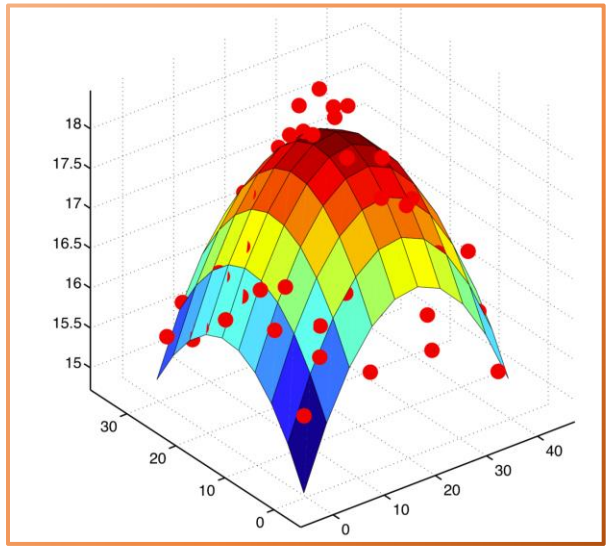


第九讲 回归分析与回归模型

云南大学

孙正宝

zbsun@ynu.edu.cn

提　纲

基本问题

•**Prediction（预测）：** Useful when the input variable is readily available, but the output variable is not.

- **Example:** *Predict stock prices next month using data from last year.*

•**Inference（推断）：** A model for f can help us understand the structure of the data — which variables influence the output, and which don't? What is the relationship between each variable and the output, e.g. linear, non-linear?

- **Example:** *What is the influence of genetic variations on the incidence of heart disease.*

相关性与回归分析

- **回归**：一种确定关系，通过一个或多个变量（自变量）的取值能够得到另一个变量（因变量）的取值，可以通过回归方程（模型）实现。
- **相关**：非确定关系，当一个（多个）变量的取值发生变化时，与它（它们）相关的变量的取值也会发生变化，但变化值是不确定的。
- 相关关系主要用于考察分析两个或多个变量之间的相关情况。
  - Pearson相关系数
  - Spearman相关系数

● Pearson相关系数

1. 适用条件

✓ 两个变量分别服从正态分布（当数据量够大时，譬如 $n>30$，根据中心极限定理，可以假定服从正态分布）；

✓ 两个变量的标准差不为0（通常都满足）。

2. 计算方法

$$\rho = \frac{E[(X-\mu_x)(Y-\mu_y)]}{\sigma_x \sigma_y} = \frac{\Sigma(x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma(x-\bar{x})^2}\sqrt{\Sigma(y-\bar{y})^2}}$$

3. 显著性检验

✓ 通常使用 t 检验来验证Pearson相关系数的显著性。

Spearman相关系数

1. 适用条件

✓ 没有特殊的限制条件，只要求数据成对即可。

2. 计算方法

✓ 原始数据：$X=[X_1,X_2,\dots X_n],Y=[Y_1,Y_2,\dots Y_n]$

✓ 转换之后的等级数据：$x=[x_1,x_2,\dots x_n],y=[y_1,y_2,\dots y_n]$

$$\rho=\frac{\Sigma(x-\bar{x})(y-\bar{y})}{\sqrt{\Sigma(x-\bar{x})^2}\sqrt{\Sigma(y-\bar{y})^2}}$$

✓ 当等级数据都是整数时，公式可简化为：$\rho=1-\frac{6\Sigma d_i^2}{n(n^2-1)}$

3. 显著性检验：t 检验。

变量与数据集——线代基础

- 离散变量

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \qquad x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

$$x_i^T = \begin{pmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{pmatrix}$$

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

- 连续变量：离散采样

模型的评价与选择

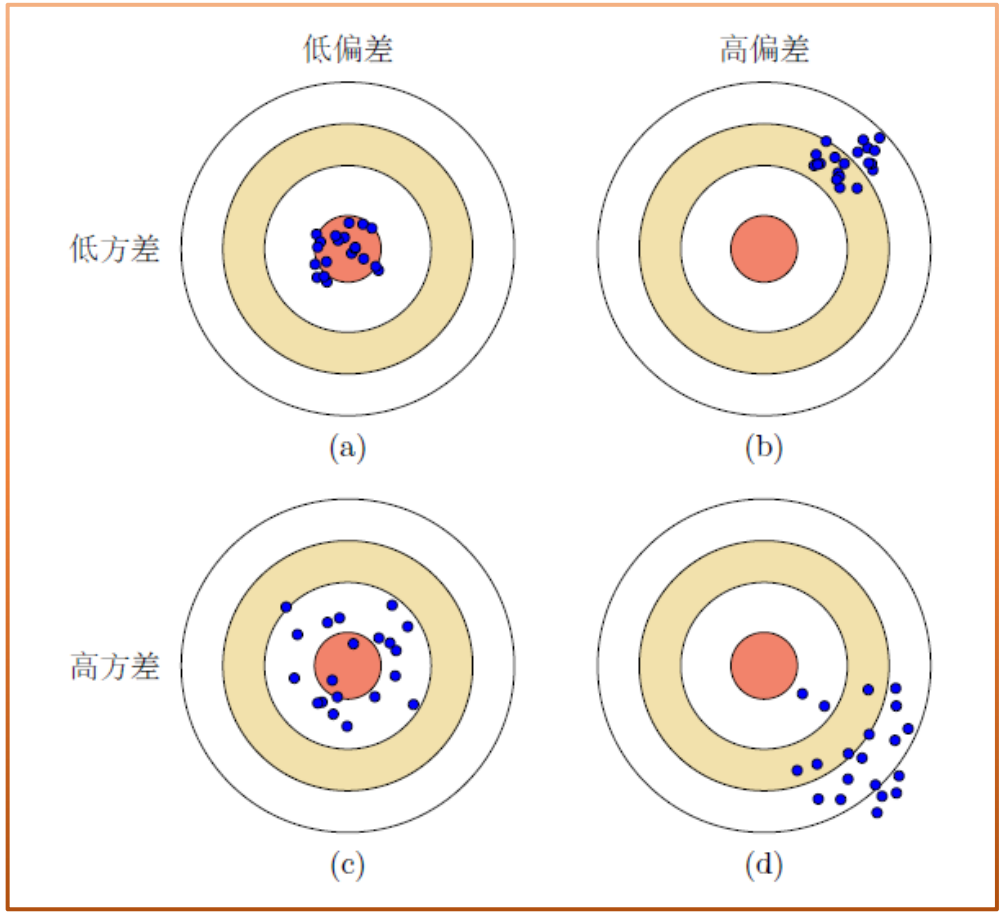
- 建模过程——模型的习得过程

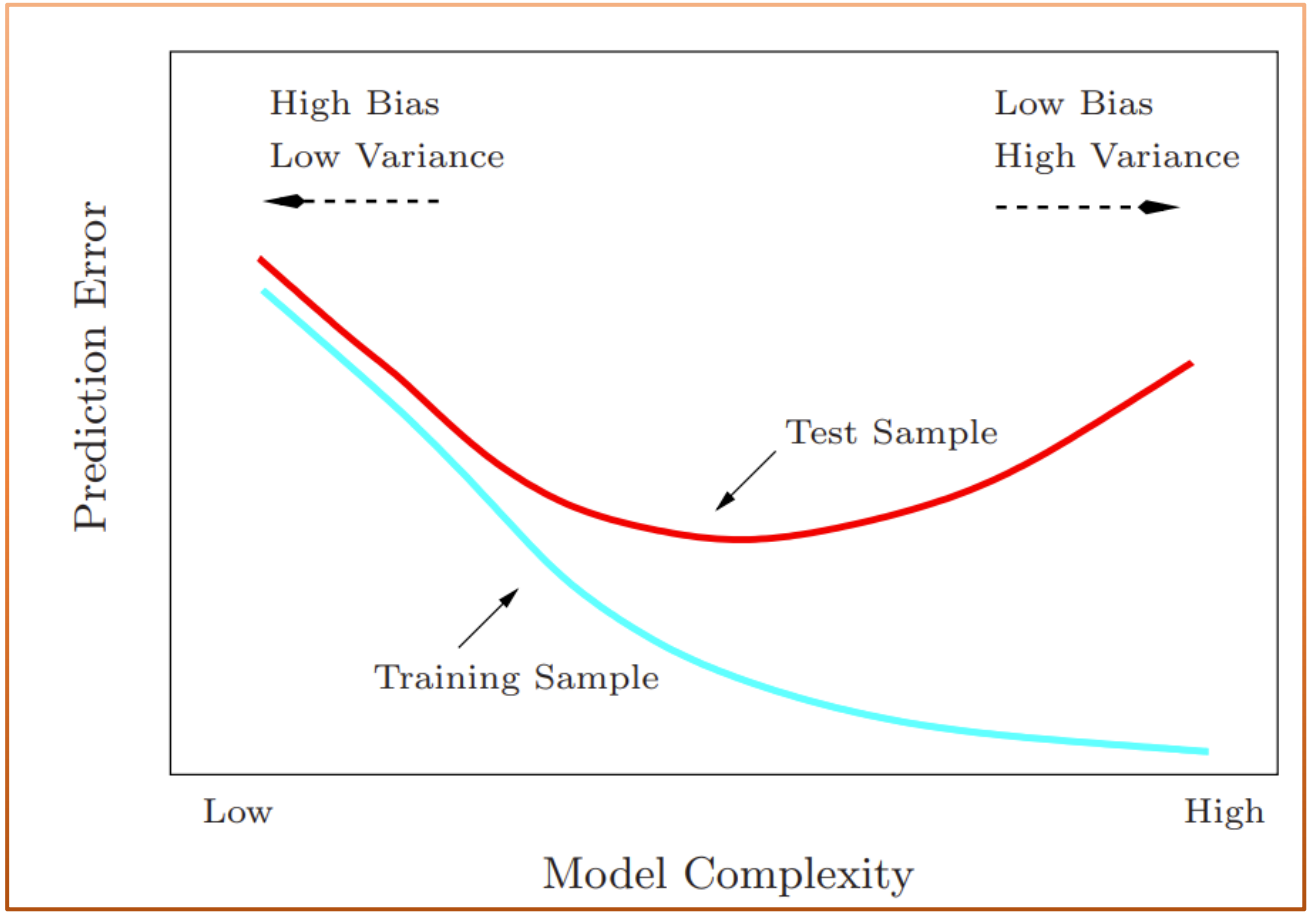
- 方差-偏差分解

$$
\begin{aligned}
E(f;D) &= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-y_D)^2\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x})+\bar{f}(\boldsymbol{x})-y_D)^2\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x}))^2\right]+\mathbb{E}_D\left[(\bar{f}(\boldsymbol{x})-y_D)^2\right] \\
&\quad +\mathbb{E}_D\left[2\left(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x})\right)\left(\bar{f}(\boldsymbol{x})-y_D\right)\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x}))^2\right]+\mathbb{E}_D\left[(\bar{f}(\boldsymbol{x})-y_D)^2\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x}))^2\right]+\mathbb{E}_D\left[(\bar{f}(\boldsymbol{x})-y+y-y_D)^2\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x}))^2\right]+\mathbb{E}_D\left[(\bar{f}(\boldsymbol{x})-y)^2\right]+\mathbb{E}_D\left[(y-y_D)^2\right] \\
&\quad +2\mathbb{E}_D\left[(\bar{f}(\boldsymbol{x})-y)(y-y_D)\right] \\
&= \mathbb{E}_D\left[(f(\boldsymbol{x};D)-\bar{f}(\boldsymbol{x}))^2\right]+(\bar{f}(\boldsymbol{x})-y)^2+\mathbb{E}_D\left[(y_D-y)^2\right]
\end{aligned}
$$

模型的评价与选择

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon_0)$$





模型的评价与选择

- 模型评价指标

$$R^2 = \frac{\mathrm{TSS} - \mathrm{RSS}}{\mathrm{TSS}} = 1 - \frac{\mathrm{RSS}}{\mathrm{TSS}}$$

$$\mathrm{TSS} = \sum (y_i - \bar{y})^2$$

$$\mathrm{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

$$\mathrm{RSE} = \sqrt{\frac{1}{n-p-1}\mathrm{RSS}}$$

MSE: Mean Squared Error
RSE: Residual Standard Error
TSS: Total Sum of Squares

提　纲

一、线性回归（Linear Regression）

- 一元线性回归

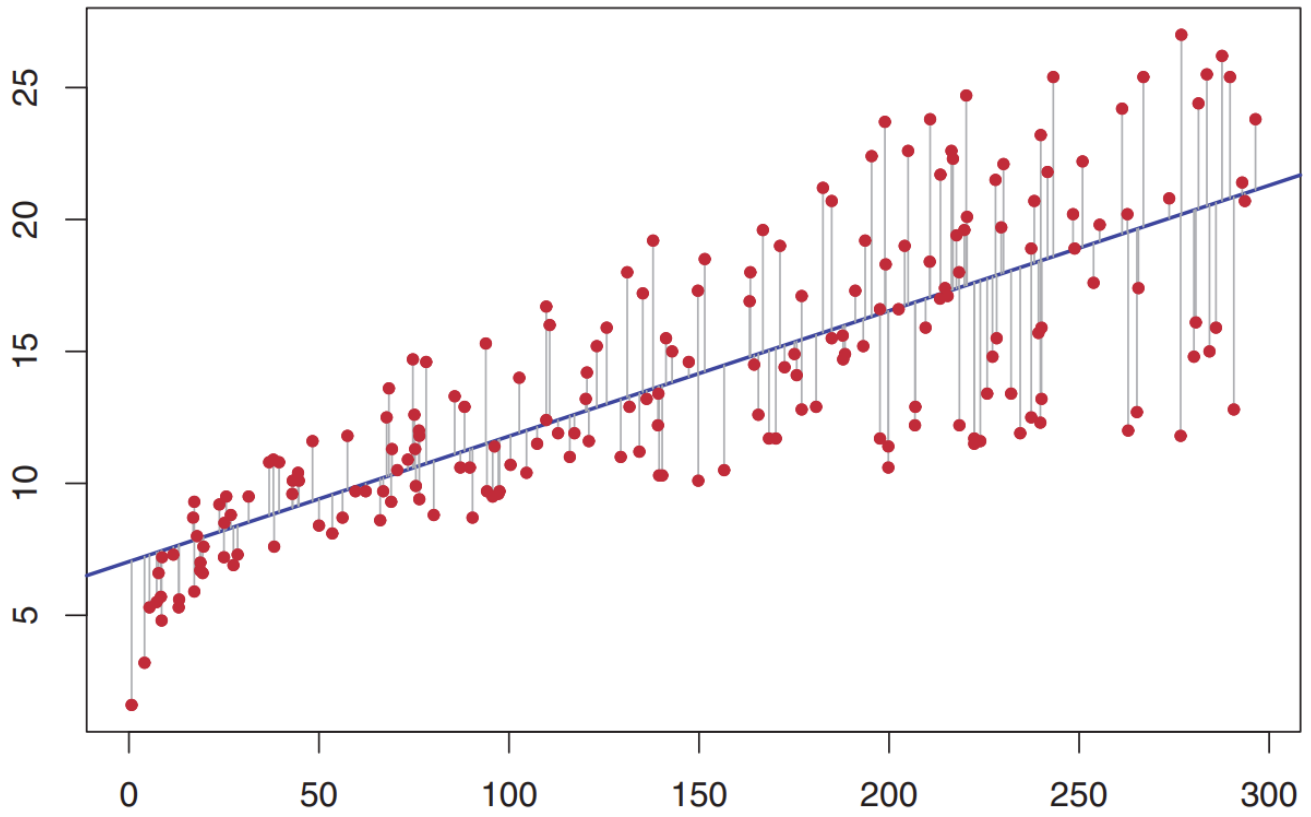
$$Y = \beta_0 + \beta_1 X + \epsilon$$

✓ 最小二乘法估计系数

$$\mathrm{RSS}(f) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

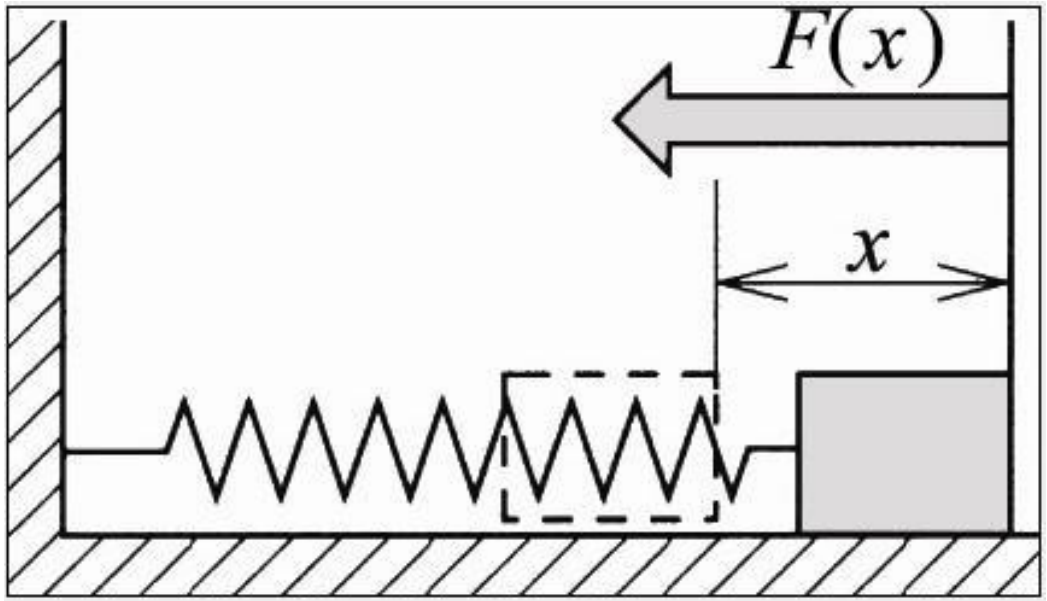
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$



RSS: Residual Sum of Squares.

一、线性回归（Linear Regression）

- 一元线性回归
  - 客观物理现象



■ 实验设想：

虽然不能给每一个x都做一次测量，但是可以选择一些不同的x进行实验，得到对应的实验结果F：

x/cm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
F/N	0.1673	0.5232	0.8221	0.9997	0.9457	1.3028	1.486	1.6431	1.7646	2.143

一、线性回归（Linear Regression）

● 一元线性回归

■ 算法设计：(x_1,F_1)

#用x计算F：

1. 获得实验数据集S，

   $S=\{(x_1,F_1), (x_2,F_2), (x_3,F_3),\ldots\}$

2. 利用S对x和F两个变量进行线性回归，

   得到模型($F=kx+b$)

3. Input x=?

4. 利用获得的模型($F=kx+b$)

   求解对应的F

5. Return F

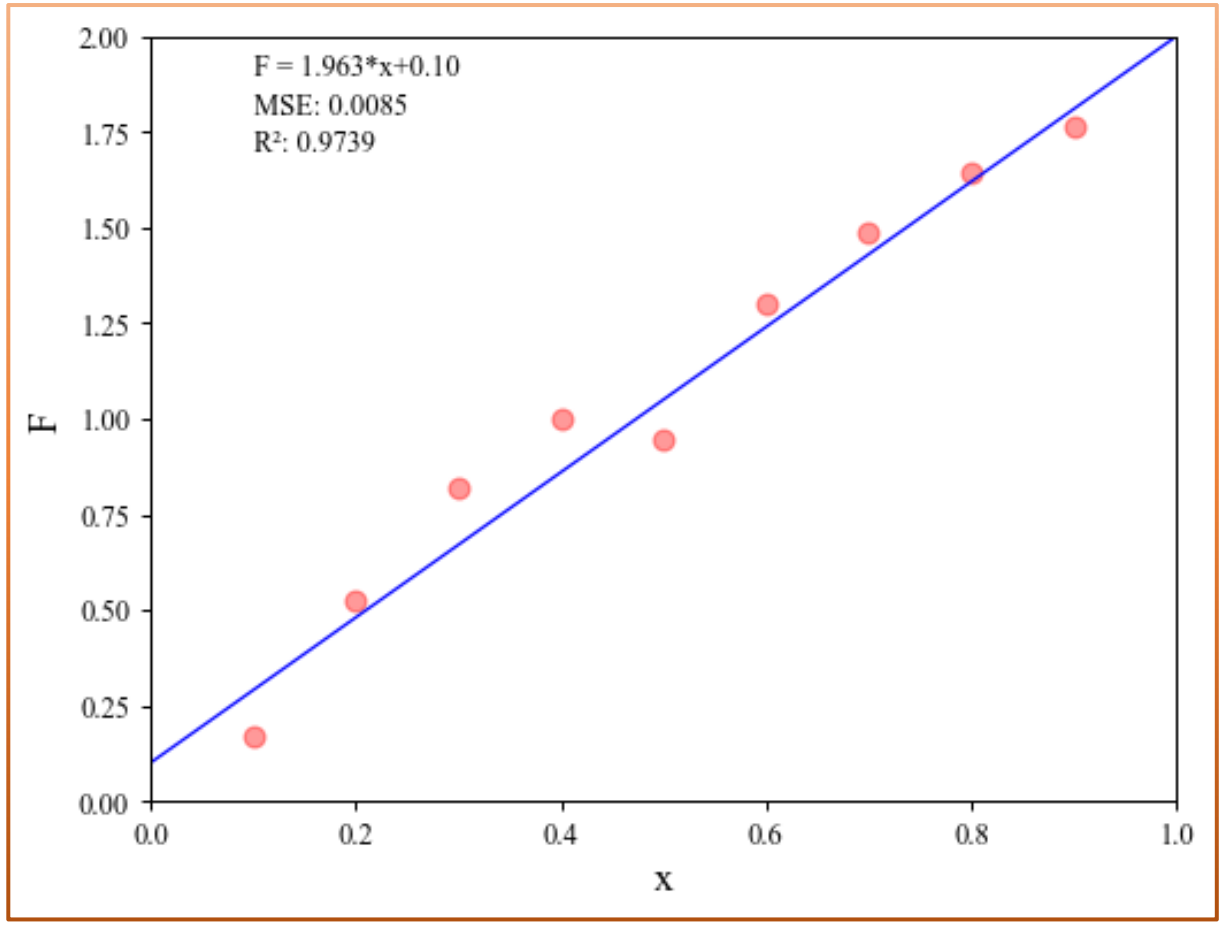
```
    Xsum=0.0
    X2sum=0.0
    Fsum=0.0
    xF=0.0
    n=len(x)
    for i in range(n):
        Xsum+=x[i]
        Fsum+=F[i]
        xF+=x[i]*F[i]
        X2sum+=x[i]**2
    k=(Xsum*Fsum/n-xF)/(Xsum**2/n-X2sum)
    b=(Fsum-k*Xsum)/n
    print('the line is F=%f*x+%f' % (k,b) )
```

✓ 0.0s

```
the line is F=1.963158*x+0.100013
```

一、线性回归（Linear Regression）

- 一元线性回归



■ 胡克定理：

1. #根据胡克定理，用*x*计算*F*，

   k=2 //*k* =2 *N/cm*

2. Input *x*=？

3. *F*=*kx*

4. Return *F*

一、线性回归（Linear Regression）

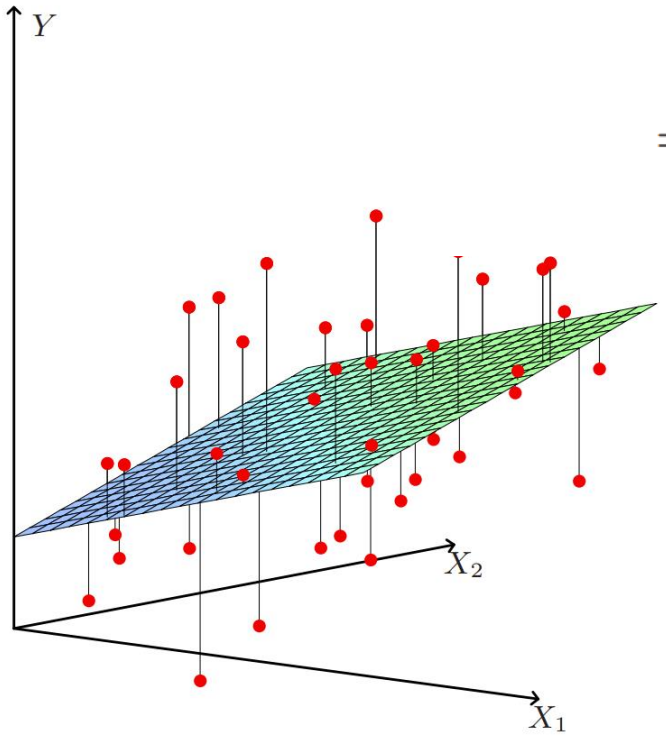
- 多元线性回归

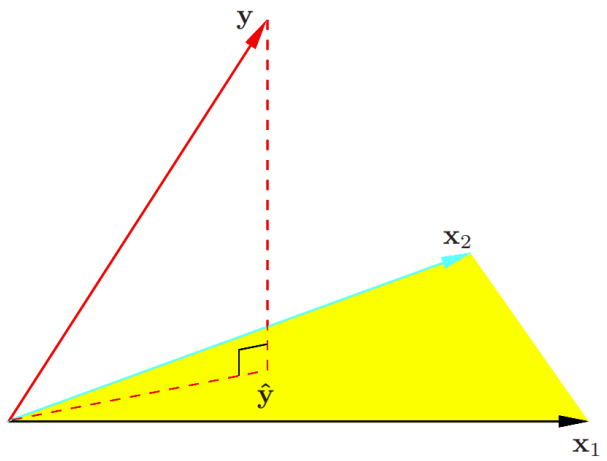
$$Y_k = \beta_{0k} + \sum_{j=1}^{p} X_j \beta_{jk} + \varepsilon_k$$

✓ 最小二乘法估计系数

$$\mathrm{RSS}(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2$$

$$= \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$





$$\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta)$$

$$\frac{\partial \mathrm{RSS}}{\partial \beta} = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta)$$

$$\frac{\partial^2 \mathrm{RSS}}{\partial \beta \partial \beta^T} = 2\mathbf{X}^T\mathbf{X}.$$

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

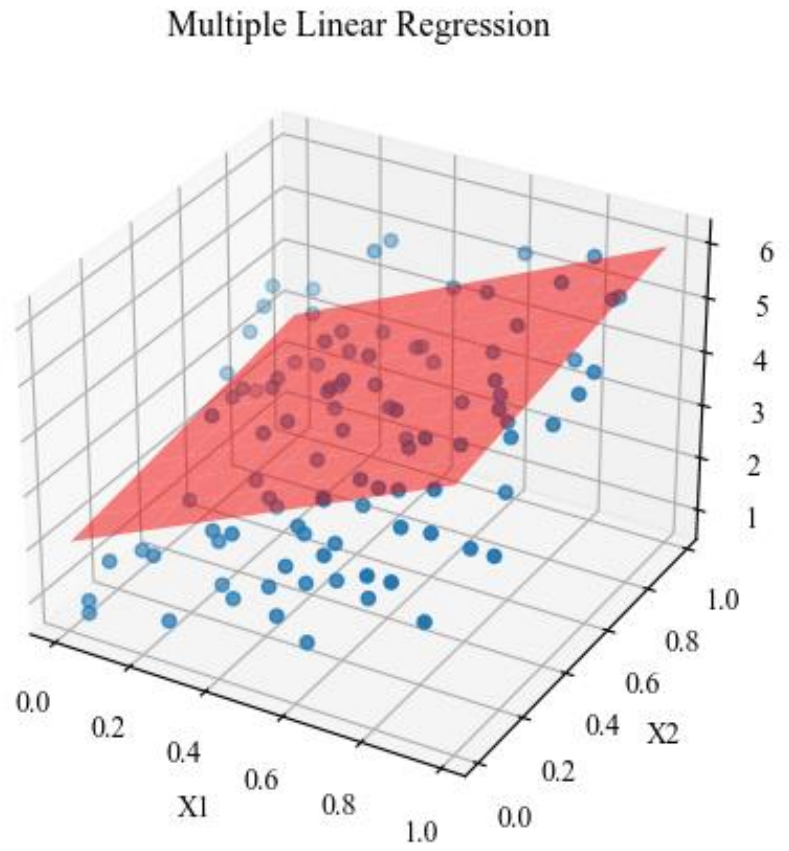
$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

一、线性回归（Linear Regression）

● 多元线性回归

```
# 拟合二元线性回归模型
X = np.column_stack((X1, X2, np.ones_like(X1)))  # 添加一列全1作为截距项
beta_hat = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)  # 最小二乘法求解

# 构建回归方程
equation = f"y = {beta_hat[0]:.2f} + {beta_hat[1]:.2f} * x1 + {beta_hat[2]:.2f} * x2"

# 出图
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X1, X2, y, label='Data')  # 绘制散点图

# 创建网格来绘制回归面
x1_vals = np.linspace(X1.min(), X1.max(), 10)
x2_vals = np.linspace(X2.min(), X2.max(), 10)
x1_grid, x2_grid = np.meshgrid(x1_vals, x2_vals)
y_grid = beta_hat[0] + beta_hat[1] * x1_grid + beta_hat[2] * x2_grid

# 绘制回归面
ax.plot_surface(x1_grid, x2_grid, y_grid, alpha=0.5, color='red', label='Regression Plane')

# 设置轴标签和图标题
ax.set_xlabel('X1')
ax.set_ylabel('X2')
ax.set_zlabel('y')
ax.set_title('Multiple Linear Regression')
```



y = 2.03 + 3.18 * x1 + 0.88 * x2

一、线性回归（Linear Regression）

- 多元线性回归

```
# 创建一个线性回归模型
def fit_linear_model(X, y):
    X = sm.add_constant(X)  # 添
    model = sm.OLS(y, X).fit()
    return model

# 拟合多元线性回归模型
model = fit_linear_model(X, y)

# 输出模型结果
print(model.summary())
```

✓ 参数显著性检验，参数显著不为0：

$$t = \frac{\hat{\beta}_k - \beta_k}{se(\hat{\beta}_k)} \sim t(n-p-1)$$

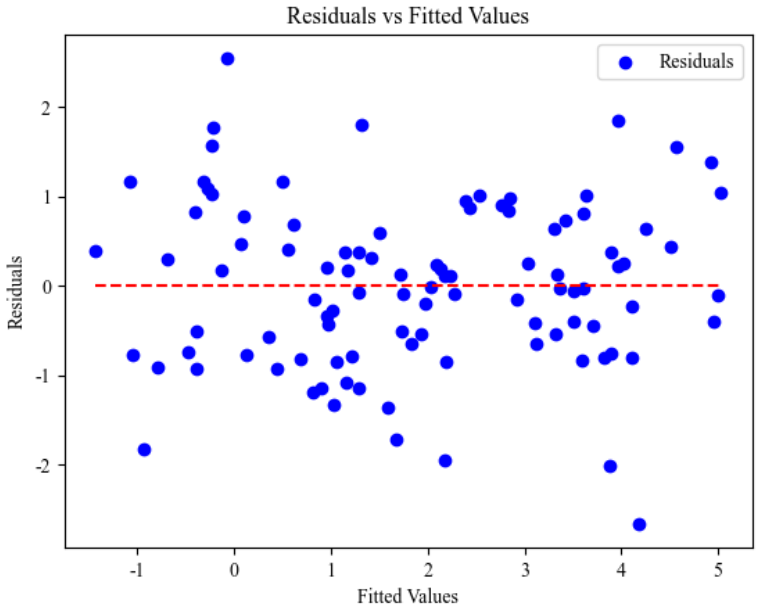
OLS Regression Results

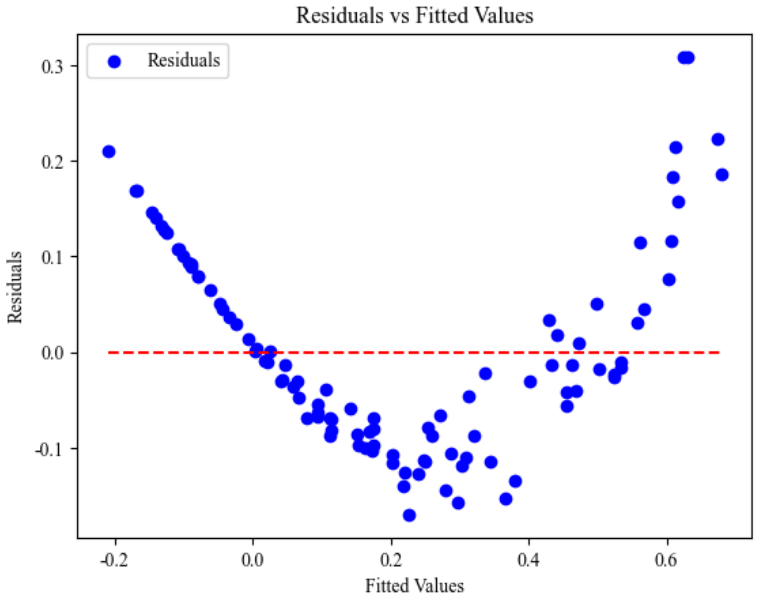
Dep. Variable:	y	R-squared:	0.713
Model:	OLS	Adj. R-squared:	0.681
Method:	Least Squares	F-statistic:	22.15
Date:	Fri, 26 Apr 2024	Prob (F-statistic):	3.88e-20
Time:	13:17:13	Log-Likelihood:	-126.38
No. Observations:	100	AIC:	274.8
Df Residuals:	89	BIC:	303.4
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.9673	0.502	3.920	0.000	0.970	2.965
x1	1.3283	0.395	3.362	0.001	0.543	2.113
x2	2.0805	0.318	6.534	0.000	1.448	2.713
x3	-3.9698	0.331	-12.007	0.000	-4.627	-3.313
x4	0.2970	0.347	0.855	0.395	-0.393	0.987
x5	0.0835	0.312	0.268	0.789	-0.536	0.703
x6	-0.1779	0.337	-0.527	0.599	-0.848	0.492
x7	0.6848	0.335	2.043	0.044	0.019	1.351
x8	0.0736	0.315	0.234	0.816	-0.553	0.700
x9	-0.1818	0.334	-0.543	0.588	-0.846	0.483
x10	-0.5218	0.337	-1.549	0.125	-1.191	0.148

一、线性回归（Linear Regression）

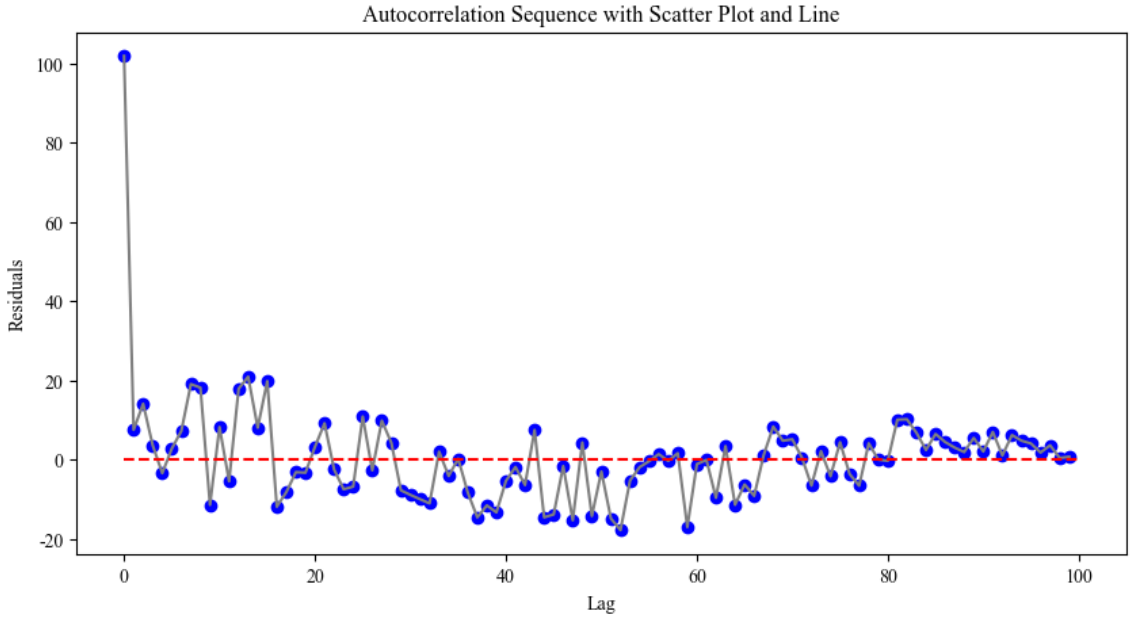
- **需要注意的问题：**
  - **数据的非线性**：线性回归模型假定预测变量和响应变量之间有直线关系。如果真实关系是非线性的，那么得出的几乎所有结论都是不可信的，而且模型的预测精度也可能显著降低。
  - **残差图**：理想情况下，残差图显示不出明显的规律。若存在明显规律，则表示线性模型的某些方面可能有问题。

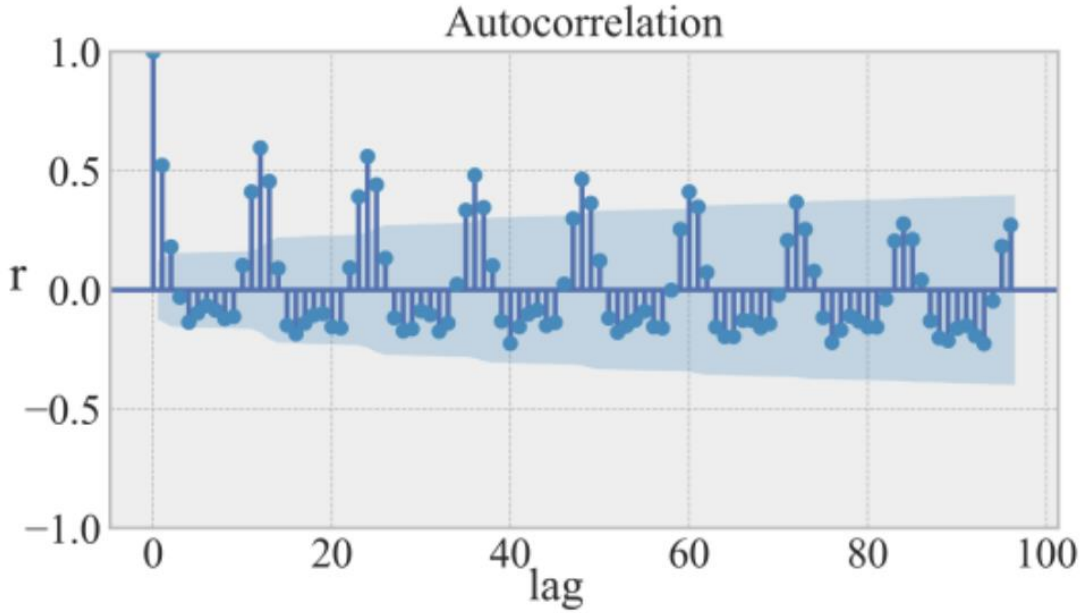




一、线性回归（Linear Regression）

- **需要注意的问题：**
  - **误差项自相关：**线性回归模型的一个重要假设是误差项不相关（理想情况下应为白噪声）。误差项相关关系经常出现在时间序列数据中。
  - **自相关函数图（ACF/PACF）：**为了确定某一给定的数据集是否有误差自相关问题，绘制作为时间函数的残差和ACF进行判断。

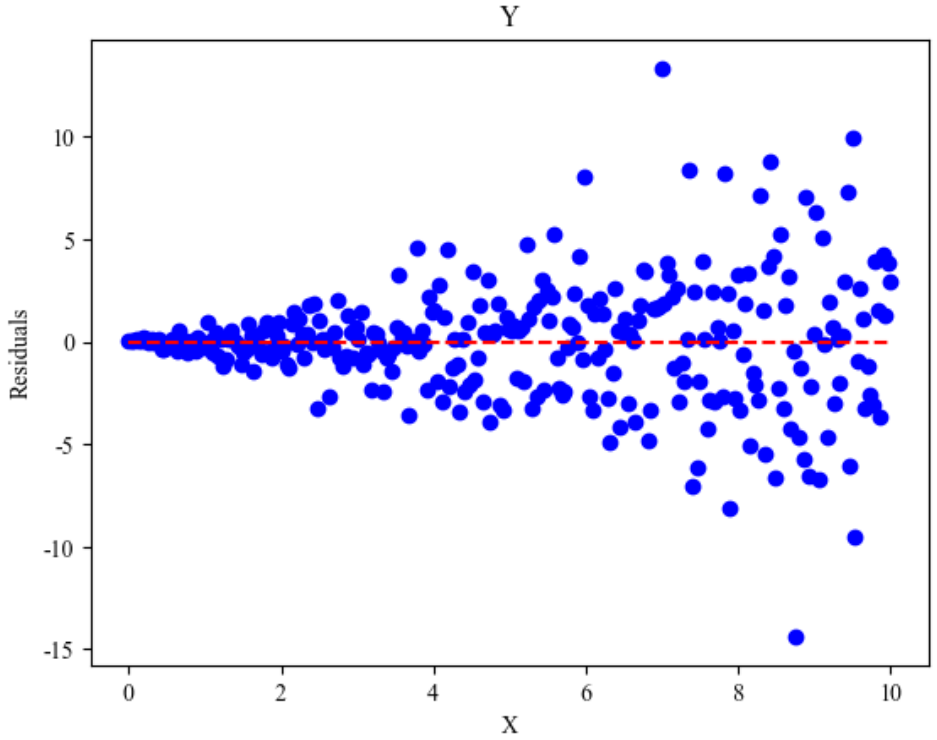


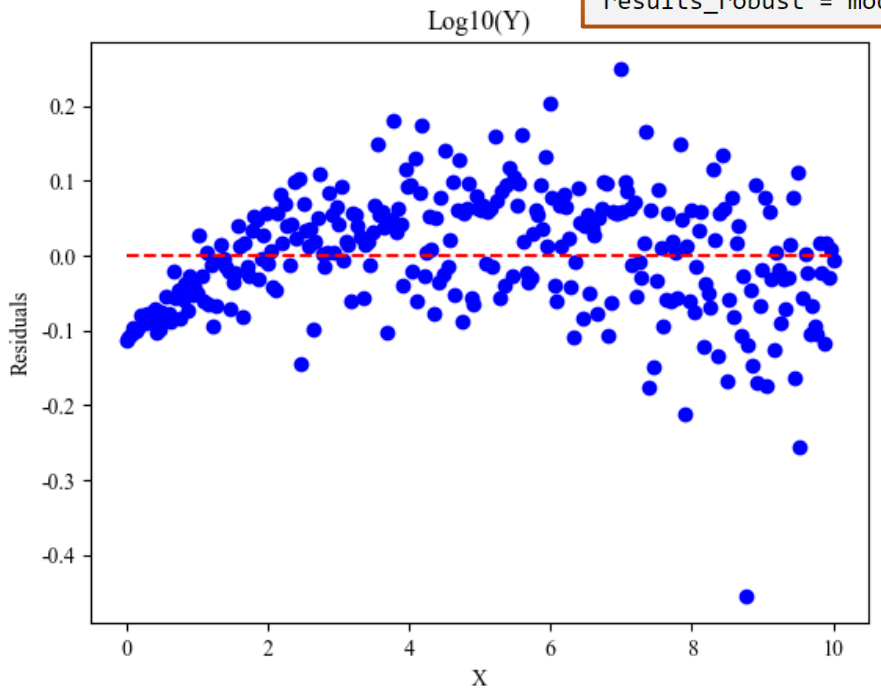


一、线性回归（Linear Regression）

- **需要注意的问题：**
  - **误差项方差非恒定：**模型的假设检验和标准误差、置信区间的计算都依赖于误差项的方差是恒定的假设。某些情况下，误差项的方差不是恒定的。例如，误差项的方差可能会随响应值的增加而增加（如图）。
  - **用凹函数对响应值y做变换：**如logY、\sqrt(Y)等。

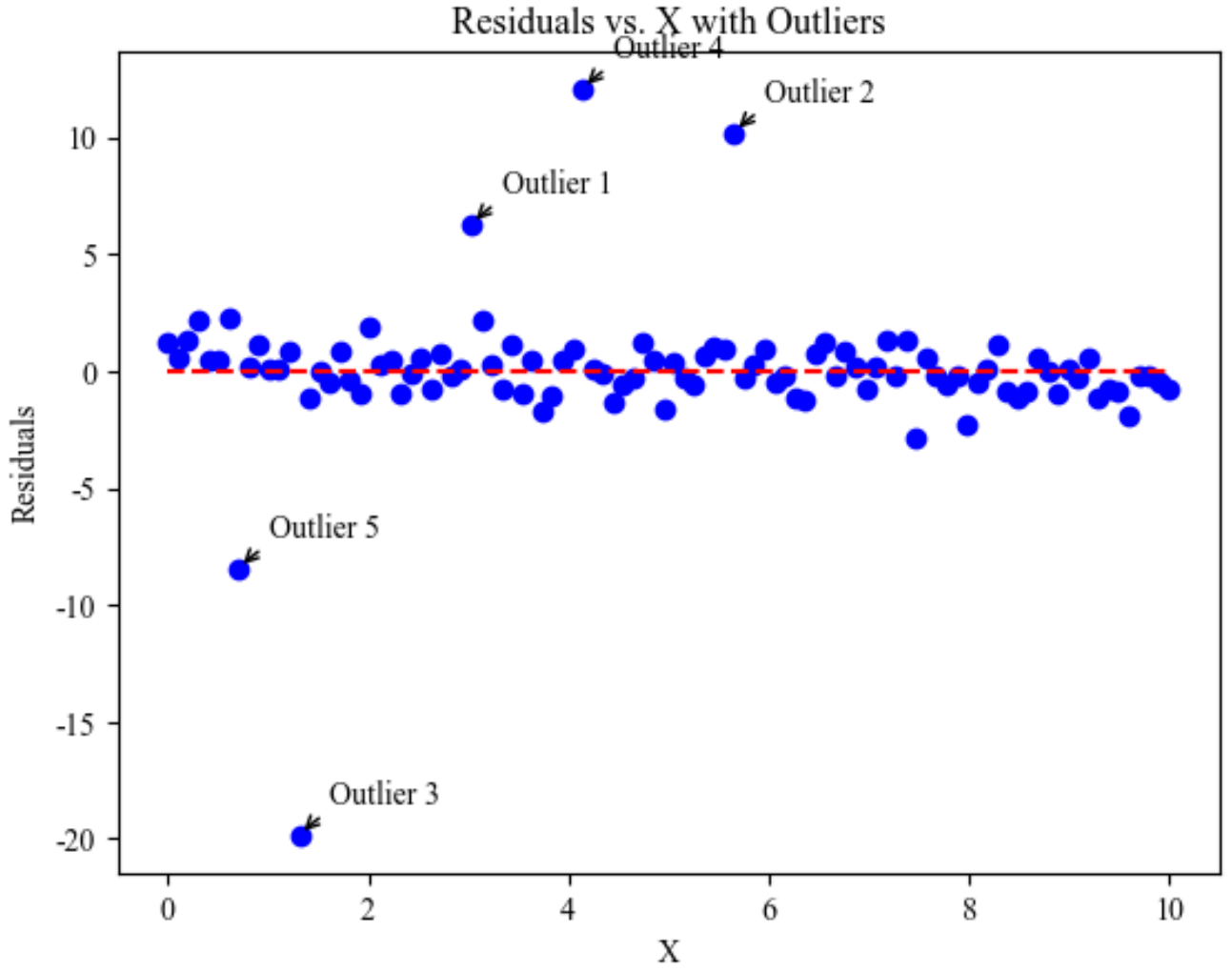
```
X = sm.add_constant(X)
Y = np.log10(Y)

# 拟合线性模型，使用异方差性稳健的标准误差
model = sm.OLS(Y, X)
results_robust = model.fit(cov_type='HC3')
```





一、线性回归（Linear Regression）

- **需要注意的问题：**
  - **离群点**：指对于给定的特征值 x_i 来说，响应值 y_i 异常的点。
  - **学生化残差**：由残差除以它的估计标准误。一般地，学生化残差绝对值大于3的观测点可能是离群点 。如果能确信某个离群点是由数据采集或记录中的错误导致的，那么一个解决方案是直接删除此观测点。

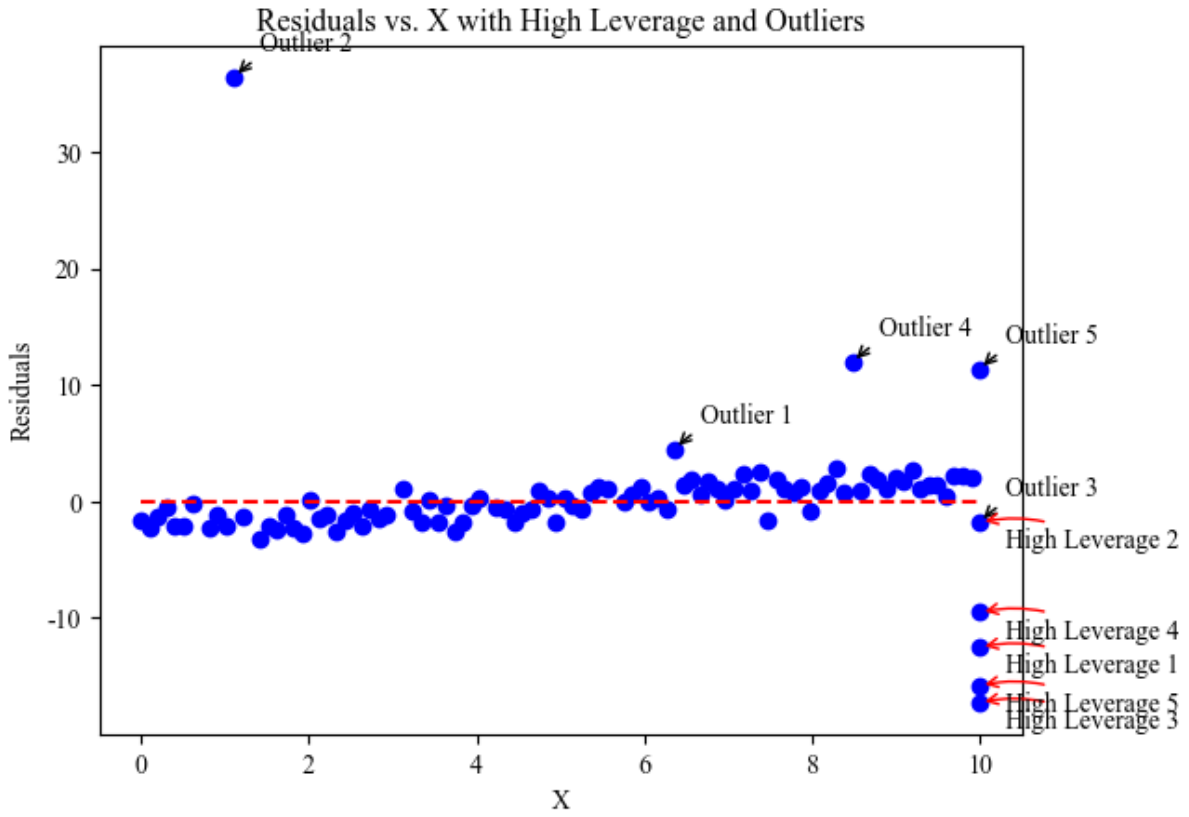


一、线性回归（Linear Regression）

- **需要注意的问题：**
  - **高杠杆点：**观测点 x_i 是异常的 。
  - **杠杆统计量 ：**

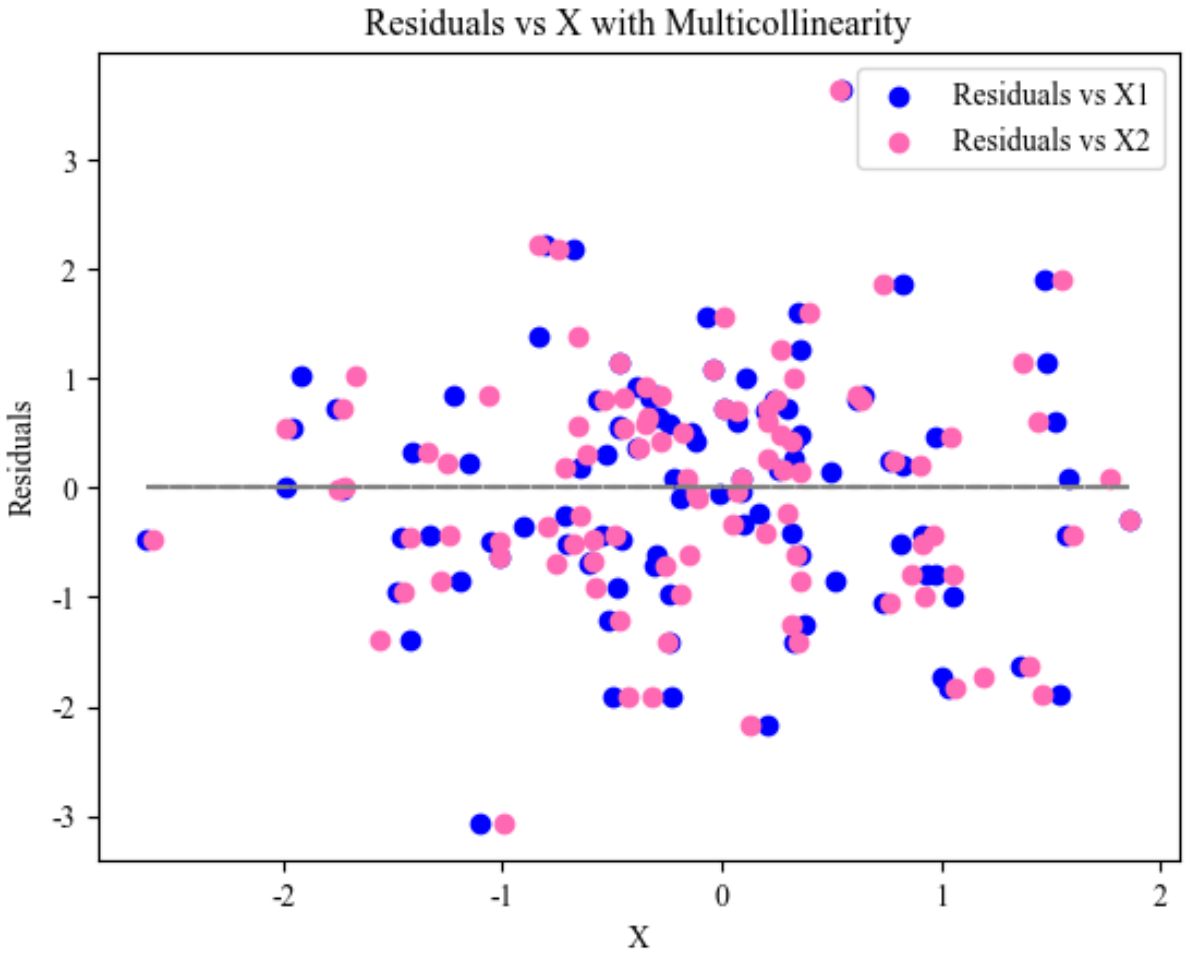
$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n} (x_{i'} - \bar{x})^2}$$

  - 杠杆统计量 的取值总是在 1/n 之间，且所有观测的平均杠杆值总是等于 (p+1)/n 。因此，如果给定观测的杠杆统计量大大超过(p+1)/n 那么我们可能会怀疑对应点有较高的杠杆作用。



一、线性回归（Linear Regression）

- **需要注意的问题：**
  - **共线性**：自变量之间存在高度相关性。模型的估计会变得不稳定，因为微小的数据变化可能导致估计的显著变化。
  - **共线性问题有两种简单的解决方案**：第一种是从回归中剔除一个问题变量；第二种解决方案是把共线变量组合成一个单一的变量，如PCA等。



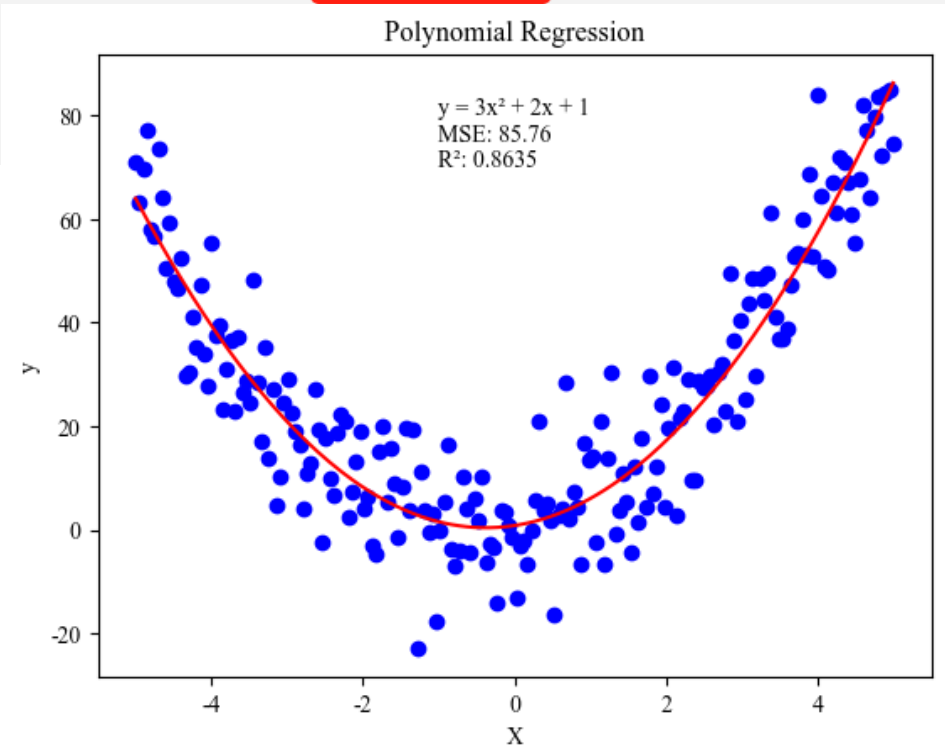
提　纲

二、多项式回归（Polynomial Regression）

- 非线性回归模型。
  - 一元多项式回归：

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

```
# 创建多项式回归模型，选择2次多项式
polynomial_regression = make_pipeline(PolynomialFeatures(degree=2), LinearRegression())

# 拟合模型
polynomial_regression.fit(X, y)
```

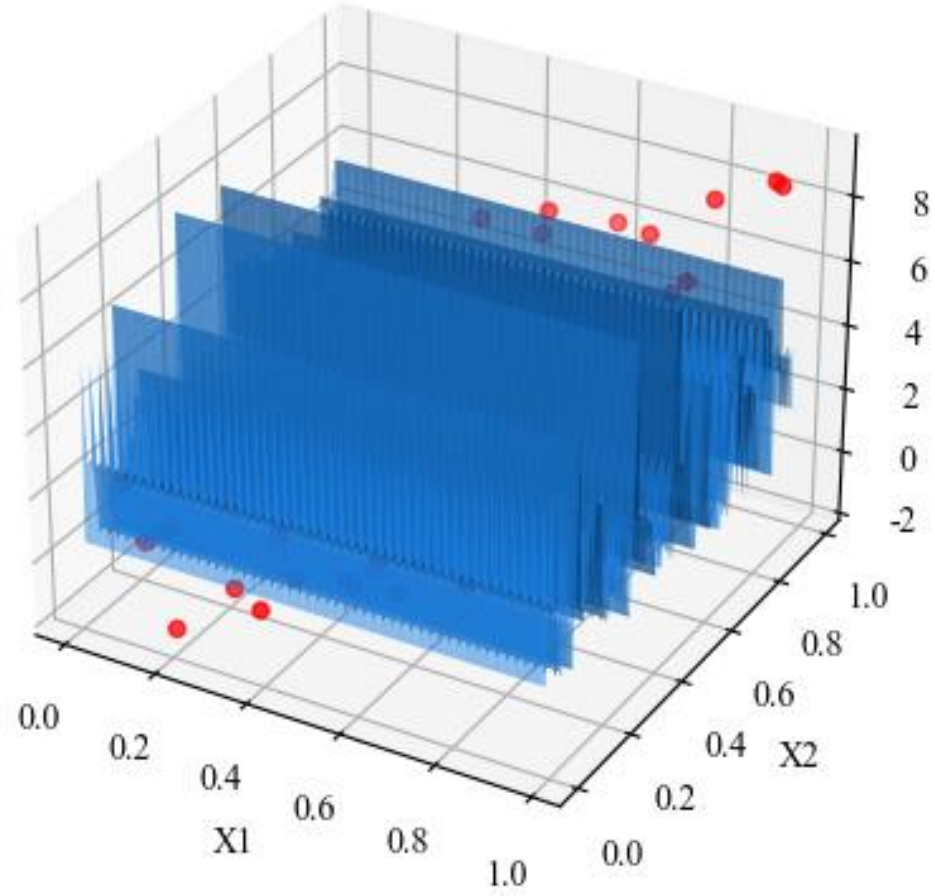


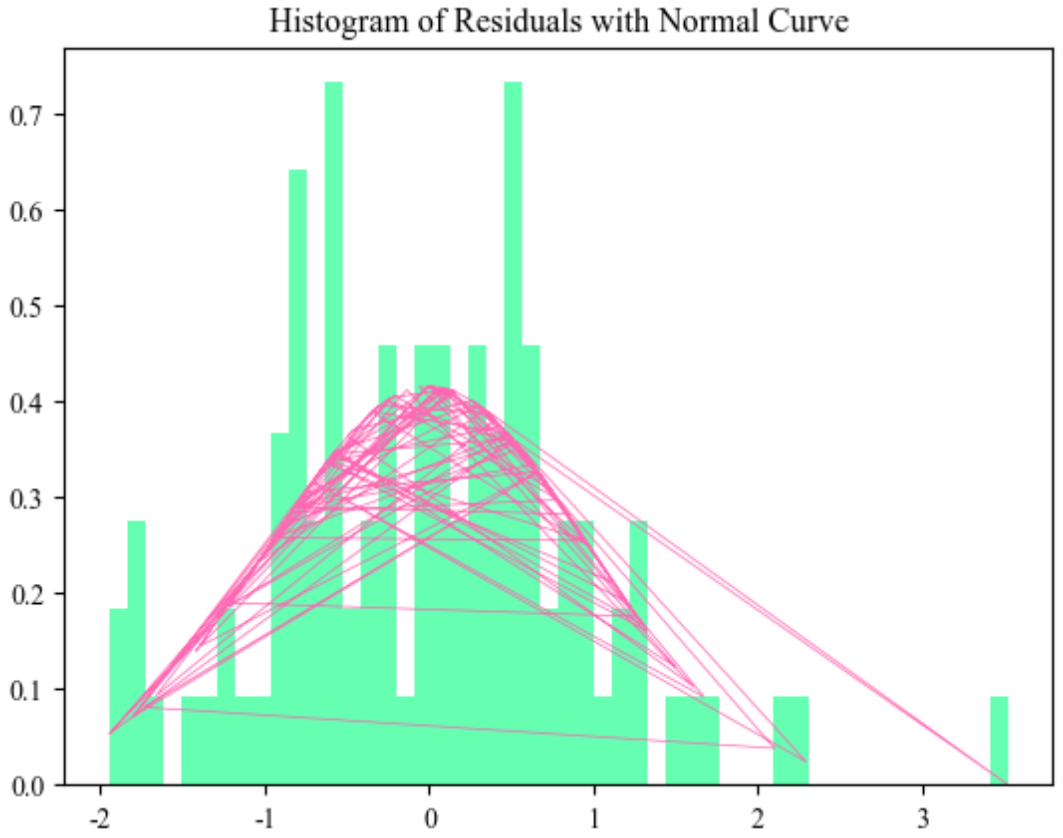
✓ **最小二乘法估计系数**

✓ **比较**：一元多项式回归与多元线性回归，教材221-223页。

二、多项式回归（Polynomial Regression）

- 非线性回归模型。
  - 二元多项式回归：





- 正态性检验

y = 3X1**2 + 2X2**2 + 4X1*X2

提　纲

一　线性回归（Linear Regression）

二　多项式回归（Polynomial Regression）

三　逐步回归（Stepwise Regression）

四　岭回归（Ridge Regression）

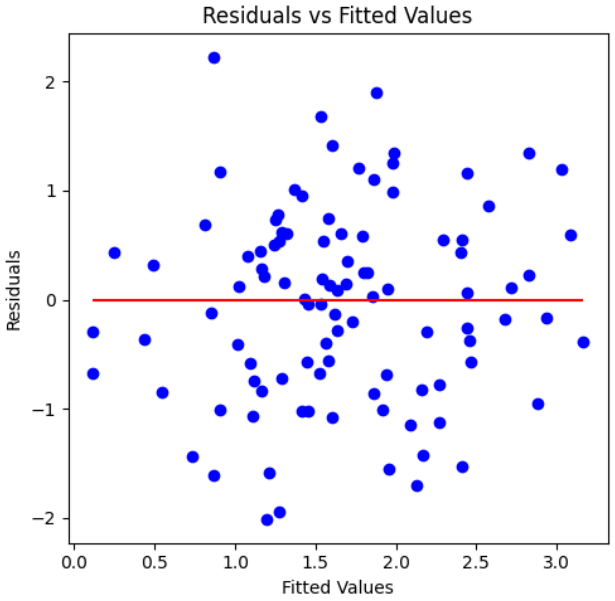
五　套索回归（Lasso Regression）

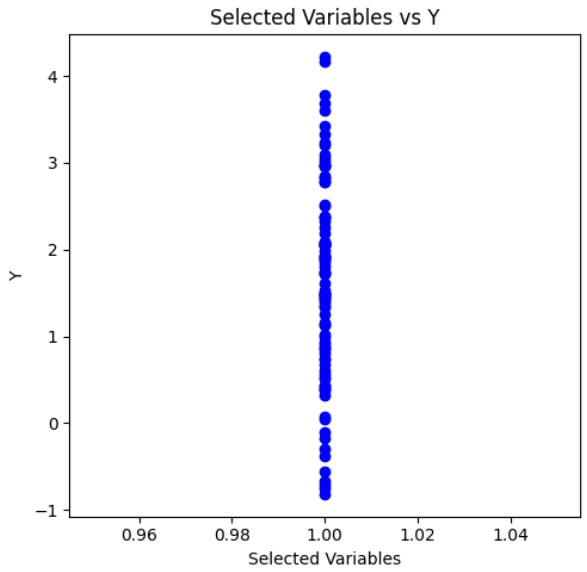
六　贝叶斯回归（Bayesian Regression）

七　随机森林回归（Random Forest Regression）

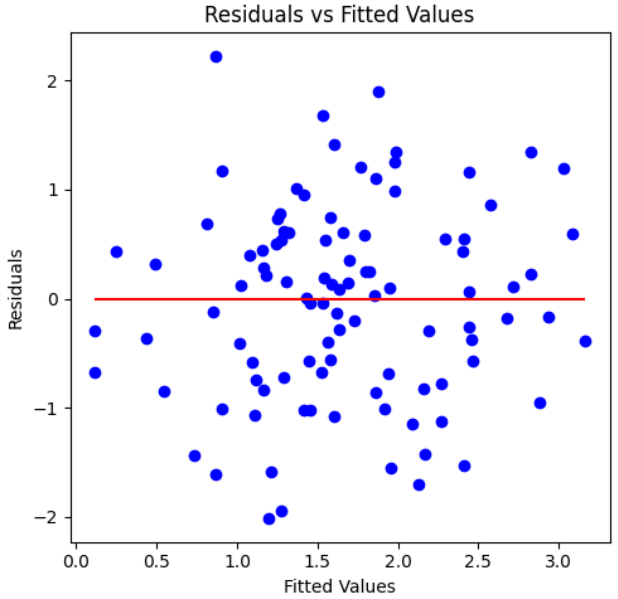
三、逐步回归（Stepwise Regression）

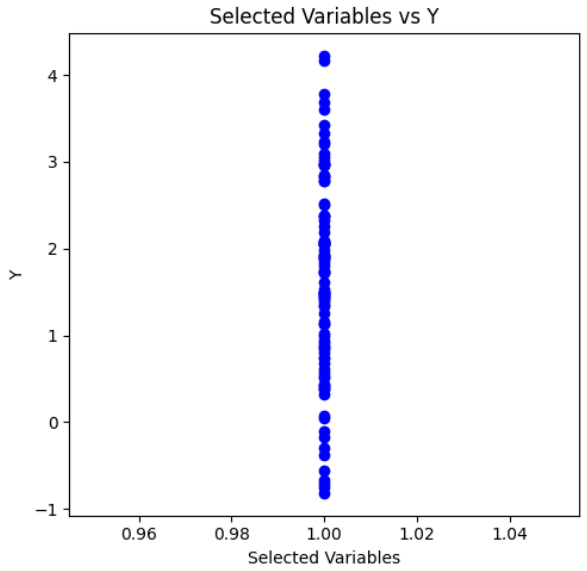
- 在处理多个自变量时常用的非线性回归模型；
- 目的是使用最少的预测变量数来最大化预测能力；
- 最佳子集是模型具有最小的残差平方和；
- 逐步回归法选择变量的过程包含两个基本步骤：
  - ✓ 一是从回归模型中剔出经检验不显著的变量，
  - ✓ 二是引入新变量到回归模型中，常用的逐步回归方法有向前法和向后法。





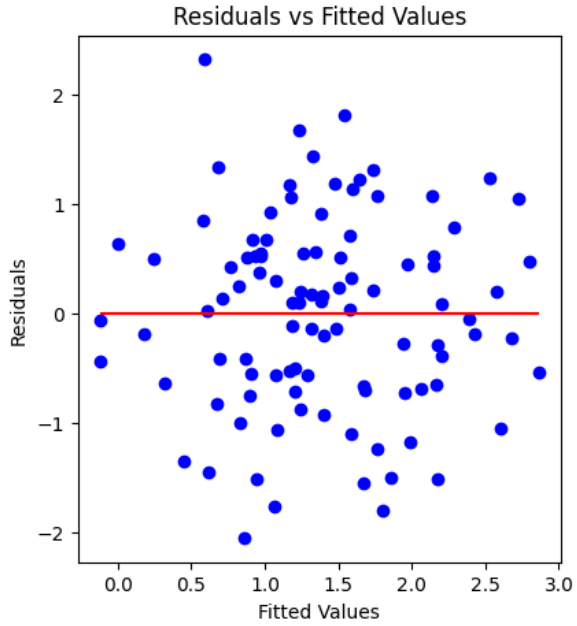
三、逐步回归（Stepwise Regression）

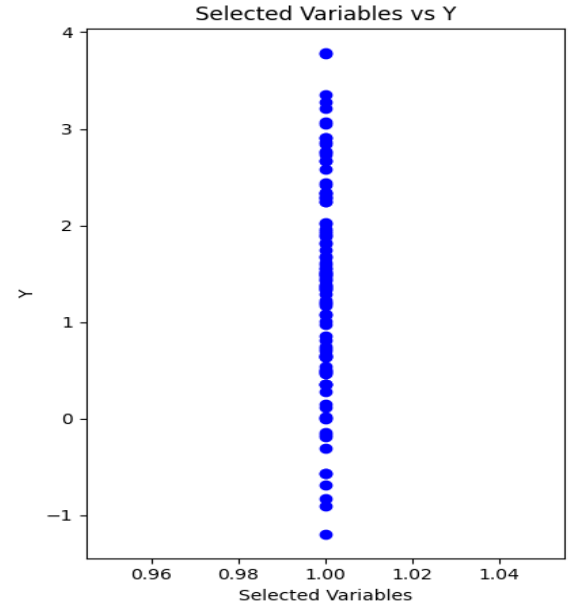




- 向前逐步回归

MSE: 0.7883
R^2: 0.4017





- 向后逐步回归

MSE: 0.7767
R^2: 0.3570

三、逐步回归（Stepwise Regression）

```python
# 向前选择函数
def forward_selection(X, y, p_value_threshold=0.05):
    selected_vars = []
    current_model = sm.OLS(y, sm.add_constant(X[:, 0])).fit()
    while len(selected_vars) < X.shape[1] - 1:
        max_p_value = float('inf')
        variable_to_add = None
        for i in range(0, X.shape[1]):
            if i not in selected_vars:
                model = sm.OLS(y, sm.add_constant(X[:, selected_vars + [i]])).fit()
                p_value = model.pvalues[i-1]
                if p_value < max_p_value:
                    max_p_value = p_value
                    variable_to_add = i

        if max_p_value < p_value_threshold:
            selected_vars.append(variable_to_add)
            current_model = model
        else:
            break
    return current_model, selected_vars
```

向前逐步回归

```python
# 向后逐步回归函数
def backward_stepwise_regression(X, y, p_value_threshold=0.05):
    best_model = sm.OLS(y, X).fit()
    best_rsquared = best_model.rsquared
    best_params = best_model.params
    best_pvalues = best_model.pvalues
    best_X = pd.DataFrame(X,columns=[:])

    while len(best_X.columns) > 1:
        # 找到p值最大的变量
        worst_var = best_pvalues.argmax()
        if worst_var == 0:  # 如果是截距项，则跳过
            break

        # 移除最不显著的变量
        X_reduced = best_X.drop(best_X.columns[worst_var], axis=1)
        model = sm.OLS(y, X_reduced)
        results = model.fit()

        # 检查R方值是否下降
        if results.rsquared >= best_rsquared:
            best_rsquared = results.rsquared
            best_params = results.params
            best_pvalues = results.pvalues
            best_X = X_reduced
        else:
            break
```

向后逐步回归

三、逐步回归（Stepwise Regression）

OLS Regression Results

Dep. Variable:	y	R-squared:	0.167
Model:	OLS	Adj. R-squared:	0.158
Method:	Least Squares	F-statistic:	19.64
Date:	Fri, 26 Apr 2024	Prob (F-statistic):	2.44e-05
Time:	20:38:59	Log-Likelihood:	-146.56
No. Observations:	100	AIC:	297.1
Df Residuals:	98	BIC:	302.3
Df Model:	1		
Covariance Type:	nonrobust		

向前逐步回归

OLS Regression Results

Dep. Variable:	y	R-squared:	0.357
Model:	OLS	Adj. R-squared:	0.344
Method:	Least Squares	F-statistic:	26.93
Date:	Fri, 26 Apr 2024	Prob (F-statistic):	4.98e-10
Time:	21:00:27	Log-Likelihood:	-129.26
No. Observations:	100	AIC:	264.5
Df Residuals:	97	BIC:	272.3
Df Model:	2		
Covariance Type:	nonrobust		

向后逐步回归

提　纲

一　线性回归（Linear Regression）

二　多项式回归（Polynomial Regression）

三　逐步回归（Stepwise Regression）

四　岭回归（Ridge Regression）

五　套索回归（Lasso Regression）

六　贝叶斯回归（Bayesian Regression）

七　随机森林回归（Random Forest Regression）

四、岭回归（Ridge Regression）

- 岭回归是线性回归的重要改进，增加了误差容忍度。
- 如果数据集合矩阵存在多重共线性（数学上称为病态矩阵），那么线性回归模型对输入变量中的噪声非常的敏感，如果输入变量x有一个微小的变动，其反应在输出结果上会变得非常大，方程的解表现出极为不稳定。为了解决这个问题，就有了优化算法——岭回归。
- 岭回归通过对系数的施加惩罚（正则化参数λ）来解决线性回归的一些问题。
- 岭回归的λ参数是一个正则化参数，它控制模型的复杂度。 λ值越大，正则化强度越大，模型越简单。在实际应用中，通常需要通过交叉验证来选择最佳的λ值。

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}$$

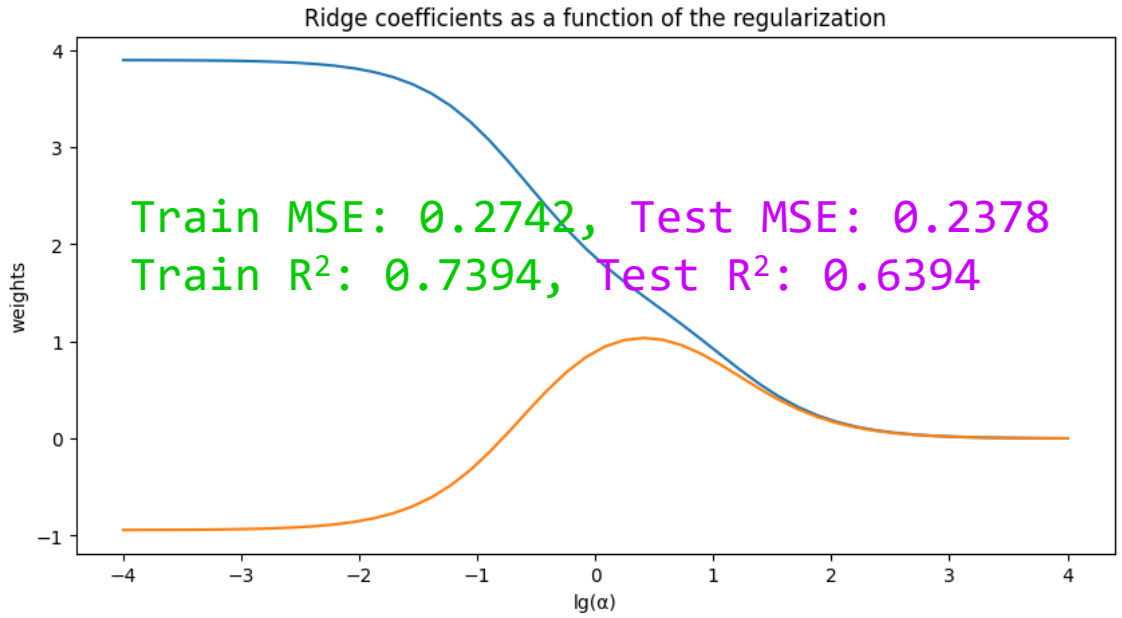
$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

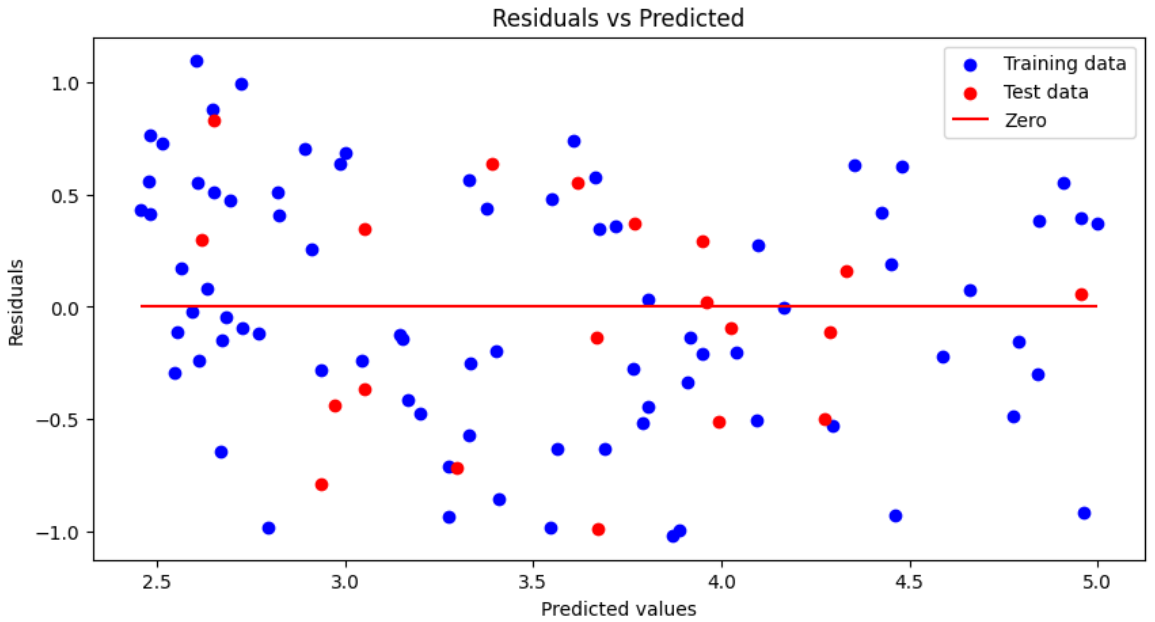
$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 \le t,$$

四、岭回归（Ridge Regression）

```python
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)

# 岭回归模型
ridge_reg = Ridge(alpha=1.0)  # alpha是正则化强度的倒数
ridge_reg.fit(X_train, y_train)

# 预测
y_train_pred = ridge_reg.predict(X_train)
y_test_pred = ridge_reg.predict(X_test)
```

尝试通过交叉验证或网格搜索的方法选择最佳惩罚系数。





提　纲

五、套索回归（Lasso Regression）

- 套索回归与岭回归类似，会对回归系数的绝对值添加一个罚值。此外，它能降低偏差并提高线性回归模型的精度。与岭回归有一点不同，它在惩罚部分使用的是绝对值，而不是平方值。这导致惩罚（即用以约束估计的绝对值之和）值使一些参数估计结果等于零。使用的惩罚值越大，估计值会越趋近于零。

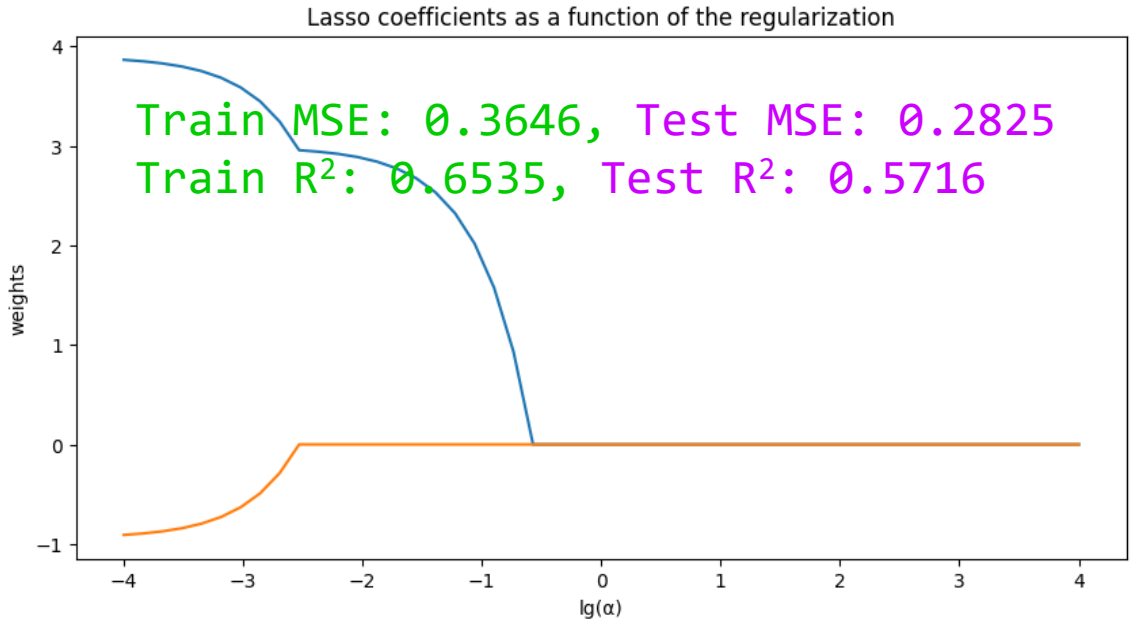
$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

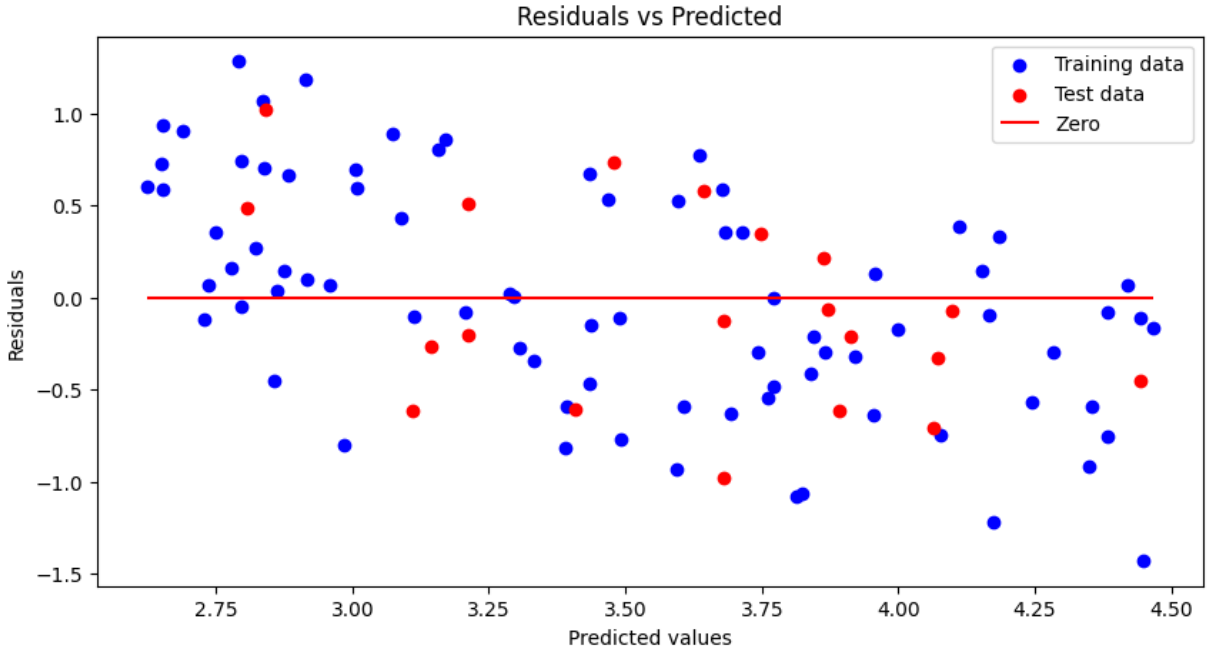
$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2$$

$$\text{subject to} \sum_{j=1}^{p} |\beta_j| \leq t.$$

五、套索回归（Lasso Regression）

```
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X_poly, y, test_size=0.2, random_state=42)

# 套索回归模型
lasso_reg = Lasso(alpha=0.1)  # alpha是正则化强度的倒数
lasso_reg.fit(X_train, y_train)

# 预测
y_train_pred = lasso_reg.predict(X_train)
y_test_pred = lasso_reg.predict(X_test)
```





提　纲

一　线性回归（Linear Regression）

二　多项式回归（Polynomial Regression）

三　逐步回归（Stepwise Regression）

四　岭回归（Ridge Regression）

五　套索回归（Lasso Regression）

六　贝叶斯回归（Bayesian Regression）

七　随机森林回归（Random Forest Regression）

六、贝叶斯回归（Bayesian Regression）

- 贝叶斯回归是一种基于贝叶斯定理的回归分析方法，它提供了对回归系数的后验分布估计，而不仅仅是点估计。
- 假设贝叶斯网络结点包含的属性为{X_1, X_2,…, X_n ,Y}。如果X_i (i =1,…,n)是连续随机变量，并且X_i相对于Y条件独立，则有

$$p(y \mid x_1, x_2, \cdots, x_n) = \frac{p(y, x_1, x_2, \cdots, x_n)}{p(x_1, x_2, \cdots, x_n)} = \frac{p(x_1, x_2, \cdots, x_n \mid y)\, p(y)}{p(x_1, x_2, \cdots, x_n)}$$

$$= \frac{\prod_{i=1}^{n} p(x_i \mid y)\, p(y)}{p(x_1, x_2, \cdots, x_n)} = \alpha \prod_{i=1}^{n} p(x_i \mid y)\, p(y)$$

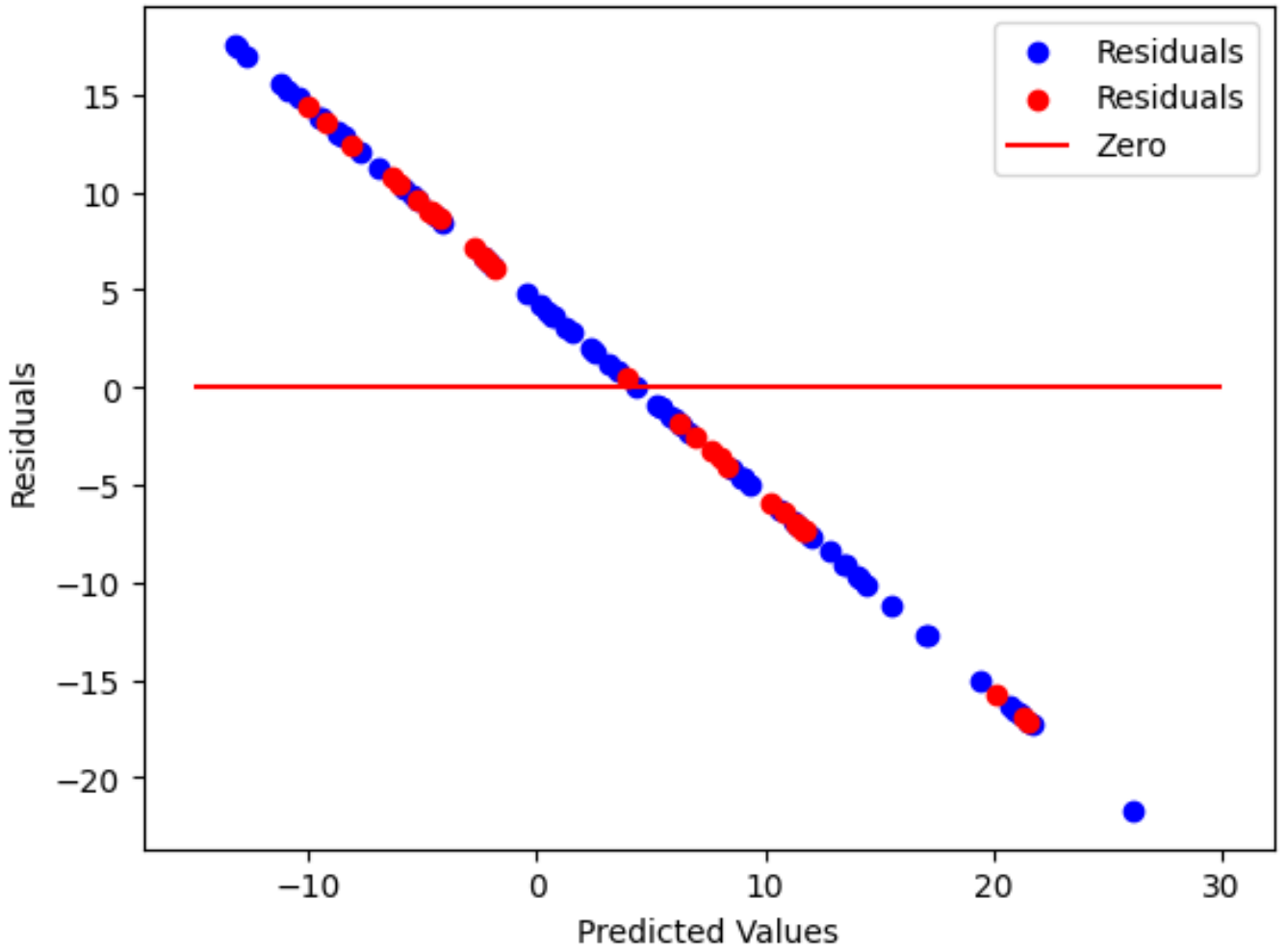
其中，α =1/p(x) 是正则化参数，贝叶斯回归以后验概率密度作为回归分析指示，即输出条件概率密度最大的回归值作为目标值。结合上式可以得到其等价形式：

$$Y^* = \arg\max p(y \mid x_1, x_2, \cdots, x_n) = \arg\max \prod_{i=1}^{n} p(x_i \mid y)\, p(y)$$

亦可作为分类模型

六、贝叶斯回归（Bayesian Regression）

```python
# 步骤3：标准化特征（对于贝叶斯回归很重要）
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 步骤4：使用BayesianRidge进行贝叶斯回归
model = BayesianRidge()
model.fit(X_train_scaled, Y_train)

# 步骤5：评估模型性能
Y_pred_train = model.predict(X_train_scaled)
mse_train = mean_squared_error(Y_train, Y_pred_train)
r2_train = r2_score(Y_train, Y_pred_train)
print(f"Train MSE: {mse_train}")
print(f"Train R^2 score: {r2_train}")

Y_pred = model.predict(X_test_scaled)
mse_test = mean_squared_error(Y_test, Y_pred)
r2_test = r2_score(Y_test, Y_pred)
print(f"Test MSE: {mse_test}")
print(f"Test R^2 score: {r2_test}")
```



优化：马尔可夫链蒙特卡洛（MCMC）采样以估计模型的后验分布。

Train MSE: 105.1293, Test MSE: 85.7537
Train R^2: 0.3731, Test R^2: 0.00523

提　纲

一　线性回归（Linear Regression）

二　多项式回归（Polynomial Regression）

三　逐步回归（Stepwise Regression）

四　岭回归（Ridge Regression）

五　套索回归（Lasso Regression）

六　贝叶斯回归（Bayesian Regression）

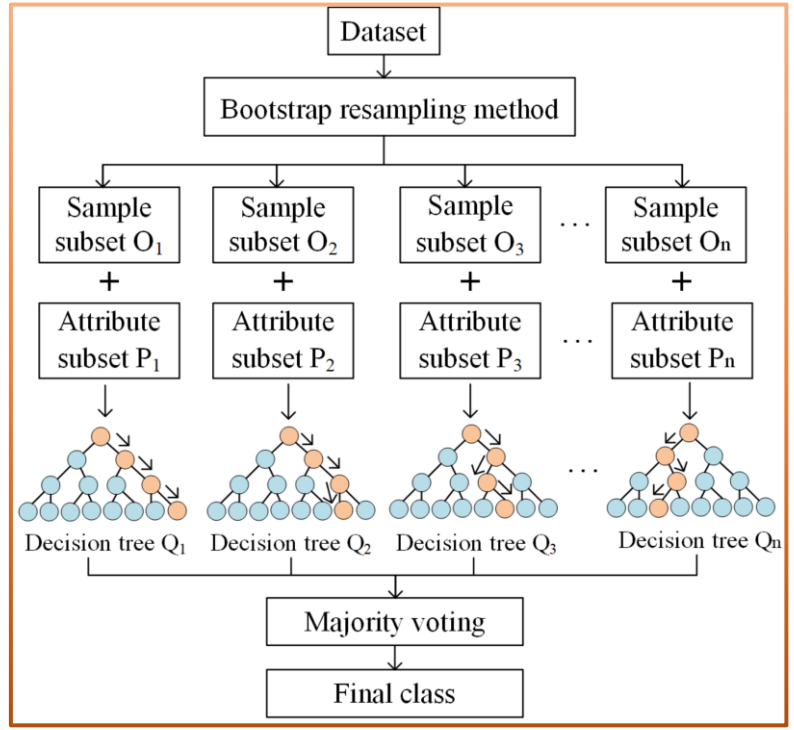
七　随机森林回归（Random Forest Regression）

七、随机森林回归（Random Forest Regression）

- 随机森林以决策树为基础，用随机的方式排列建立，森林里每棵决策树之间都是没有关联的。
- 集成学习的基本思想就是将多个模型组合，从而实现一个更好的预测效果。集成算法大致可以分为：Bagging，Boosting 和 Stacking 三大类型。
- 随机森林属于Bagging集成算法。通过组合多个弱模型，集思广益，使得整体模型具有较高的精确度和泛化性能。
- Bagging是一种在原始数据集上，通过有放回抽样分别选出k个新数据集，来训练模型的集成算法。
- 随机森林可以应用在分类和回归问题上。实现这一点，取决于随机森林的每颗cart树是分类树还是回归树。如果是回归树，则cart树是回归树，采用的原则是最小均方差。

七、随机森林回归（Random Forest Regression）

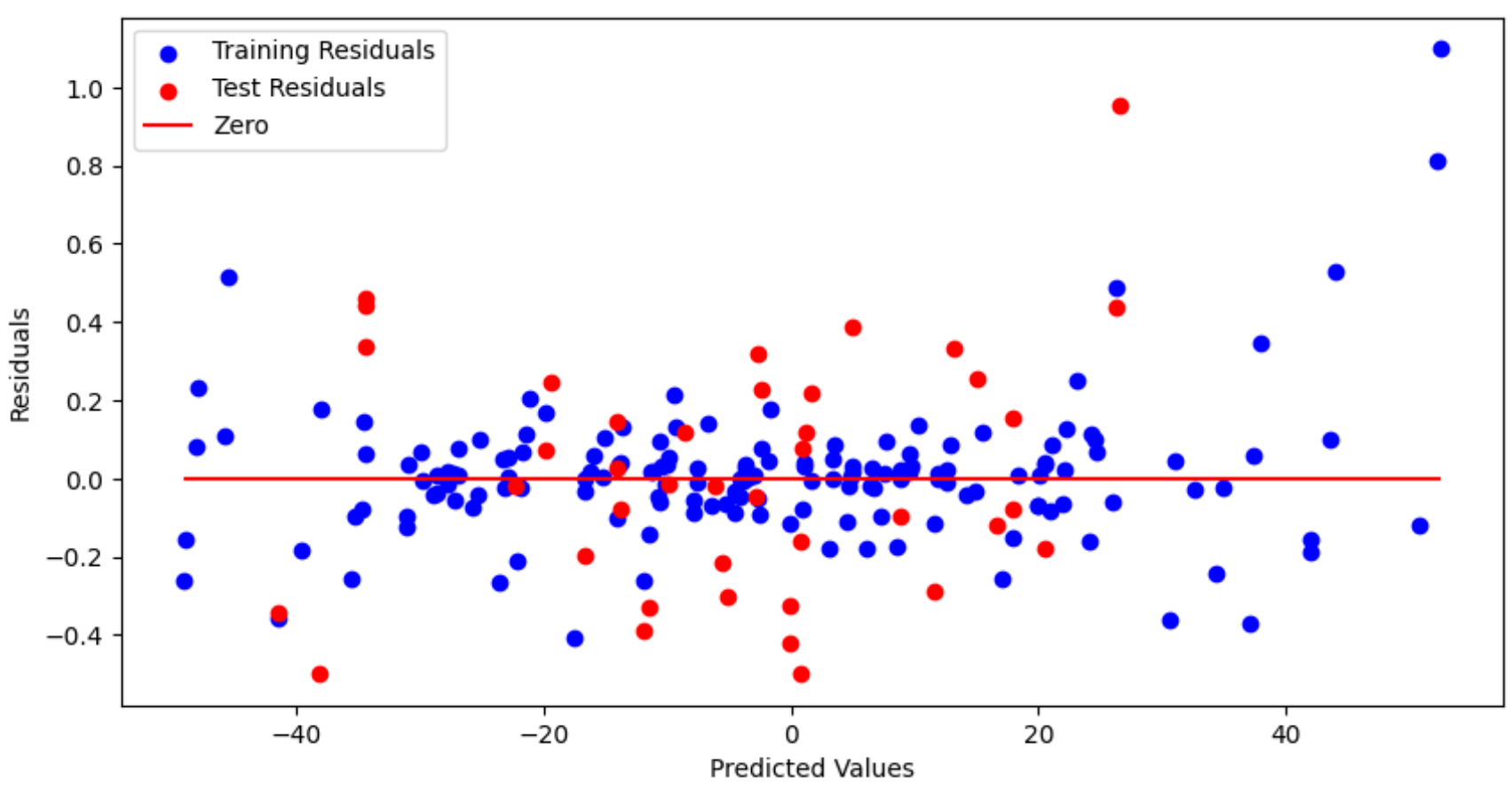
- **随机森林的随机性:**
  - **数据集的随机选取：**从原始的数据集中采取有放回的抽样（bagging），构造子数据集，子数据集的数据量是和原始数据集相同的。不同子数据集的元素可以重复，同一个子数据集中的元素也可以重复。
  - **待选特征的随机选取：**与数据集的随机选取类似，随机森林中的子树的每一个分裂过程并未用到所有的待选特征，而是从所有的待选特征中随机选取一定的特征，之后再在随机选取的特征中选取最优的特征。



七、随机森林回归（Random Forest Regression）

```python
# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 创建随机森林回归模型
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)

# 训练模型
rf_reg.fit(X_train, y_train)

# 进行预测
y_train_pred = rf_reg.predict(X_train)
y_test_pred = rf_reg.predict(X_test)

# 计算精度评价指标
mse_train = mean_squared_error(y_train, y_train_pred)
r2_train = r2_score(y_train, y_train_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
r2_test = r2_score(y_test, y_test_pred)
```

Train MSE: 0.03047, Test MSE: 0.0951
Train R^2: 0.9999, Test R^2: 0.9996



回归模型实验

- **数据：教材270页，习题4。**
- **实验要求：**
  - 根据给定数据构建回归模型；
  - 考虑题目给定情形：固定其中两个变量，例如固定P和K分别为196 kg/ha、372 kg/ha；
  - 考虑不固定N、P、K任一变量的情形；
  - 尝试多种回归模型，并进行模型评估，选择最佳模型；
  - 实验分析与评价。

思考题

1. 简述回归模型的主要步骤，并以一种方法为例说明其求解过程。

2. 简述回归模型的评价与选择。

3. 试比较一元多项式回归与多元回归。

4. 试比较岭回归与套索回归。