.show()
```



[18]:
```
def lin_regplot(X, y, model):
    plt.scatter(X, y, c='steelblue', edgecolor='white', s=70)
    plt.plot(X, model.predict(X), color='black', lw=2)
    return
```

[19]:
```
lin_regplot(X_std, y_std, lr)
plt.xlabel('Average number of rooms [RM] (standardized)')
plt.ylabel('Price in $1000s [MEDV] (standardized)')

#plt.savefig('images/10_06.png', dpi=300)
plt.show()
```



[20]:
```
print('Slope: %.3f' % lr.w_[1])
print('Intercept: %.3f' % lr.w_[0])
```

```
Slope: 0.695
Intercept: -0.000
```

[21]:
```
from distutils.version import LooseVersion
import sklearn


num_rooms_std = sc_x.transform(np.array([[5.0]]))
price_std = lr.predict(num_rooms_std)

if LooseVersion(sklearn.__version__) >= LooseVersion('0.23.0'):
    print("Price in $1000s: %.3f" % sc_y.inverse_transform(price_std[:, np.
 ↪newaxis]).flatten())
else:
    print("Price in $1000s: %.3f" % sc_y.inverse_transform(price_std))
```

```
Price in $1000s: 10.840
```

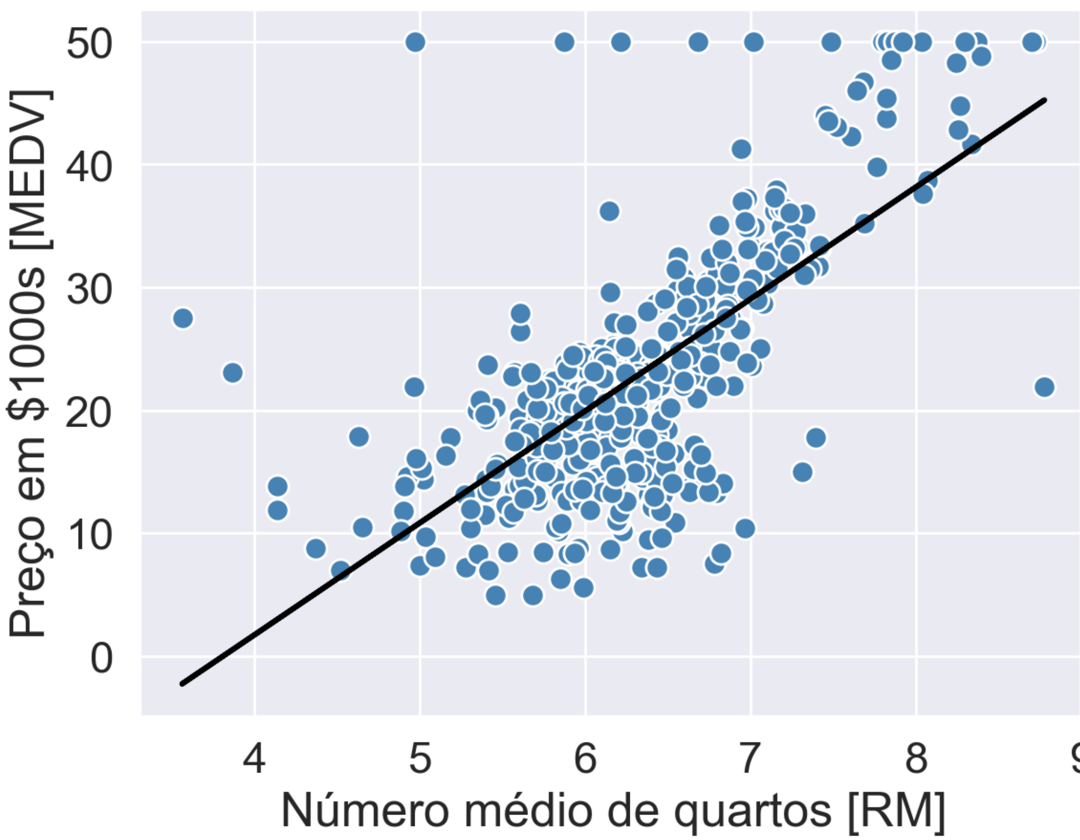
0.2 Estimando o coeficiente de um modelo de regressão via scikit-learn

[22]:
```
from sklearn.linear_model import LinearRegression
```

[23]:
```
slr = LinearRegression()
slr.fit(X, y)
y_pred = slr.predict(X)
print('Slope: %.3f' % slr.coef_[0])
print('Intercept: %.3f' % slr.intercept_)
```

```
Slope: 9.102
Intercept: -34.671
```

[24]:
```
lin_regplot(X, y, slr)
plt.xlabel('Número médio de quartos [RM]')
plt.ylabel('Preço em $1000s [MEDV]')

#plt.savefig('images/10_07.png', dpi=300)
plt.show()
```



[25]:

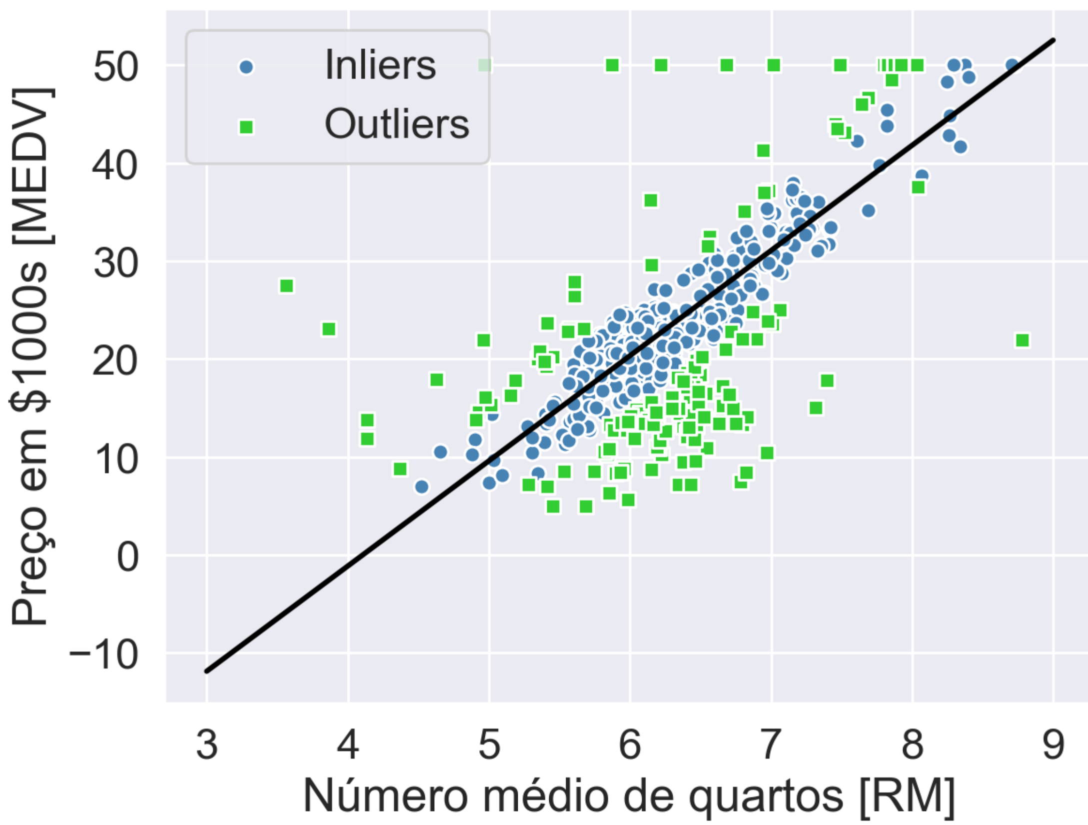
```
# adding a column vector of "ones"
Xb = np.hstack((np.ones((X.shape[0], 1)), X))
w = np.zeros(X.shape[1])
z = np.linalg.inv(np.dot(Xb.T, Xb))
w = np.dot(z, np.dot(Xb.T, y))

print('Slope: %.3f' % w[1])
print('Intercept: %.3f' % w[0])
```

```
Slope: 9.102
Intercept: -34.671
```

[26]:

```
from sklearn.linear_model import RANSACRegressor

ransac = RANSACRegressor(LinearRegression(),
                         max_trials=100,
                         min_samples=50,
                         loss='absolute_loss',
                         residual_threshold=5.0,
                         random_state=0)


ransac.fit(X, y)

inlier_mask = ransac.inlier_mask_
outlier_mask = np.logical_not(inlier_mask)

line_X = np.arange(3, 10, 1)
line_y_ransac = ransac.predict(line_X[:, np.newaxis])
plt.scatter(X[inlier_mask], y[inlier_mask],
            c='steelblue', edgecolor='white',
            marker='o', label='Inliers')
plt.scatter(X[outlier_mask], y[outlier_mask],
            c='limegreen', edgecolor='white',
            marker='s', label='Outliers')
plt.plot(line_X, line_y_ransac, color='black', lw=2)
plt.xlabel('Número médio de quartos [RM]')
plt.ylabel('Preço em $1000s [MEDV]')
plt.legend(loc='upper left')

#plt.savefig('images/10_08.png', dpi=300)
plt.show()
```

```
/usr/lib/python3/dist-packages/sklearn/linear_model/_ransac.py:369:
FutureWarning: The loss 'absolute_loss' was deprecated in v1.0 and will be
removed in version 1.2. Use `loss='absolute_error'` which is equivalent.
  warnings.warn(
```


Inliers
Outliers
Preço em $1000s [MEDV]
Número médio de quartos [RM]
50
40
30
20
10
0
−10
3
4
5
6
7
8
9