# 回归评价指标

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [2]:
```
x = np.loadtxt('boston/x.txt')
y = np.loadtxt('boston/y.txt')
x = x[:, 5]
```

In [3]:
```
x.shape
```

Out[3]:
```
(506,)
```

In [4]:
```
y.shape
```

Out[4]:
```
(506,)
```

In [5]:
```
from ML.model_selection import train_test_split
from ML.linear import LinearRegression
```

In [6]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, seed = 10
)
```

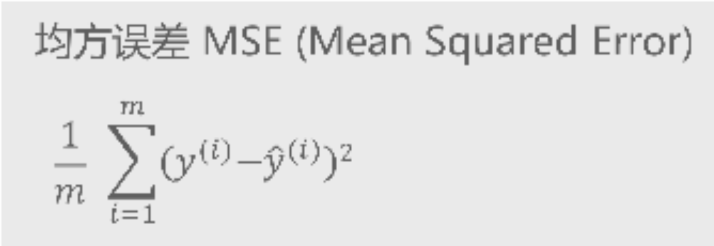
In [7]:
```
reg = LinearRegression()
reg.fit(x_train, y_train)
```

Out[7]:
```
<ML.linear.LinearRegression at 0x11692d8d0>
```

In [8]:
```
y_predict = reg.predict(x_test)
```

## MSE

```
In [9]: img = mpimg.imread('MSE.png')
        img.shape
        plt.imshow(img)
        plt.axis('off')
        plt.show()
```



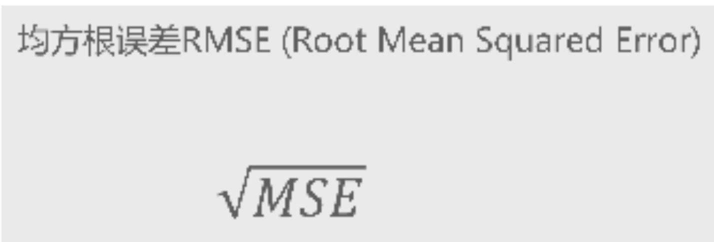

```
In [10]: def mean_squared_error(y_true, y_predict):
             assert len(y_true) == len(y_predict), 'y_true 与 y_predict 长度需要相同'
             return sum((y_true - y_predict) ** 2) / len(y_true)
```

```
In [11]: mean_squared_error(y_test, y_predict)

Out[11]: 48.11712557600194
```

## RMSE

```
In [12]: img = mpimg.imread('RMSE.png')
         img.shape
         plt.imshow(img)
         plt.axis('off')
         plt.show()
```



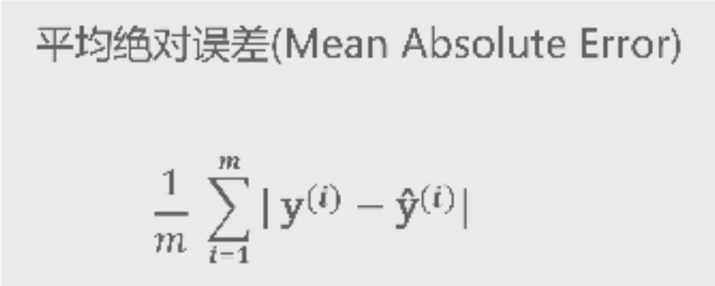

```
In [13]: def root_mean_squared_error(y_true, y_predict):
             return mean_squared_error(y_true, y_predict) ** (1/2)
```

```
In [14]: root_mean_squared_error(y_test, y_predict)

Out[14]: 6.936650890451525
```

## MAE

```
In [15]: img = mpimg.imread('MAE.png')
         img.shape
         plt.imshow(img)
         plt.axis('off')
         plt.show()
```




```
In [16]: def mean_absolute_error(y_true, y_predict):
             assert len(y_true) == len(y_predict), 'y_true 与 y_predict 长度需要相同'
             return sum(np.absolute(y_true - y_predict)) / len(y_true)
```

```
In [17]: mean_absolute_error(y_test, y_predict)
```

```
Out[17]: 4.837836369273781
```

## R Squared

In [18]:

```
img1 = mpimg.imread('R Squared1.png')
img1.shape
plt.imshow(img1)
plt.axis('off')
plt.show()
img2 = mpimg.imread('R Squared2.png')
img2.shape
plt.imshow(img2)
plt.axis('off')
plt.show()
```

$$R^2 = 1 - \frac{SSR}{SST} \quad \begin{matrix} \text{(Residual Sum of Squares)} \\ \text{(Total Sum of Squares)} \end{matrix}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2}{\sum_{i=1}^{m} \left(\bar{y} - y^{(i)}\right)^2}$$

$$R^2 = 1 - \frac{\left(\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)}\right)^2\right)/m}{\left(\sum_{i=1}^{m} \left(\bar{y} - y^{(i)}\right)^2\right)/m}$$

$$= 1 - \frac{MSE(\hat{y}, y)}{Var(y)}$$

In [19]:

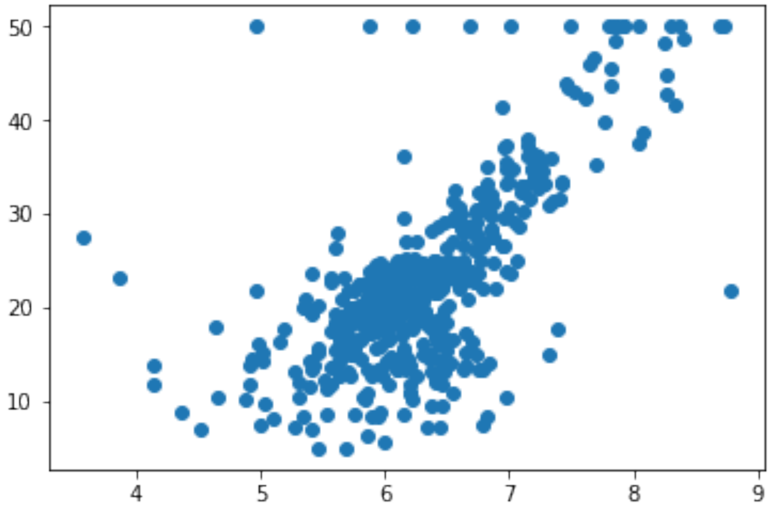
```
def r2_score(y_true, y_predict):
    return 1 - mean_squared_error(y_true, y_predict) / np.var(y_true)
```

In [20]:

```
r2_score(y_test, y_predict)
```

Out[20]: 0.5142606186922454

In [21]:
```
plt.scatter(x, y)
plt.show()
```



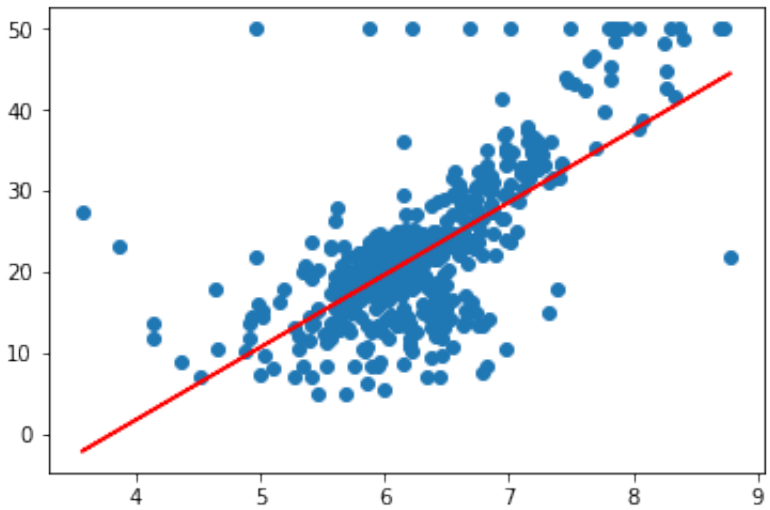

In [22]:
```
reg.a_
```

Out[22]: 8.929662121469637

In [23]:
```
reg.b_
```

Out[23]: -33.92477502563315

In [24]:
```
plt.scatter(x, y)
plt.plot(x_train, reg.predict(x_train), color='red')
plt.show()
```




In [25]:
```
reg.accuracy_rate(x_test, y_test)
```

Out[25]: 0.5142606186922454