Selecting k best features

- one of the scoring functions (f_regression for F-values)
- Linear model for testing the individual effect of each of many regressors
- Precision - ability of classifier NOT to label as positive a sample that is negative
- Recall - ability of the classifier to find all the positive samples
- F-score = 2*(precision * recall)/(precision + recall)

In [1]:
```
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn import datasets
from sklearn.feature_selection import SelectKBest, f_regression

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, scorer
```

In [3]:
```
boston_dataset = datasets.load_boston()
```

In [4]:
```
X_full = boston_dataset.data
y = boston_dataset.target

print(X_full.shape)
print(y.shape)
```

```
(506, 13)
(506,)
```

In [5]:
```
print(boston_dataset.feature_names)
```

```
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

SelectKBest(score_func, k = 10)

- Select features according to the k highest scores

In [6]:
```
# Select the top two features to use for Linear Regression

selector = SelectKBest(f_regression, k=2)
selector.fit(X_full, y)
```

Out[6]:
```
SelectKBest(k=2, score_func=<function f_regression at 0x111631378>)
```

In [7]:
```
selector.get_support()
```

Out[7]:
```
array([False, False, False, False, False,  True, False, False, False,
       False, False, False,  True])
```

In [8]:
```
print(boston_dataset.feature_names[selector.get_support()])
```

```
['RM' 'LSTAT']
```

In [9]:
```
selector.scores
```

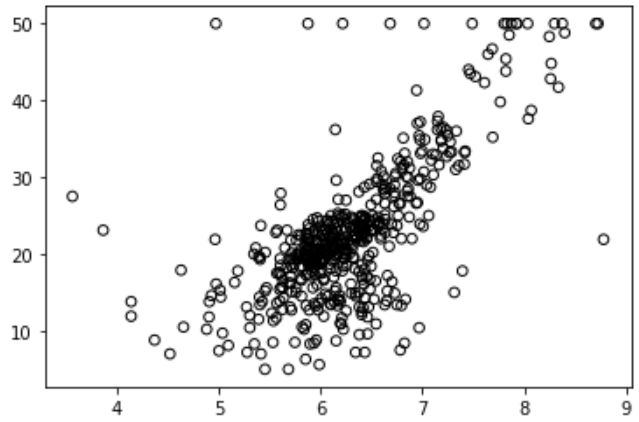
Out[9]:
```
array([ 89.48611476,  75.2576423 , 153.95488314,  15.97151242,
       112.59148028, 471.84673988,  83.47745922,  33.57957033,
        85.91427767, 141.76135658, 175.10554288,  63.05422911,
       601.61787111])
```

In [10]:
```
X = X_full[:, selector.get_support()]
print(X.shape)
```

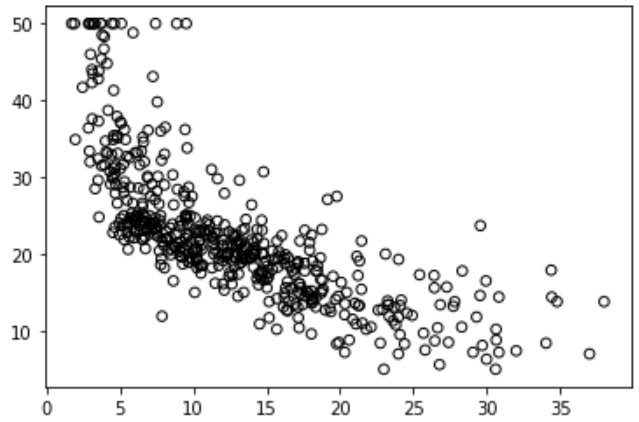
```
(506, 2)
```

In [11]:
```
def plot_scatter(X,Y,R=None):
    plt.scatter(X, Y, s=32, marker='o', facecolors='none', edgecolors='k')
    if R is not None:
        plt.scatter(X, R, color='red', linewidth=0.5)
    plt.show()
```

In [12]:
```
plot_scatter(X[:,0], y)
```



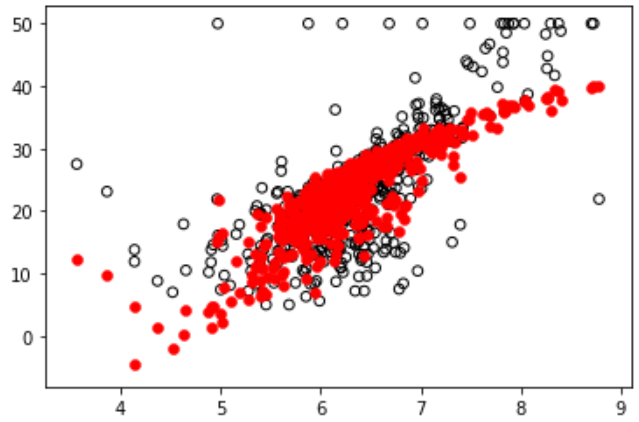
In [13]:
```
plot_scatter(X[:,1], y)
```



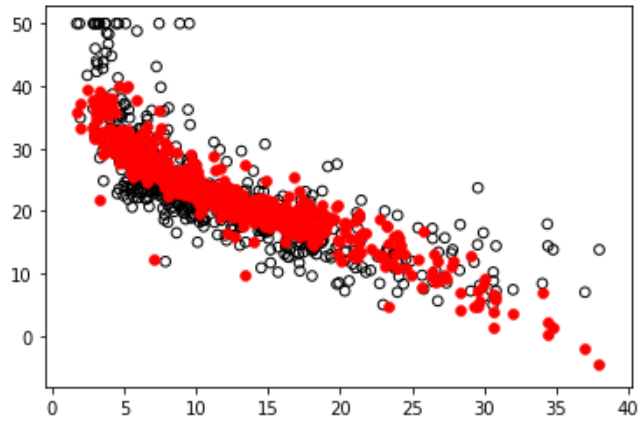
In [14]:
```
regressor = LinearRegression(normalize=True).fit(X, y)
y_pred    = regressor.predict(X)
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/sklearn/linear_model/base.py:503:
RuntimeWarning: internal gelsd driver lwork query error, required iwork dimension not returned. This is likely
the result of LAPACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gelss' driver.
  linalg.lstsq(X, y)
```

In [15]:
```
plot_scatter(X[:,0], y, y_pred)
```



In [16]:
```
plot_scatter(X[:,1], y, y_pred)
```



In [17]:
```
print("R-squared score: {:.4f}".format(
    r2_score(y, y_pred)))
```

```
R-squared score: 0.6386
```

In [ ]:
```

```