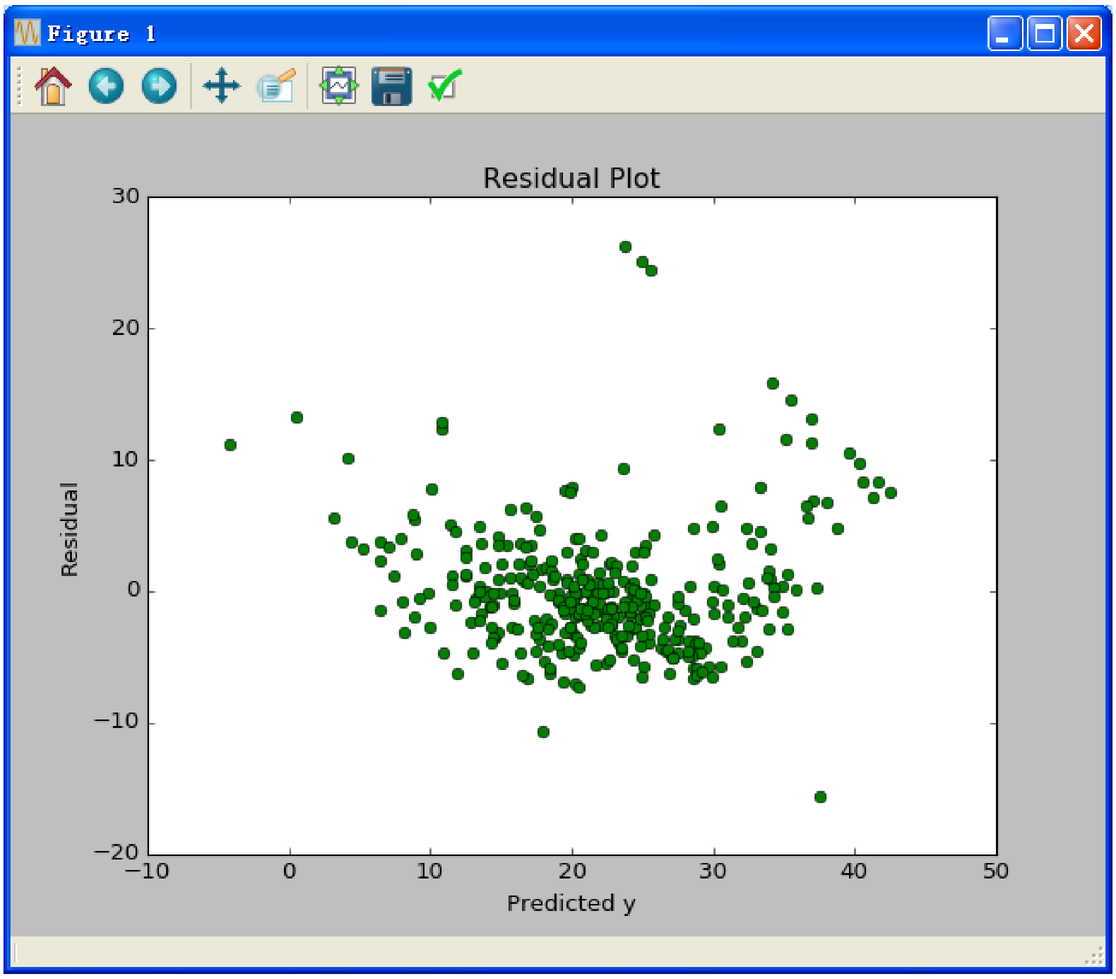
```
# 线性回归分析 boston 数据集
In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: from sklearn.datasets import load_boston

In [4]: from sklearn.cross_validation import train_test_split

In [5]: from sklearn.linear_model import LinearRegression

In [6]: from sklearn.metrics import mean_squared_error

In [7]: import matplotlib.pyplot as plt

In [8]: from sklearn.preprocessing import PolynomialFeatures

# 将数据集划分成训练集,开发集,测试集
In [9]: data = load_boston()

In [10]: X = data['data']

In [11]: Y = data['target']

In [13]: X_train,X_test_all,y_train,y_test_all =
            train_test_split(X,Y,test_size=0.3,random_state=9)

In [14]: X_dev,X_test,y_dev,y_test =
            train_test_split(X_test_all,y_test_all,
                             test_size=0.3,random_state=9)

# 建模
In [15]: model = LinearRegression(normalize=True,fit_intercept=True)

In [16]: model.fit(X_train,y_train)

In [17]: y_predicted = model.predict(X_train)

# 绘制残差
In [18]: %matplotlib

In [19]: plt.cla()

In [20]: plt.xlabel("Predicted y")
In [21]: plt.ylabel("Residual")

In [22]: plt.title("Residual Plot")

In [23]: plt.figure(1)

In [24]: diff = y_train - y_predicted

In [25]: plt.plot(y_predicted,diff,'go')
```



```
# 查看模型系数
In [26]: for i,coef in enumerate(model.coef_):
    ...:     print "Coefficient %d %0.3f" % (i+1,coef)
    ...:
Coefficient 1 -0.109
Coefficient 2 0.043
Coefficient 3 0.053
Coefficient 4 2.237
Coefficient 5 -15.879
Coefficient 6 3.883
Coefficient 7 0.001
Coefficient 8 -1.321
Coefficient 9 0.284
Coefficient 10 -0.012
Coefficient 11 -0.904
Coefficient 12 0.009
Coefficient 13 -0.529

In [27]: print "Intercept %0.3f" % (model.intercept_)
Intercept 33.288

# 评价模型
In [28]: print "MSE = %0.2f"
            % (mean_squared_error(y_train,y_predicted))
MSE = 23.18

# 模型作用到开发集上
In [29]: y_predicted = model.predict(X_dev)

In [31]: print "MSE = %0.2f"
             % (mean_squared_error(y_dev,y_predicted))
MSE = 18.25    # 在开发集上 MSE 更小

# 准备一些多项式特征
In [32]: poly_features = PolynomialFeatures(2)

In [33]: poly_features.fit(X_train)

In [35]: X_train_poly = poly_features.transform(X_train)

In [36]: X_dev_poly = poly_features.transform(X_dev)

In [49]: X_train_poly.shape
Out[49]: (354, 105)

In [50]: X_train.shape
Out[50]: (354, 13)

# 用多项式特征建模
In [37]: model_poly = LinearRegression(normalize=True,
                                       fit_intercept=True)
```

```
In [38]: model_poly.fit(X_train_poly,y_train)

In [39]: y_predicted = model_poly.predict(X_train_poly)

In [40]: print "MSE = %0.2f"
             % (mean_squared_error(y_train,y_predicted))
MSE = 5.46    # 很有效

# 将模型作用到开发集上
In [41]: y_predicted = model_poly.predict(X_dev_poly)

In [42]: print "MSE = %0.2f"
             % (mean_squared_error(y_dev,y_predicted))
MSE = 13.23

# 将模型应用到测试集
In [43]: X_test_poly = poly_features.transform(X_test)

In [44]: y_predicted = model_poly.predict(X_test_poly)

In [45]: print "MSE = %0.2f"
             % (mean_squared_error(y_test,y_predicted))
MSE = 15.13

In [46]: y_predicted = model.predict(X_test)

In [47]: print "MSE = %0.2f"
             % (mean_squared_error(y_test,y_predicted))
MSE = 21.66
```