现代预测建模：从回归到机器学习

李世纪

2025-11-26

目录

回归、分类模型及 R、Python 实现，作为回归分析和机器学习入门课程讲义、学习手册

前言

写这本书是为了更好的服务教学，现在的《回归分析》教材，内容过时，工具只用 SPSS，并且和《计量经济学》课程内容重复。本书特点在于：
A：增加了机器学习的思想和内容，增加机器学习回归和分类, 同时弱化删除了一部分不再重要的内容，比如适用于计算能力差的时代的逐步回归，和计量经济学重复的用于因果分析的自相关和异方差部分；
B: 工具选择用 R、Pyhon，和科研界、工业界接轨，更实用，更容易和工作衔接。
C：互动性教学，基于软件的进步，可以使用 shiny 等进行互动性分析，读者可以更直观的看到参数变化对分析结果的影响

版本信息：
使用 Quarto 出版，结果可以是 html、docx、PDF 等形式。内容和代码部分主要是 markdown 格式，可移植性、可编译性都较好。quarto 版本为 1.8.26，R 版本为 4.5.2，Python 版本为 3.13.5，使用 Positron 或者 Rstudio 编译。

在书的第一部分，软件工具以 R 为主，同时引入 Python。R 在分析传统统计学模型、因果效应分析的时候具有优势，毕竟 R 是统计学家和数据学家的语言，第二部分，R、Python 并重，读者可以看到两者面对同样问题时的不同表现，在第三部分部分，机器学习部分，特别是非线性分析，以 Python 的 sklearn 分析框架为主，也是事实上的主流机器学习工具，同时也加入了 R 的 mlr3 工具包，其基准测试具有快速识别模型，批量运算对比的优势。可以认为 R 在处理传统统计学问题、线性分析模型比如因果分析、参数估计、假设检验和时间序列数据上具有优势，绘图功能是其特长；Python 在机器学习领域、非线性分析优势不可替代，其统一的输入接口和输出格式，具备强大的生命力，而且在数据清洗领域优于 R。本书综合使用两者优点，同时提供单 R 或者单 Python 的解决方案，读者根据自己的需要组合使用或者取舍。

I 基础-线性回归模型

第一部分我们将介绍线性回归分析的基本假设和方法。随后我们将扩展到更复杂的统计建模技术，包括以下内容：- 基本概念 - 线性回归模型的假设检验 - 参数估计方法和假设检验 - 异方差 - 自相关

1 预测模型与评估

本章导读

预测建模是现代数据分析的核心任务，无论是预测连续数值（回归）还是预测类别标签（分类），都需要建立系统的分析和评估框架。本章将介绍预测建模的基本概念，重点讨论过拟合现象及其危害，并建立完整的模型评估体系，，补充了工具软件 R、Python 的相关使用说明，为后续学习各种预测模型奠定基础。

1.1 预测建模的基本概念

1.1.1 回归与分类任务

回归分析：回归任务旨在预测连续型数值变量。例如：- 预测房屋价格（连续金额）- 预测股票收益率（连续百分比）- 预测气温变化（连续温度值）- 预测销售额（连续数量）

数学上，回归模型试图建立从特征变量 X 到连续响应变量 Y 的映射：

$$Y = f(X) + \varepsilon$$

其中 ε 是随机误差项，代表模型无法解释的部分。

分类分析：分类任务旨在预测离散型类别变量。

例如：

- 判断邮件是否为垃圾邮件（二分类：是/否）
- 诊断患者是否患病（二分类：健康/患病）
- 识别图像中的物体类别（多分类：猫/狗/车等）
- 信用风险评估（多分类：低风险/中风险/高风险）

分类模型建立从特征变量 X 到类别标签 Y 的映射：

$$Y = g(X), \quad Y \in \{C_1, C_2, \dots, C_K\}$$

1.1.2 特征变量与响应变量

特征变量（自变量）：用于预测的输入变量

- 可以是数值型、分类型或顺序型
- 表示为 $X = (X_1, X_2, \dots, X_p)$

特征工程：从原始数据中提取有意义的特征

响应变量（因变量）：

- 需要预测的目标变量
- 回归问题中为连续数值
- 分类问题中为离散类别
- 应该与业务目标紧密相关

示例分析：以房价预测为例：

- 特征变量：房屋面积、卧室数量、地理位置、房龄等
- 响应变量：房屋价格（连续值）
- 问题类型：回归问题

以垃圾邮件检测为例：

- 特征变量：邮件关键词频率、发件人信誉、邮件长度等
- 响应变量：邮件类型（垃圾邮件/正常邮件）
- 问题类型：二分类问题

1.1.3 预测建模的流程

完整工作流：

1. 问题定义：
    - 明确业务目标

- 确定预测任务类型（回归/分类）
- 制定成功标准

2. 数据准备：
   - 数据收集与整合
   - 数据清洗与预处理
   - 探索性数据分析
3. 特征工程：
   - 特征选择
   - 特征变换
   - 特征创建
4. 模型选择：
   - 根据问题特点选择算法家族
   - 考虑数据规模和特征类型
   - 评估模型假设的合理性
5. 模型训练：
   - 参数估计
   - 超参数调优
   - 模型验证
6. 模型评估：
   - 性能指标计算
   - 模型比较
   - 误差分析
7. 模型部署：
   - 生产环境集成
   - 性能监控
   - 模型更新

1.2 过拟合与模型复杂度

1.2.1 过拟合的概念与机制

过拟合的本质：模型过度适应训练数据中的特定模式（包括噪声），导致在新数据上泛化能力下降。

过拟合的数学描述：设 f 为真实函数，$\hat{f}$ 为估计函数，则：

- 训练误差：$\text{Err}_{\text{train}} = E[L(Y, \hat{f}(X))|\text{train}]$
- 测试误差：$\text{Err}_{\text{test}} = E[L(Y, \hat{f}(X))|\text{test}]$

当 $Err_{train} \ll Err_{test}$ 时，发生过拟合。

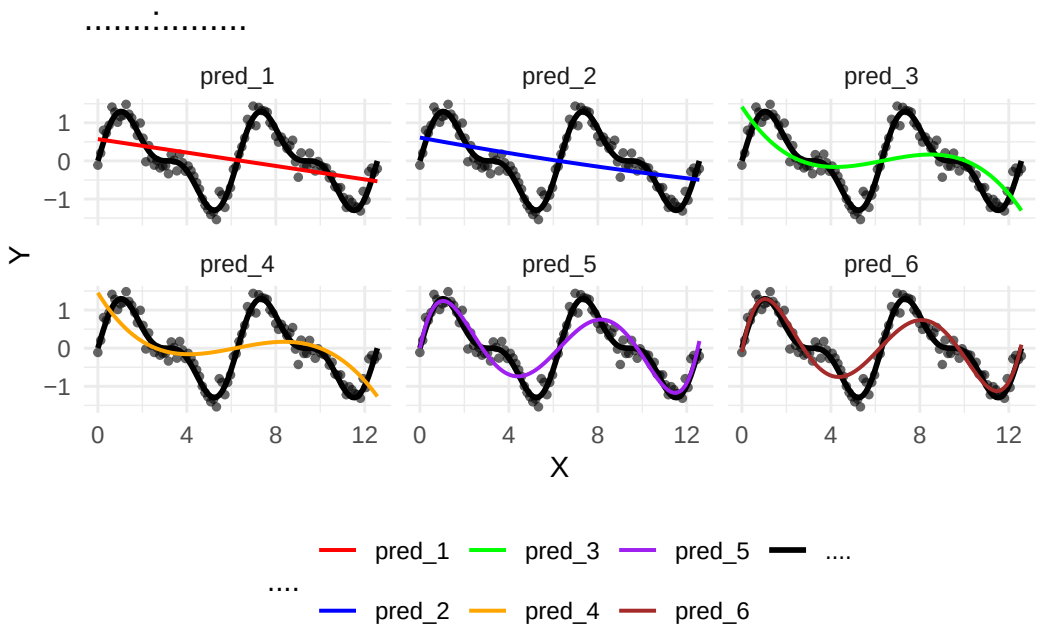
过拟合的视觉示例：考虑多项式回归：- 适当阶数：平滑曲线，很好拟合真实趋势 - 过高阶数：曲线剧烈波动，拟合每个数据点（包括噪声）

欠拟合：与过拟合相对，模型过于简单，无法捕捉数据中的基本模式：- Err_{train} 很大 - Err_{test} 也很大 - 模型能力不足

编程演示：过拟合现象

```
# 过拟合现象演示 - 生成数据
# 生成理想的过拟合演示数据
generate_overfitting_demo_data <- function(n = 100) {
  set.seed(123)
  x <- seq(0, 4 * pi, length.out = n)
  # 真实关系：正弦函数 + 噪声
  y_true <- sin(x) + 0.5 * sin(2*x)
  y_observed <- y_true + rnorm(n, 0, 0.2)
  return(data.frame(x = x, y_true = y_true, y_observed = y_observed))
}

# 生成数据
demo_data <- generate_overfitting_demo_data(100)

# 拟合不同复杂度的多项式模型
library(ggplot2)
fit_polynomial_models <- function(data, max_degree = 6) {
  models <- list()
  predictions <- data.frame(x = data$x)

  for (degree in 1:max_degree) {
    model <- lm(y_observed ~ poly(x, degree), data = data)
    models[[paste0("degree_", degree)]] <- model
    predictions[[paste0("pred_", degree)]] <- predict(model, newdata = data)
  }

  return(list(models = models, predictions = predictions))
}

# 拟合模型
```

```r
results <- fit_polynomial_models(demo_data)

# 可视化过拟合现象
plot_overfitting_demo <- function(data, predictions) {
  plot_data <- cbind(data, predictions)
  plot_data_long <- reshape2::melt(plot_data, id.vars = c("x", "y_true", "y_
                                   variable.name = "model", value.name = "pre

  ggplot(plot_data_long, aes(x = x)) +
    geom_point(aes(y = y_observed), alpha = 0.6, size = 1) +
    geom_line(aes(y = y_true, color = " 真实关系"), size = 1) +
    geom_line(aes(y = prediction, color = model), size = 0.7) +
    facet_wrap(~ model, ncol = 3) +
    labs(title = " 过拟合现象演示：不同阶数多项式回归",
         x = "X", y = "Y",
         color = " 曲线类型") +
    theme_minimal() +
    scale_color_manual(values = c(" 真实关系" = "black",
                                  "pred_1" = "red", "pred_2" = "blue",
                                  "pred_3" = "green", "pred_4" = "orange",
                                  "pred_5" = "purple", "pred_6" = "brown")) +
    theme(legend.position = "bottom")
}

# 显示图形
plot_overfitting_demo(demo_data, results$predictions)
```



1.2.2 偏差-方差权衡的深入分析

回忆《数理统计》里参数估计的评价标准：无偏性：估计量的期望等于被估计参数。有效性：在所有无偏估计量中，方差最小。一致性：随着样本量增加，估计量趋近于被估计参数。在预测建模中，我们关注的是预测误差的来源。偏差-方差权衡（Bias-Variance Tradeoff）是理解模型复杂度与预测性能关系的关键概念。

期望预测误差的严格分解：

对于回归问题，在平方损失下：

$$\begin{aligned} E[(Y - \hat{f}(X))^2] &= E[(Y - E[Y|X] + E[Y|X] - \hat{f}(X))^2] \\ &= E[(Y - E[Y|X])^2] + E[(E[Y|X] - \hat{f}(X))^2] \\ &= \text{Var}(Y|X) + \text{Bias}^2(\hat{f}(X)) + \text{Var}(\hat{f}(X)) \end{aligned}$$

偏差：- $\text{Bias}(\hat{f}(X)) = E[\hat{f}(X)] - f(X)$ - 系统性错误，模型与真实关系的差距 - 高偏差原因：模型过于简单，假设太强

方差：- $\text{Var}(\hat{f}(X)) = E[(\hat{f}(X) - E[\hat{f}(X)])^2]$ - 模型对训练数据变化的敏感性 - 高方差原因：模型过于复杂，对噪声敏感

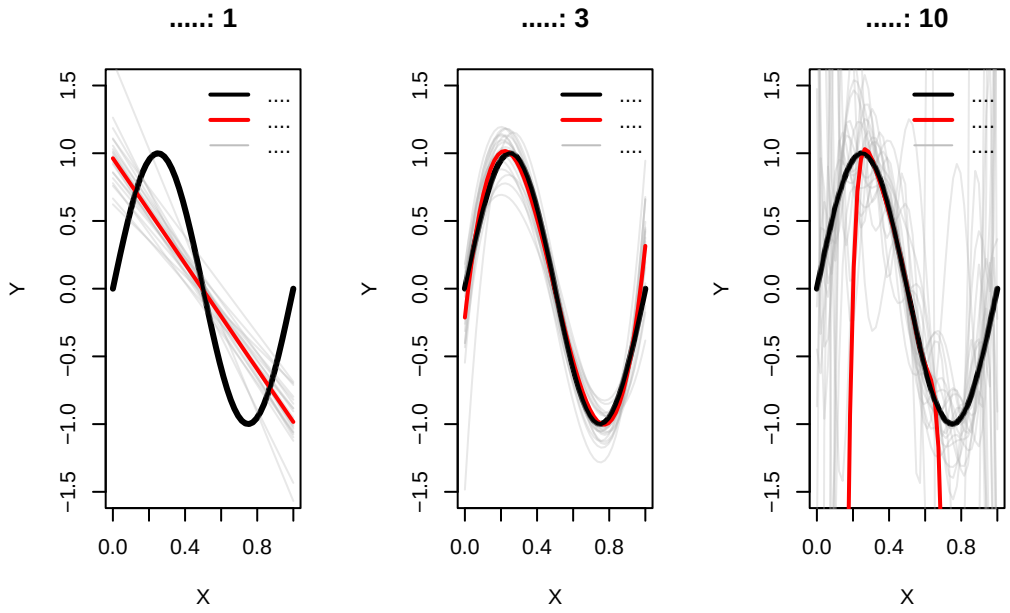
不可约误差：- $\sigma_\varepsilon^2 = \text{Var}(Y|X)$ - 数据内在的随机性 - 无法通过改进模型减少

编程演示：偏差-方差权衡

```r
# 1.2.2 偏差-方差权衡演示
demonstrate_bias_variance <- function() {
  set.seed(123)
  n_simulations <- 100
  x <- seq(0, 1, length.out = 50)
  true_function <- function(x) sin(2 * pi * x)

  # 生成多个训练集
  generate_training_set <- function() {
    x_train <- runif(20, 0, 1)
    y_train <- true_function(x_train) + rnorm(20, 0, 0.3)
    return(data.frame(x = x_train, y = y_train))
  }

  # 拟合不同复杂度的模型
  fit_models <- function(data, degree) {
    if (degree == 1) {
      lm(y ~ x, data = data)
    } else {
      lm(y ~ poly(x, degree), data = data)
    }
  }

  # 计算偏差和方差
  results <- list()
  degrees <- c(1, 3, 10)  # 低、中、高复杂度

  for (degree in degrees) {
    predictions <- matrix(0, nrow = length(x), ncol = n_simulations)

    for (i in 1:n_simulations) {
      train_data <- generate_training_set()
      model <- fit_models(train_data, degree)
      pred <- predict(model, newdata = data.frame(x = x))
      predictions[, i] <- pred
    }

    # 计算偏差和方差
```

```
    mean_pred <- rowMeans(predictions)
    bias_sq <- mean((mean_pred - true_function(x))^2)
    variance <- mean(apply(predictions, 1, var))

    results[[as.character(degree)]] <- list(
      bias_sq = bias_sq,
      variance = variance,
      total_error = bias_sq + variance + 0.3^2,  # 加上噪声方差
      predictions = predictions,
      mean_pred = mean_pred
    )
  }

  return(results)
}

# 运行演示
bias_variance_results <- demonstrate_bias_variance()

# 可视化结果
plot_bias_variance <- function(results, x) {
  true_y <- sin(2 * pi * x)

  par(mfrow = c(1, 3))
  for (i in 1:3) {
    degree <- c(1, 3, 10)[i]
    res <- results[[as.character(degree)]]

    plot(x, true_y, type = "l", lwd = 3, col = "black",
         ylim = c(-1.5, 1.5), main = paste(" 多项式阶数:", degree),
         xlab = "X", ylab = "Y")

    # 绘制多个拟合曲线
    for (j in 1:20) {
      lines(x, res$predictions[, j], col = rgb(0.7, 0.7, 0.7, 0.3))
    }

    # 绘制平均预测和真实函数
```

```
    lines(x, res$mean_pred, col = "red", lwd = 2)
    lines(x, true_y, col = "black", lwd = 2)

    # 添加图例
    legend("topright",
           legend = c(" 真实函数", " 平均预测", " 单个拟合"),
           col = c("black", "red", "gray"), lwd = c(2, 2, 1), bty = "n")
  }
  par(mfrow = c(1, 1))
}

# 创建 x 值用于绘图
x_plot <- seq(0, 1, length.out = 50)
plot_bias_variance(bias_variance_results, x_plot)
```




```
# 打印偏差-方差分解结果
cat(" 偏差-方差分解结果:\n")
```

偏差-方差分解结果:

```
for (degree in c(1, 3, 10)) {
  res <- bias_variance_results[[as.character(degree)]]
  cat(sprintf(" 阶数 %d: 偏差 ²=%.4f, 方差 =%.4f, 总误差 =%.4f\n",
              degree, res$bias_sq, res$variance, res$total_error))
```

```
}
```

```
阶数  1: 偏差²=0.2124, 方差=0.0309, 总误差=0.3333
阶数  3: 偏差²=0.0068, 方差=0.0269, 总误差=0.1236
阶数 10: 偏差²=479112.8409, 方差=48825232.2140, 总误差=49304345.1449
```

1.2.3 模型复杂度的影响分析

复杂度与误差的关系：

复杂度状态	训练误差	测试误差	偏差	方差
低复杂度	高	高	高	低
最优复杂度	中等	最低	中等	中等
高复杂度	很低	高	低	高

学习曲线分析：绘制训练集大小与误差的关系：- 训练误差：随样本增加而增加（趋于稳定）- 测试误差：随样本增加而减少（趋于稳定）- 两者差距：反映方差大小

实际指导意义：- 高偏差：增加模型复杂度，增加特征 - 高方差：减少模型复杂度，增加数据，正则化

1.3 模型评估框架

1.3.1 数据划分的详细策略

为什么需要数据划分：- 评估模型泛化能力 - 避免对训练数据的过度优化 - 提供无偏的性能估计

训练集：- 用于模型参数估计 - 应该足够大以学习数据模式 - 通常占 60-80%

验证集：- 用于模型选择和超参数调优 - 提供模型比较的基础 - 通常占 10-20%

测试集：- 用于最终模型评估 - 在整个建模过程中只能使用一次 - 通常占 10-20%

数学表达：将数据集 D 划分为：

$$D = D_{\text{train}} \cup D_{\text{val}} \cup D_{\text{test}}$$

其中：

- $D_{\text{train}} \cap D_{\text{val}} = \emptyset$

- $D_{\text{train}} \cap D_{\text{test}} = \emptyset$
- $D_{\text{val}} \cap D_{\text{test}} = \emptyset$

特殊情况的处理：

- - 小样本：使用交叉验证
- - 时间序列：按时间顺序划分
- - 不平衡数据：分层抽样

编程演示：数据划分策略

```r
# 1.3.1 数据划分策略演示
demonstrate_data_splitting <- function() {
  set.seed(123)
  n <- 1000
  # 生成回归数据
  x1 <- rnorm(n)
  x2 <- rnorm(n)
  y <- 2*x1 + 3*x2 + rnorm(n, 0, 1)
  data <- data.frame(x1 = x1, x2 = x2, y = y)

  # 简单划分（70% 训练, 30% 测试）
  train_idx_simple <- sample(1:n, size = 0.7 * n)
  test_idx_simple <- setdiff(1:n, train_idx_simple)

  cat(" 简单划分结果:\n")
  cat(" 训练集大小:", length(train_idx_simple), "\n")
  cat(" 测试集大小:", length(test_idx_simple), "\n")

  # 检查分布是否相似
  cat("\n训练集和测试集分布比较:\n")
  cat(" 训练集 y 均值:", mean(data$y[train_idx_simple]), "\n")
  cat(" 测试集 y 均值:", mean(data$y[test_idx_simple]), "\n")

  return(list(
    data = data,
    train_idx = train_idx_simple,
    test_idx = test_idx_simple
  ))
```

```
}

# 运行演示
split_results <- demonstrate_data_splitting()
```

```
简单划分结果：
训练集大小： 700
测试集大小： 300

训练集和测试集分布比较：
训练集y均值： 0.1586161
测试集y均值： 0.09502576
```

1.3.2 交叉验证方法

k 折交叉验证的完整流程：

1. 数据准备：
   - 随机打乱数据
   - 将数据分为 k 个大小相等的子集 $D_1, D_2, \ldots, D_k$
2. 交叉验证循环：For $i = 1$ to k:
   - 训练集：$D_{\text{train}}^{(i)} = D \setminus D_i$
   - 验证集：$D_{\text{val}}^{(i)} = D_i$
   - 在 $D_{\text{train}}^{(i)}$ 上训练模型
   - 在 $D_{\text{val}}^{(i)}$ 上计算验证误差 $\text{Err}^{(i)}$
3. 结果汇总：
   - 平均验证误差：$CV(k) = \frac{1}{k} \sum_{i=1}^{k} \text{Err}^{(i)}$
   - 误差标准差：$SE_{CV} = \sqrt{\frac{1}{k-1} \sum_{i=1}^{k} (\text{Err}^{(i)} - CV(k))^2}$

k 值选择：

- - $k = 5$ 或 $k = 10$：常用选择
- - $k = n$（留一法）：计算量大，方差可能较高
- - 小 k 值：偏差较小，方差较大 - 大 k 值：偏差较大，方差较小

分层 k 折交叉验证：对于分类问题，保持每个折中类别比例与整体一致。

时间序列交叉验证：对于时间相关数据，确保验证集时间在训练集之后。

1.4 评估指标

1.4.1 回归问题评估指标

均方误差：

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

- 优点：数学性质好，便于优化

- 缺点：对异常值敏感，量纲与原始数据不同

均方根误差：

$$RMSE = \sqrt{MSE}$$

- 优点：量纲与原始数据相同

  - 缺点：仍然对异常值敏感

平均绝对误差：

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- 优点：对异常值不敏感

- 缺点：数学性质较差，优化困难

平均绝对百分比误差：

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

- 优点：相对误差，易于解释

- 缺点：$y_i = 0$ 时无法计算，对负值处理困难

决定系数 R^2：

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- 优点：标准化，易于比较

- 缺点：随特征增加而增加，可能误导

调整 R^2：

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(n-1)}{n-p-1}$$

其中 p 是特征数，惩罚特征过多的模型。

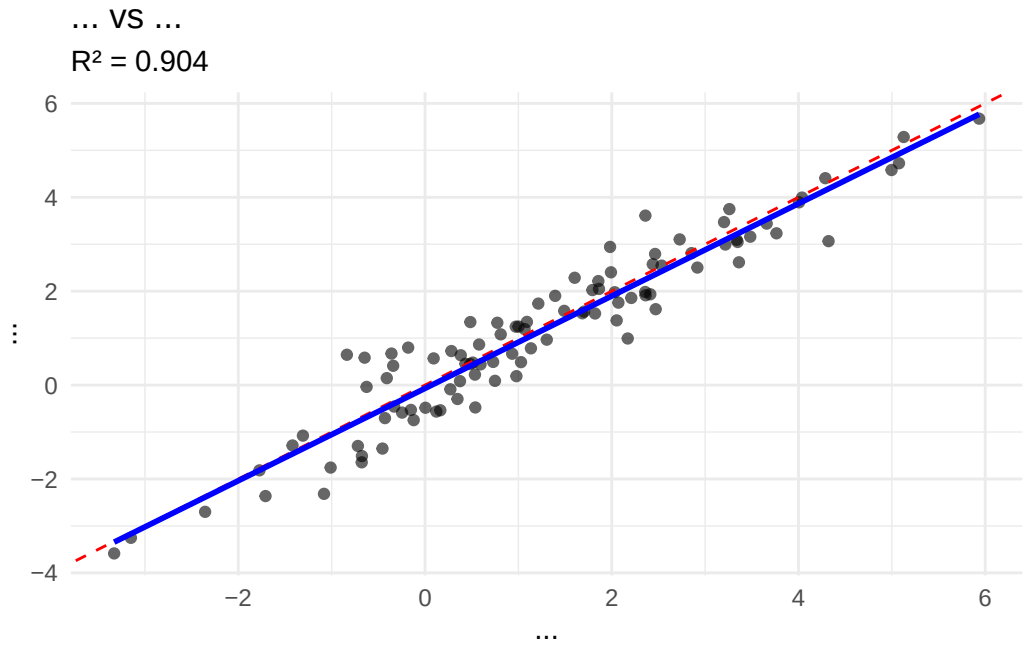
** 演示：回归评估指标 **

```
# 1.4.1 回归评估指标演示
demonstrate_regression_metrics <- function() {
  set.seed(123)
  n <- 100
  # 生成数据
  x <- rnorm(n)
  y_true <- 2*x + 1
  y_pred <- y_true + rnorm(n, 0, 0.5)  # 预测值（带有误差）
  y_actual <- y_true + rnorm(n, 0, 0.3)  # 实际观测值（带有噪声）

  # 计算各种评估指标
  calculate_regression_metrics <- function(actual, predicted) {
    n <- length(actual)
    residuals <- actual - predicted

    mse <- mean(residuals^2)
    rmse <- sqrt(mse)
    mae <- mean(abs(residuals))

    # R-squared
    ss_residual <- sum(residuals^2)
    ss_total <- sum((actual - mean(actual))^2)
    r_squared <- 1 - (ss_residual / ss_total)

    # 调整 R-squared
    p <- 1  # 特征数
    r_squared_adj <- 1 - (1 - r_squared) * (n - 1) / (n - p - 1)

    return(list(
      MSE = mse,
      RMSE = rmse,
      MAE = mae,
      R2 = r_squared,
      R2_adj = r_squared_adj
    ))
  }
```

```
  metrics <- calculate_regression_metrics(y_actual, y_pred)

  cat(" 回归评估指标:\n")
  cat("MSE (均方误差):", round(metrics$MSE, 4), "\n")
  cat("RMSE (均方根误差):", round(metrics$RMSE, 4), "\n")
  cat("MAE (平均绝对误差):", round(metrics$MAE, 4), "\n")
  cat("R² (决定系数):", round(metrics$R2, 4), "\n")
  cat(" 调整 R²:", round(metrics$R2_adj, 4), "\n")

  # 可视化预测效果
  plot_data <- data.frame(
    Actual = y_actual,
    Predicted = y_pred,
    Perfect = y_actual  # 完美预测线
  )

  library(ggplot2)
  p <- ggplot(plot_data, aes(x = Actual, y = Predicted)) +
    geom_point(alpha = 0.6) +
    geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed"
    geom_smooth(method = "lm", se = FALSE, color = "blue") +
    labs(title = " 预测值 vs 实际值",
         subtitle = paste("R² =", round(metrics$R2, 4)),
         x = " 实际值", y = " 预测值") +
    theme_minimal()

  print(p)

  return(metrics)
}

# 运行回归指标演示
regression_metrics <- demonstrate_regression_metrics()
```

```
回归评估指标:
MSE (均方误差): 0.3116
RMSE (均方根误差): 0.5582
MAE (平均绝对误差): 0.4523
```

```
R² (决定系数): 0.904
调整R²: 0.9031
```



1.4.2 分类问题评估指标

混淆矩阵的扩展：对于多分类问题，混淆矩阵是 $K \times K$ 的表格。

准确率的局限性：- 在不平衡数据中可能误导 - 例：99% 负例，全预测负类仍有 99% 准确率

精确率与召回率的权衡：- 精确率：预测为正的样本中实际为正的比例 - 召回率：实际为正的样本中被预测为正的比例 - 通常存在 trade-off 关系

Fβ 分数：

$$F_\beta = (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}}$$

- $\beta > 1$：更重视召回率 - $\beta < 1$：更重视精确率 - $\beta = 1$：平衡（F1 分数）

马修斯相关系数：

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

适用于不平衡数据的综合指标。

- 编程演示：分类评估指标 *

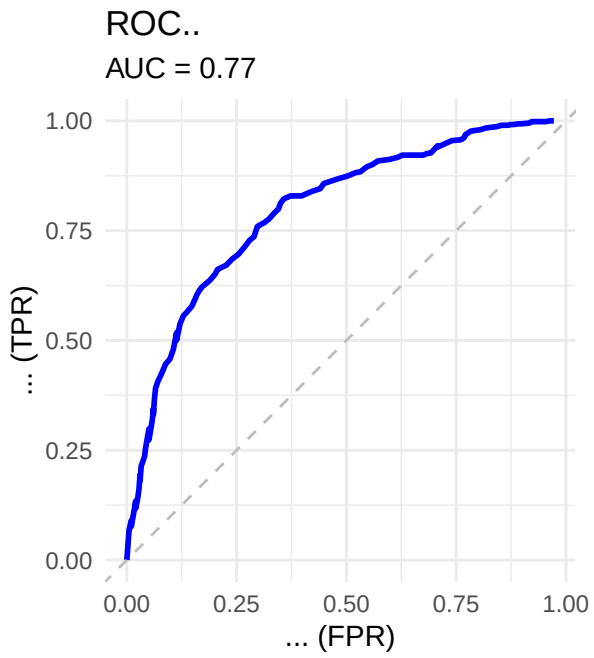
```r
# 1.4.2 分类评估指标演示
demonstrate_classification_metrics <- function() {
  set.seed(123)
  n <- 1000
  # 生成二分类数据
  feature1 <- rnorm(n)
  feature2 <- rnorm(n)

  # 生成真实概率
  true_prob <- plogis(0.5 + 1.2 * feature1 - 0.8 * feature2)
  true_labels <- rbinom(n, 1, true_prob)

  # 生成预测概率（带有一些误差）
  pred_prob <- true_prob + rnorm(n, 0, 0.1)
  pred_prob <- pmin(pmax(pred_prob, 0), 1)  # 限制在 [0,1] 范围内
  pred_labels <- ifelse(pred_prob > 0.5, 1, 0)

  # 计算混淆矩阵
  calculate_confusion_matrix <- function(actual, predicted) {
    tp <- sum(actual == 1 & predicted == 1)
    tn <- sum(actual == 0 & predicted == 0)
    fp <- sum(actual == 0 & predicted == 1)
    fn <- sum(actual == 1 & predicted == 0)

    return(list(TP = tp, TN = tn, FP = fp, FN = fn))
  }

  cm <- calculate_confusion_matrix(true_labels, pred_labels)

  # 计算各种指标
  accuracy <- (cm$TP + cm$TN) / n
  precision <- cm$TP / (cm$TP + cm$FP)
  recall <- cm$TP / (cm$TP + cm$FN)
  f1_score <- 2 * (precision * recall) / (precision + recall)

  # 计算 AUC
  calculate_auc <- function(actual, prob) {
    # 简单实现 AUC 计算
```

```
  positive_probs <- prob[actual == 1]
  negative_probs <- prob[actual == 0]

  comparisons <- 0
  correct <- 0

  # 抽样计算以加快速度
  n_sample <- min(100, length(positive_probs), length(negative_probs))
  pos_sample <- sample(positive_probs, n_sample)
  neg_sample <- sample(negative_probs, n_sample)

  for (p in pos_sample) {
    for (n in neg_sample) {
      comparisons <- comparisons + 1
      if (p > n) correct <- correct + 1
      else if (p == n) correct <- correct + 0.5
    }
  }

  return(correct / comparisons)
}

auc_score <- calculate_auc(true_labels, pred_prob)

cat(" 分类评估指标:\n")
cat(" 准确率:", round(accuracy, 4), "\n")
cat(" 精确率:", round(precision, 4), "\n")
cat(" 召回率:", round(recall, 4), "\n")
cat("F1 分数:", round(f1_score, 4), "\n")
cat("AUC:", round(auc_score, 4), "\n")

# 打印混淆矩阵
cat("\n混淆矩阵:\n")
confusion_df <- data.frame(
  Actual_0 = c(cm$TN, cm$FN),
  Actual_1 = c(cm$FP, cm$TP)
)
rownames(confusion_df) <- c("Predicted_0", "Predicted_1")
```

```r
print(confusion_df)

# 可视化 ROC 曲线
plot_roc_curve <- function(actual, prob) {
  thresholds <- seq(0, 1, 0.01)
  tpr <- numeric(length(thresholds))
  fpr <- numeric(length(thresholds))

  for (i in 1:length(thresholds)) {
    pred <- ifelse(prob > thresholds[i], 1, 0)
    cm_temp <- calculate_confusion_matrix(actual, pred)
    tpr[i] <- cm_temp$TP / (cm_temp$TP + cm_temp$FN)  # 真正率
    fpr[i] <- cm_temp$FP / (cm_temp$FP + cm_temp$TN)  # 假正率
  }

  roc_data <- data.frame(FPR = fpr, TPR = tpr)

  ggplot(roc_data, aes(x = FPR, y = TPR)) +
    geom_line(color = "blue", size = 1) +
    geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "gr
    labs(title = "ROC 曲线",
         subtitle = paste("AUC =", round(auc_score, 4)),
         x = " 假正率 (FPR)", y = " 真正率 (TPR)") +
    theme_minimal() +
    coord_equal()
}

roc_plot <- plot_roc_curve(true_labels, pred_prob)
print(roc_plot)

return(list(
  accuracy = accuracy,
  precision = precision,
  recall = recall,
  f1_score = f1_score,
  auc = auc_score,
  confusion_matrix = cm
))
```

```
}

# 运行分类指标演示
classification_metrics <- demonstrate_classification_metrics()
```

```
分类评估指标:
准确率: 0.748
精确率: 0.779
召回率: 0.8126
F1分数: 0.7955
AUC: 0.77

混淆矩阵:
             Actual_0 Actual_1
Predicted_0       258      139
Predicted_1       113      490
```



1.4.3 ROC 曲线与 AUC 曲线

ROC 曲线的绘制：1. 计算不同分类阈值下的 TPR 和 FPR 2. 以 FPR 为横轴，TPR 为纵轴绘制曲线 3. 曲线越靠近左上角，性能越好

AUC 的概率解释：AUC 等于随机选取的正样本得分高于随机选取的负样本得分的概率。

AUC 的优势：- 与分类阈值无关 - 反映模型整体排序能力 - 适用于不平衡数据

AUC 的局限性：- 只关注排序，不关注具体概率值 - 在某些情况下可能过于乐观

精确率-召回率曲线：在不平衡数据中通常比 ROC 曲线更有信息量。

1.5 模型选择原则

1.5.1 奥卡姆剃刀原理的统计基础

简约性原则：在同样能够解释数据的模型中，选择最简单的模型。

数学形式化：选择使以下目标函数最小的模型：

$$\text{Objective} = \text{拟合优度} + \lambda \times \text{模型复杂度}$$

模型复杂度的度量：- 参数个数 - VC 维 - 有效参数个数

1.5.2 信息准则的详细推导

AIC 的推导：基于 Kullback-Leibler 散度，目标是选择最接近真实分布的模型。

$$AIC = 2k - 2\ln(\hat{L})$$

其中：- k：模型参数个数 - $\hat{L}$：最大似然值

BIC 的推导：基于贝叶斯因子，在样本量较大时的一致性选择。

$$BIC = \ln(n)k - 2\ln(\hat{L})$$

AICc：小样本修正的 AIC：

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

信息准则的比较：

准则	目标	一致性	效率
AIC	预测精度	否	是
BIC	选择真实模型	是	否
AICc	小样本预测	否	是

1.5.3 实际选择策略

多准则综合评估：

- 计算多个信息准则
- 分析学习曲线
- 考虑业务约束
- 评估计算成本

领域知识的作用：

- 理解数据的生成机制
- 考虑特征的物理意义
- 评估模型的可解释性

1.6 软件使用

本书主要使用开源软件 R、Python。

16.1 R 语言

R+Rstudio，使用到的包 (package) 包括 knir、tidyverse、ggplot2 等，文件格式为.rmd，可以混编显示文本、代码块（chunk）、代码运行结果等各种格式内容，进行交互式的分析，可以直接生成 html、doc、PDF、PPT 等格式，撰写实验报告、论文、书籍等。

典型 **R** 包结构

```
a_r_package/
├── DESCRIPTION                    # □ 包的"身份证"
│   - Package: my_r_package        # 包名
│   - Version: 0.1.0               # 版本号
│   - Title: 包的功能描述          # 标题
│   - Author: 作者信息             # 作者
│   - Description: 详细描述        # 详细描述
│   - License: MIT                 # 许可证
│   - Imports: dplyr, ggplot2      # 依赖包
│   - Suggests: testthat           # 建议依赖
```

```
│
├── NAMESPACE                          # □ 函数的"门卫"
│   - export(hello_world)              # 导出函数供用户使用
│   - import(dplyr)                    # 导入依赖包的函数
│   - importFrom(ggplot2, aes)         # 导入特定函数
│
├── R/                                 # □ 核心代码目录
│   ├── hello.R                        # 函数定义文件1
│   │   # 函数定义
│   │   hello_world <- function() {
│   │     print("Hello from R package!")
│   │   }
│   │
│   ├── data_processing.R              # 函数定义文件2
│   │   clean_data <- function(df) {
│   │     dplyr::filter(df, !is.na(value))
│   │   }
│   │
│   └── utils.R                        # 工具函数（内部使用）
│       internal_func <- function() {
│         # 不导出，仅内部使用
│       }
│
├── man/                               # □ 文档目录（.Rd文件）
│   ├── hello_world.Rd                 # 函数的帮助文档
│   │   \name{hello_world}
│   │   \title{打招呼函数}
│   │   \description{详细描述}
│   │   \usage{hello_world()}
│   │   \examples{hello_world()}
│   │
│   └── my_r_package-package.Rd        # 包的总体文档
│
├── tests/                             # □ 测试目录
│   ├── testthat/                      # testthat测试框架
│   │   ├── test-hello.R               # 测试文件
│   │   │   test_that("hello works", {
│   │   │     expect_output(hello_world(), "Hello")
```

```
│   │   │   })
│   │   └── test-data.R
│   └── testthat.R                  # 测试运行器
│
├── vignettes/                      # □ 长篇教程/案例
│   └── introduction.Rmd            # 包的使用教程
│
├── data/                           # □ 包内置数据集
│   ├── sample_data.rda             # R数据文件
│   └── internal_data.rda           # 内部数据
│
├── inst/                           # □ 安装时包含的文件
│   ├── CITATION                    # 引用信息
│   ├── extdata/                    # 外部数据示例
│   └── templates/                  # 模板文件
│
└── .Rbuildignore                   # □ 构建时忽略的文件
```

R 数据分析典型项目结构

```
data_analysis_project/          # □ 数据分析项目
├── data/                       # □ 数据管理
│   ├── raw/                    # 原始数据（只读）
│   ├── processed/              # 处理后的数据
│   └── external/               # 外部数据源
│
├── R/                          # □ R代码
│   ├── 01_data_cleaning.R      # 数据清洗
│   ├── 02_exploratory.R        # 探索性分析
│   ├── 03_modeling.R           # 建模
│   ├── 04_visualization.R      # 可视化
│   └── functions/              # 自定义函数
│       ├── utils.R
│       └── plotting_functions.R
│
├── analysis/                   # □ 分析文档
│   ├── report.Rmd              # R Markdown报告
│   ├── presentation.Rmd        # 演示文稿
│   └── dashboard/              # Shiny应用
```

```
│
├── outputs/                        # □ 输出结果
│   ├── figures/                    # 生成的图表
│   ├── tables/                     # 生成的表格
│   └── models/                     # 保存的模型
│
├── tests/                          # □ 测试
├── references/                     # □ 参考文献/文档
├── .Rprofile                       # □ R启动配置
├── .Renviron                       # □ 环境变量
└── data_analysis_project.Rproj # □ RStudio项目文件
```

16.2 Python

依托 Anaconda 平台，该平台打包了 python 和重要的模块，包括环境的设置，自带 IDE 包括 jupyter notebook、jupyterlab、pycharm 等，使用到的库（library）包括 pandas、numpy、sklearn，由 jupyter notebook 编译文件格式为.ipynb，可以实现文本、代码块（cell）、代码运行结果、结果分析等混编，进行交互式的分析，可以直接生成 html、doc、PDF 等格式。

Python 包结构

```
a_python_package/
├── pyproject.toml                  # □ 现代构建配置（替代setup.py）
│   [build-system]
│   requires = ["setuptools", "wheel"]
│
│   [project]
│   name = "my_python_package"
│   version = "0.1.0"
│   dependencies = [
│     "numpy>=1.20",
│     "pandas>=1.3"
│   ]
│
├── src/                                 # □ 源代码目录（推荐结构）
│   └── my_python_package/             # 包的主目录
│       ├── __init__.py                # □ 包的入口点
│       │   """包的主模块"""
│       │   __version__ = "0.1.0"
```

```
│       │   __author__ = "Your Name"
│       │
│       │   # 导入关键函数，方便用户访问
│       │   from .core import (
│       │       normalize_data,
│       │       process_dataframe
│       │   )
│       │
│       │   # 也可以使用 __all__ 控制导入
│       │   __all__ = ["normalize_data", "process_dataframe"]
│       │
│       ├── core.py                    # □ 核心模块
│       │   """核心功能实现"""
│       │   import numpy as np
│       │   import pandas as pd
│       │
│       │   def normalize_data(x):
│       │       """标准化数据"""
│       │       if not isinstance(x, (np.ndarray, list)):
│       │           raise TypeError("输入必须是数组或列表")
│       │       x = np.array(x)
│       │       return (x - np.mean(x)) / np.std(x)
│       │
│       │   def _internal_helper():
│       │       """内部函数（以下划线开头）"""
│       │       return "internal"
│       │
│       ├── utils/                     # □ 子包/工具模块目录
│       │   ├── __init__.py
│       │   ├── file_utils.py
│       │   └── math_utils.py
│       │
│       ├── data/                      # □ 数据管理
│       │   ├── __init__.py
│       │   ├── datasets.py           # 数据集加载函数
│       │   └── constants.py          # 常量定义
│       │
│       └── tests/                     # □ 测试目录（可选放这里）
```

```
│           ├── __init__.py
│           ├── test_core.py
│           └── test_utils/
│
├── tests/                              # □ 测试目录（或放外面）
│   ├── test_core.py
│   │   def test_normalize():
│   │       from src.my_python_package.core import normalize_data
│   │       result = normalize_data([1, 2, 3])
│   │       assert np.allclose(result.mean(), 0)
│   │
│   └── conftest.py                     # pytest配置
│
├── docs/                               # □ 文档
│   ├── conf.py                         # Sphinx配置
│   ├── index.rst                       # 文档首页
│   └── api.rst                         # API文档
│
├── examples/                           # □ 使用示例
│   ├── basic_usage.ipynb               # Jupyter示例
│   └── advanced_demo.py
│
├── .github/                            # □ GitHub配置
│   ├── workflows/
│   │   └── ci.yml                      # 持续集成
│   └── ISSUE_TEMPLATE/                 # Issue模板
│
├── README.md                           # □ 项目说明
├── LICENSE                             # □ 许可证
├── requirements.txt                    # □ 依赖列表（可选）
└── setup.cfg                           # □ 传统配置（兼容性）
```

Python 数据分析项目结构

```
my_project/                         # □ 机器学习项目
├── src/                            # □ 源代码
│   ├── data/                       # 数据模块
│   │   ├── __init__.py
│   │   ├── make_dataset.py         # 数据准备
```

```
│   │   └── preprocessing.py    # 预处理
│   │
│   ├── features/               # 特征工程
│   │   ├── __init__.py
│   │   ├── build_features.py
│   │   └── selection.py
│   │
│   ├── models/                 # 模型
│   │   ├── __init__.py
│   │   ├── train_model.py
│   │   └── predict_model.py
│   │
│   └── visualization/          # 可视化
│       ├── __init__.py
│       └── plot_results.py
│
├── notebooks/                  # □ Jupyter笔记本
│   ├── 01_exploratory.ipynb    # 探索性分析
│   ├── 02_feature_engineering.ipynb
│   └── 03_model_training.ipynb
│
├── data/                       # □ 数据
│   ├── raw/                    # 原始数据
│   ├── interim/                # 中间数据
│   └── processed/              # 处理后的数据
│
├── models/                     # □ 模型存储
│   ├── trained_models/
│   └── model_metrics.json
│
├── reports/                    # □ 报告
│   ├── figures/                # 图表
│   └── final_report.md
│
├── tests/                      # □ 测试
├── configs/                    # □ 配置文件
│   ├── data_config.yaml
│   └── model_config.yaml
```

```
│
├── requirements.txt              # □ 依赖
├── setup.py                      # □ 安装配置
├── pyproject.toml                # □ 现代配置
├── Dockerfile                    # □ 容器化
└── README.md                     # □ 说明文档
```

1.6.3 学习工具平台搭建

统计学专业的同学，可以考虑 R+Rtools+Rstudio，使用 Rstudio 作为分析平台，文件格式为 RMD，如果想混合 Python 使用，可以加载包 reticulate，连接调用 python 使用。

面向机器学习的同学，应该以 Python 为主，学习、笔记、报告、作业等考虑 Anaconda 里面的 IDE 工具 jupyter notebook，文件格式为 ipynb，或者考虑使用 jupyterlab。

面向工程开发、项目管理的同学，考虑使用 VScode，如果工作偏向于数据科学，可以使用 Positren，是一个类似 vscode 的数据科学定制版，可以交叉调用 R、Python、Julia 等语言进行混合编程，集成 git 进行版本控制和发布，本书即使用 Positren 编写。

1.7 综合案例

1.7.1 mtcars 数据集分析：R 语言实现

```r
analyze_mtcars_dataset <- function() {
  # 加载数据
  data(mtcars)
  cat("mtcars 数据集基本信息:\n")
  cat(" 样本数:", nrow(mtcars), "\n")
  cat(" 变量数:", ncol(mtcars), "\n")
  cat("\n变量名称:\n")
  print(names(mtcars))

  # 数据概览
  cat("\n数据概览:\n")
  print(summary(mtcars))

  # 探索性数据分析
```

```r
library(ggplot2)
library(patchwork)

# 目标变量: mpg（每加仑行驶英里数）
p1 <- ggplot(mtcars, aes(x = mpg)) +
  geom_histogram(bins = 15, fill = "steelblue", alpha = 0.5) +
  labs(title = "MPG 分布", x = "MPG", y = " 频数") +
  theme_minimal()

p2 <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  labs(title = " 车重与 MPG 关系", x = " 重量", y = "MPG") +
  theme_minimal()

p3 <- ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  geom_boxplot(fill = "lightgreen", alpha = 0.7) +
  labs(title = " 气缸数与 MPG 关系", x = " 气缸数", y = "MPG") +
  theme_minimal()

# 组合图形
combined_plot <- (p1 | p2 | p3)
print(combined_plot)

# 数据划分
set.seed(123)
train_indices <- sample(1:nrow(mtcars), 0.7 * nrow(mtcars))
test_indices <- setdiff(1:nrow(mtcars), train_indices)

cat("\n数据划分结果:\n")
cat(" 训练集大小:", length(train_indices), "\n")
cat(" 测试集大小:", length(test_indices), "\n")

# 训练不同复杂度的模型
models <- list()

# 简单模型
models[["simple"]] <- lm(mpg ~ wt, data = mtcars[train_indices, ])
```

```r
# 中等复杂度模型
models[["medium"]] <- lm(mpg ~ wt + cyl + hp, data = mtcars[train_indices,

# 复杂模型
models[["complex"]] <- lm(mpg ~ wt + cyl + hp + disp + drat + qsec,
                          data = mtcars[train_indices, ])

# 评估模型性能
evaluate_model <- function(model, test_data) {
  predictions <- predict(model, newdata = test_data)
  actual <- test_data$mpg

  mse <- mean((actual - predictions)^2)
  rmse <- sqrt(mse)
  mae <- mean(abs(actual - predictions))
  r2 <- 1 - sum((actual - predictions)^2) / sum((actual - mean(actual))^2)

  return(list(MSE = mse, RMSE = rmse, MAE = mae, R2 = r2))
}

# 比较所有模型
test_data <- mtcars[test_indices, ]
performance <- data.frame()

for (model_name in names(models)) {
  model <- models[[model_name]]
  perf <- evaluate_model(model, test_data)

  performance <- rbind(performance, data.frame(
    Model = model_name,
    Parameters = length(coef(model)),
    MSE = perf$MSE,
    RMSE = perf$RMSE,
    MAE = perf$MAE,
    R2 = perf$R2
  ))
}
```

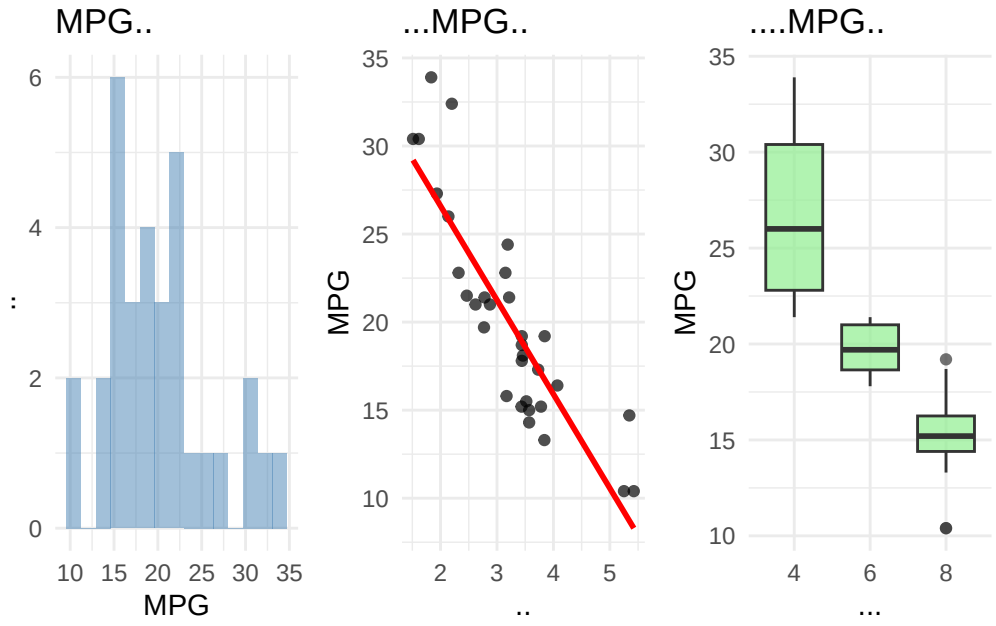
```r
cat("\n模型性能比较:\n")
print(performance)

# 交叉验证比较
library(mlr3)
library(mlr3verse)

# 创建任务
mtcars_task <- as_task_regr(mtcars[, c("mpg", "wt", "cyl", "hp", "disp", "dr
                            target = "mpg", id = "mtcars")

# 定义学习器
learners <- list(
  lrn("regr.lm", id = "simple_lm"),
  lrn("regr.rpart", id = "tree"),
  lrn("regr.kknn", id = "knn")
)

# 交叉验证
resampling <- rsmp("cv", folds = 5)

cv_results <- data.frame()
for (learner in learners) {
  rr <- resample(mtcars_task, learner, resampling)
  cv_results <- rbind(cv_results, data.frame(
    Model = learner$id,
    CV_RMSE = sqrt(rr$aggregate(msr("regr.mse"))),
    CV_MAE = rr$aggregate(msr("regr.mae")),
    CV_R2 = rr$aggregate(msr("regr.rsq"))
  ))
}

cat("\n交叉验证结果:\n")
print(cv_results)

# 过拟合分析
analyze_overfitting <- function() {
```

```
train_errors <- numeric(3)
test_errors <- numeric(3)
model_names <- c("simple", "medium", "complex")

for (i in 1:3) {
  model <- models[[model_names[i]]]

  # 训练误差
  train_pred <- predict(model, newdata = mtcars[train_indices, ])
  train_errors[i] <- mean((mtcars$mpg[train_indices] - train_pred)^2)

  # 测试误差
  test_pred <- predict(model, newdata = test_data)
  test_errors[i] <- mean((test_data$mpg - test_pred)^2)
}

overfitting_data <- data.frame(
  Model = model_names,
  Train_MSE = train_errors,
  Test_MSE = test_errors,
  Overfitting_Gap = test_errors - train_errors
)

cat("\n过拟合分析:\n")
print(overfitting_data)

# 可视化过拟合现象
p <- ggplot(overfitting_data, aes(x = Model)) +
  geom_line(aes(y = Train_MSE, color = " 训练误差", group = 1), size = 1)
  geom_line(aes(y = Test_MSE, color = " 测试误差", group = 1), size = 1) +
  geom_point(aes(y = Train_MSE, color = " 训练误差"), size = 3) +
  geom_point(aes(y = Test_MSE, color = " 测试误差"), size = 3) +
  labs(title = " 过拟合分析: 训练误差 vs 测试误差",
       x = " 模型复杂度", y = " 均方误差 (MSE)",
       color = " 误差类型") +
  theme_minimal() +
  scale_color_manual(values = c(" 训练误差" = "blue", " 测试误差" = "red"))
```

```
    print(p)

    return(overfitting_data)
  }

  overfitting_analysis <- analyze_overfitting()

  return(list(
    performance = performance,
    cv_results = cv_results,
    overfitting_analysis = overfitting_analysis,
    models = models
  ))
}

# 运行综合案例分析
mtcars_analysis <- analyze_mtcars_dataset()
```

```
mtcars数据集基本信息:
样本数: 32
变量数: 11

变量名称:
 [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
[11] "carb"

数据概览:
      mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
      drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000
```

```
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
       am              gear            carb
 Min.   :0.0000   Min.   :3.000   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
 Median :0.0000   Median :4.000   Median :2.000
 Mean   :0.4062   Mean   :3.688   Mean   :2.812
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :1.0000   Max.   :5.000   Max.   :8.000
```



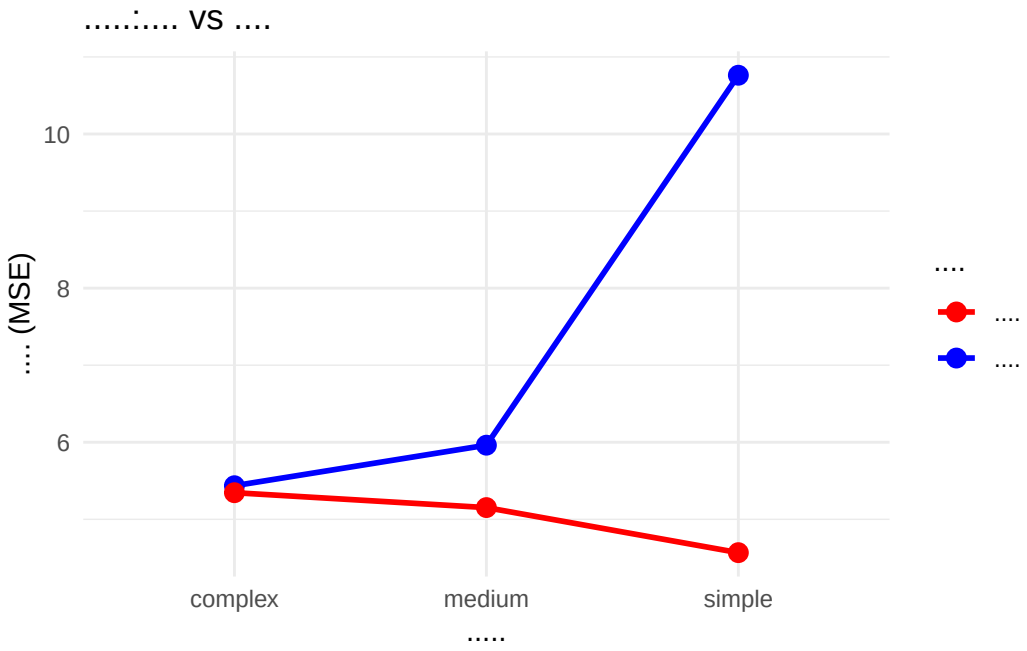

```
数据划分结果:
训练集大小: 22
测试集大小: 10
```

```
模型性能比较:
    Model Parameters      MSE     RMSE      MAE        R2
1  simple          2 4.567618 2.137199 1.845953 0.6379045
2  medium          4 5.152415 2.269893 1.877027 0.5915450
3 complex          7 5.345615 2.312059 1.765717 0.5762291
```

```
交叉验证结果:
```

```
              Model  CV_RMSE   CV_MAE      CV_R2
regr.mse  simple_lm 2.871556 2.279431  0.7249341
regr.mse1      tree 4.899379 3.895609 -0.2874332
regr.mse2       knn 2.723761 2.179108  0.6664141
```

```
过拟合分析:
    Model Train_MSE Test_MSE Overfitting_Gap
1  simple 10.762613 4.567618     -6.19499560
2  medium  5.962236 5.152415     -0.80982161
3 complex  5.436092 5.345615     -0.09047663
```



1.7.2 mtcars 数据集分析：Python 语言实现

```python
# mtcars 数据集分析 - Python 版本
def analyze_mtcars_python():
    import pandas as pd
    import numpy as np
    import matplotlib
    matplotlib.rc("font",family="Microsoft YaHei")
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_
from sklearn.preprocessing import StandardScaler

# 加载 mtcars 数据集
try:
    # 尝试从在线源加载
    url = "https://raw.githubusercontent.com/plotly/datasets/master/mtca
    mtcars = pd.read_csv(url)
except:
    # 如果网络不可用，使用内置数据集或创建模拟数据
    print(" 网络数据加载失败，使用模拟数据")
    np.random.seed(42)
    n = 32
    mtcars = pd.DataFrame({
        'mpg': np.random.normal(20, 6, n),
        'cyl': np.random.choice([4, 6, 8], n),
        'disp': np.random.normal(200, 100, n),
        'hp': np.random.normal(150, 50, n),
        'wt': np.random.normal(3, 1, n),
        'qsec': np.random.normal(18, 2, n)
    })

print("mtcars 数据集基本信息:")
print(f" 样本数: {mtcars.shape[0]}")
print(f" 变量数: {mtcars.shape[1]}")
print("\n数据概览:")
print(mtcars.describe())

# 数据可视化
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# MPG 分布
axes[0,0].hist(mtcars['mpg'], bins=10, alpha=0.7, color='steelblue')
axes[0,0].set_title('MPG 分布')
axes[0,0].set_xlabel('MPG')
```

```
    axes[0,0].set_ylabel('频数')

    # 车重与 MPG 关系
    axes[0,1].scatter(mtcars['wt'], mtcars['mpg'], alpha=0.7)
    z = np.polyfit(mtcars['wt'], mtcars['mpg'], 1)
    p = np.poly1d(z)
    axes[0,1].plot(mtcars['wt'], p(mtcars['wt']), "r--", alpha=0.8)
    axes[0,1].set_title('车重与 MPG 关系')
    axes[0,1].set_xlabel('重量')
    axes[0,1].set_ylabel('MPG')

    # 马力与 MPG 关系
    axes[1,0].scatter(mtcars['hp'], mtcars['mpg'], alpha=0.7)
    z = np.polyfit(mtcars['hp'], mtcars['mpg'], 1)
    p = np.poly1d(z)
    axes[1,0].plot(mtcars['hp'], p(mtcars['hp']), "r--", alpha=0.8)
    axes[1,0].set_title('马力与 MPG 关系')
    axes[1,0].set_xlabel('马力')
    axes[1,0].set_ylabel('MPG')

    # 气缸数与 MPG 关系
    mtcars.boxplot(column='mpg', by='cyl', ax=axes[1,1])
    axes[1,1].set_title('气缸数与 MPG 关系')
    axes[1,1].set_xlabel('气缸数')

    plt.tight_layout()
    plt.show()

    # 准备数据
    X = mtcars[['wt', 'cyl', 'hp', 'disp', 'qsec']]
    y = mtcars['mpg']

    # 数据划分
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.3, random_state=42
    )

    print(f"\n数据划分结果:")
```

```
print(f" 训练集大小: {X_train.shape[0]}")
print(f" 测试集大小: {X_test.shape[0]}")

# 训练不同模型
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'K-Neighbors': KNeighborsRegressor()
}

# 评估函数
def evaluate_model(model, X_train, X_test, y_train, y_test):
    model.fit(X_train, y_train)
    y_pred_train = model.predict(X_train)
    y_pred_test = model.predict(X_test)

    metrics = {
        'Train_MSE': mean_squared_error(y_train, y_pred_train),
        'Test_MSE': mean_squared_error(y_test, y_pred_test),
        'Train_R2': r2_score(y_train, y_pred_train),
        'Test_R2': r2_score(y_test, y_pred_test),
        'Overfitting_Gap': mean_squared_error(y_test, y_pred_test) - mea
    }
    return metrics

# 比较模型性能
results = []
for name, model in models.items():
    metrics = evaluate_model(model, X_train, X_test, y_train, y_test)
    results.append({
        'Model': name,
        **metrics
    })

results_df = pd.DataFrame(results)
print("\n模型性能比较:")
print(results_df)
```

```
    # 交叉验证
    print("\n交叉验证结果 (5 折):")
    cv_results = []
    for name, model in models.items():
        cv_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squar
        cv_rmse = np.sqrt(-cv_scores)
        cv_results.append({
            'Model': name,
            'CV_RMSE_mean': cv_rmse.mean(),
            'CV_RMSE_std': cv_rmse.std()
        })

    cv_df = pd.DataFrame(cv_results)
    print(cv_df)

    # 可视化结果
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5))

    # 过拟合分析
    x_pos = np.arange(len(models))
    width = 0.35

    train_mse = results_df['Train_MSE'].values
    test_mse = results_df['Test_MSE'].values

    ax1.bar(x_pos - width/2, train_mse, width, label='训练 MSE', alpha=0.7)
    ax1.bar(x_pos + width/2, test_mse, width, label='测试 MSE', alpha=0.7)
    ax1.set_xlabel('模型')
    ax1.set_ylabel('均方误差 (MSE)')
    ax1.set_title('过拟合分析')
    ax1.set_xticks(x_pos)
    ax1.set_xticklabels(results_df['Model'])
    ax1.legend()

    # R² 比较
    train_r2 = results_df['Train_R2'].values
    test_r2 = results_df['Test_R2'].values
```

```
    ax2.bar(x_pos - width/2, train_r2, width, label='训练 R²', alpha=0.7)
    ax2.bar(x_pos + width/2, test_r2, width, label='测试 R²', alpha=0.7)
    ax2.set_xlabel('模型')
    ax2.set_ylabel('R²')
    ax2.set_title('拟合优度比较')
    ax2.set_xticks(x_pos)
    ax2.set_xticklabels(results_df['Model'])
    ax2.legend()

    plt.tight_layout()
    plt.show()

    return {
        'performance': results_df,
        'cv_results': cv_df,
        'data': mtcars
    }

# 运行 Python 分析
mtcars_python_analysis = analyze_mtcars_python()
```

1.7.3 案例总结与最佳实践

关键发现总结

- 过拟合验证：在 mtcars 数据集中，复杂模型在训练集上表现更好，但在测试集上可能表现更差
- 交叉验证价值：相比单一划分，交叉验证提供更稳定的性能估计
- 模型选择：不同评估指标可能选择不同的最优模型
- 数据质量：真实数据集通常包含复杂的变量关系和数据问题

最佳实践建议

- 始终使用交叉验证：特别是对于小数据集
- 监控过拟合：比较训练和测试性能的差异
- 多指标评估：使用多个评估指标全面评价模型
- 理解数据：进行充分的探索性数据分析

本章总结

核心概念回顾

- 预测建模类型：回归分析预测连续值，分类分析预测离散类别
- 过拟合现象：模型在训练数据上表现过好，但在新数据上表现差
- 偏差-方差权衡：模型复杂度需要在偏差和方差之间取得平衡
- 数据划分：训练集用于参数估计，验证集用于模型选择，测试集用于最终评估
- 交叉验证：k 折交叉验证提供更稳定的误差估计
- 评估指标：回归问题用 MSE、RMSE、R^2，分类问题用准确率、精确率、召回率、AUC 等
- 模型选择：基于信息准则和业务需求选择合适模型

数据划分最佳实践：

- 在数据分析开始前划分测试集
- 使用随机种子确保结果可重现
- 对于时间序列数据，按时间顺序划分
- 对于不平衡数据，使用分层抽样

交叉验证实施要点：

1. 选择合适的 k 值（通常 5 或 10）

2. 使用分层抽样保持类别比例

3. 多次运行取平均以减少随机性

4. 记录每次交叉验证的结果和标准差

模型评估综合策略：

- 使用多个评估指标全面评估
- 考虑业务场景选择重点指标
- 分析错误案例理解模型局限
- 比较基准模型确认改进效果

重要数学公式总结

1. 偏差-方差分解：$E[(Y-\hat{f})^2] = \text{Bias}^2 + \text{Var} + \sigma_\varepsilon^2$
2. k 折交叉验证：$CV(k) = \frac{1}{k}\sum_{i=1}^{k} \text{Err}(D_{\text{test}}^{(i)})$
3. 回归评估：

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- $R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$

4. 分类评估：
   - Precision $= \frac{TP}{TP+FP}$
   - Recall $= \frac{TP}{TP+FN}$
   - $F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
5. 信息准则：
   - $AIC = 2k - 2\ln(L)$
   - $BIC = \ln(n)k - 2\ln(L)$

与后续章节的联系

本章建立的评估框架将贯穿整个教材：

- - 第 2-4 章：线性模型的评估与选择，重点关注 R^2、MSE 等指标
- - 第 5-7 章：复杂模型的过拟合控制，应用交叉验证和正则化
- - 第 8 章：集成学习的偏差-方差分析，比较不同集成策略
- - 第 9-10 章：支持向量机和神经网络的调优策略，使用验证集选择超参数

理解本章内容是学习后续所有预测模型的基础，良好的评估习惯是构建有效预测模型的关键。在实际应用中，应该根据具体问题选择合适的评估指标和方法，并始终关注模型的泛化能力。

2 线性回归

本章导读

线性回归和线性分析是一个强大而又经久不衰的工具，其思想的简洁性、数学的优雅性、解释的直观性以及广泛的可扩展性让人赞叹。理解线性回归，不仅是掌握一个工具，更是理解现代定量科学如何从数据中提取信息、建立模型的基本范式，它是通往更复杂数据世界的一扇不可或缺的大门。下面我们从历史的角度回溯线性回归思想的历程：

（1）起源：从 “平均主义” 到 “关系量化”（18-19 世纪）

线性回归的萌芽并非源于复杂的数学理论，而是来自天文学和测量学中对 “误差” 处理的迫切需求。

a) 核心问题：在测量同一天文现象（如行星轨道）时，不同观测者或同一观测者的多次测量会得到略有差异的结果。哪一个是最可信的 “真值”？

b) 先驱与关键思想：

- 高斯与勒让德：德国数学家高斯和法国数学家勒让德在 18 世纪末至 19 世纪初独立提出了最小二乘法。
- 核心思想：最有可能的 “真值” 或最佳拟合线，是能使所有观测值与预测值之差的平方和达到最小的那条线。误差平方（而非绝对差）在数学上更易处理，且能惩罚大的误差。
- 首次应用：勒让德在 1805 年明确发表了最小二乘法，用于分析彗星轨道；高斯声称更早使用它预测了谷神星的轨迹。

此时，线性回归更多地被视为一种精妙的 “数值计算技术”，而非一个统计模型。

（2）发展：奠定统计学基石（19 世纪末 - 20 世纪中叶）

随着生物学、经济学等社会科学对数据分析的需求增长，学者们开始探究这种 “关系” 背后的不确定性和统计意义。

a) 弗朗西斯 • 高尔顿：被誉为回归的 “概念之父”。他在 1886 年研究遗传学（身高遗传）时发现，虽然高个子父母的孩子也倾向于高，但其身高会 “回归” 到一个平均趋势。他引入了 “回归”（Regression）

一词，原意指 “倒退、退回平均值”。这一现象揭示了相关但非完美的关系，是现代相关与回归分析的起点。

b) 卡尔 • 皮尔逊：高尔顿的学生，将回归与相关分析系统化、数学化。他建立了皮尔逊相关系数，并发展了完整的双变量正态分布理论，为回归分析提供了坚实的概率分布基础。

c) 罗纳德 • 费希尔：20 世纪统计学巨擘，完成了线性回归的现代统计框架。他的关键贡献在于：方差分析、显著性检验、最大似然估计。他将回归从描述性工具升级为推断性工具——我们不仅能拟合一条线，还能检验斜率是否显著不为零（即变量间是否存在统计显著的关系），并进行预测和置信区间估计。

至此，线性回归从一个 “曲线拟合” 方法，演变为一套完整的、基于概率论的统计推断体系。

（3）成熟与扩展：应对复杂现实（20 世纪下半叶至今）

计算机的出现和科学研究的复杂化，推动了线性回归模型的极大丰富。

a) 计算革命：最小二乘法的求解涉及矩阵运算。计算机使快速处理海量数据和多个变量成为可能，催生了多元线性回归的广泛应用。

b) 理论扩展与变体：

- 广义线性模型：由内尔德和韦德伯恩提出，允许因变量不限于连续正态分布（如二项分布的 Logistic 回归、泊松分布的计数数据回归）。
- 正则化方法：为应对高维数据、多重共线性和过拟合问题，产生了岭回归、Lasso 回归等，它们在损失函数中加入惩罚项，提升模型稳健性和可解释性。
- 稳健回归：针对异常值敏感问题，发展出如 Huber 回归等方法。

c) 成为机器学习的基础：

- 在机器学习领域，线性回归被视为最简单的监督学习算法，是理解参数学习、梯度下降优化等概念的理想起点。
- 它也是许多复杂模型（如神经网络中的单个神经元）的构成模块。

（4）核心思想与局限

a) 核心思想：建模确定性关系：假设因变量（Y）可以由自变量（X）的线性组合加上一个随机误差来完美解释。目标：通过数据估计线性方程的系数，以量化 X 变化时 Y 的平均变化量，并用于预测和解释。

b) 主要局限：

- 线性假设强：无法直接捕捉变量间的非线性关系。

- 对假设敏感：要求误差独立、同方差、正态分布等，现实数据常不严格满足。
- 易受异常值影响。
- 相关非因果：这是最重要的一点。回归揭示的是关联性，而非因果性。忽视混淆变量会导致错误的因果解读。

线性回归的发展脉络清晰可见：起源于天文学的误差处理（最小二乘法），概念化于生物学的遗传研究（高尔顿的 “回归”），体系化于统计学的推断革命（皮尔逊、费希尔），扩展化于计算机时代的复杂数据分析（正则化、广义模型），基础化于人工智能/机器学习的模型基石。

线性回归是预测建模最基础、最重要的方法。本章将系统介绍线性回归模型的三种表述形式，详细推导参数估计的两种主要方法——最小二乘估计和极大似然估计，深入探讨模型拟合优度的评价指标，并建立完整的统计推断框架。最后，我们将讨论如何利用建立好的模型进行预测，为后续的模型诊断和扩展奠定坚实基础。

2.1 线性回归模型

变量之间的关系：函数关系，统计关系，没有关系

回忆数理统计的内容，相关系数：

$$\rho = \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}$$

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

取值范围：$-1 \leq \rho \leq 1$。

- $\rho = 1$：完全正相关（所有点落在一条斜向上的直线上）。
- $\rho = -1$：完全负相关（所有点落在一条斜向下的直线上）。
- $\rho = 0$：总体中不存在线性相关。但这不意味着没有其他关系（如曲线关系）。

我们几乎永远无法收集到整个总体的数据，因此 ρ 对我们来说通常是未知的。

样本相关系数由英国统计学家卡尔 • 皮尔逊提出，故常称 “皮尔逊积矩相关系数”。公式本质是样本协方差除以各自样本标准差的乘积取值范围：$-1 \leq r \leq 1$。它是一个统计量，如果你从同一个总体中抽取不同的样本，计算出的 r 值会不同。这种波动称为抽样变异。

2.1.1 总体回归模型

总体回归模型：

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

其中：

- β_0 是截距项
- $\beta_1, \beta_2, \cdots, \beta_p$ 是回归系数
- ε 是随机误差项，满足 $E(\varepsilon|X) = 0$

总体回归函数描述了因变量 Y 与自变量 $X_1, X_2, \cdots, X_p$ 之间的真实关系：

$$E(Y|X_1, \cdots, X_p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

2.1.2 样本回归模型

基于 n 个观测样本 $(x_{i1}, x_{i2}, \cdots, x_{ip}, y_i), i = 1, 2, \cdots, n$，样本回归模型为：

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip} + e_i$$

其中：

- $\hat{\beta}_0, \hat{\beta}_1, \cdots, \hat{\beta}_p$ 是总体参数的估计值
- $e_i = y_i - \hat{y}_i$ 是第 i 个观测的残差
- $\hat{y}_i$ 是第 i 个观测的拟合值

样本回归函数：

$$\hat{y_i} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}$$

2.1.3 矩阵形式的回归模型

令：

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

则线性回归模型的矩阵形式为：

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

其中：

- $\mathbf{y}$ 是 $n \times 1$ 的因变量向量
- $\mathbf{X}$ 是 $n \times (p+1)$ 的设计矩阵
- $\boldsymbol{\beta}$ 是 $(p+1) \times 1$ 的参数向量
- $\boldsymbol{\varepsilon}$ 是 $n \times 1$ 的误差项向量

2.1.4 基本假设

假设 1：线性关系

- 内容：因变量 Y 与自变量 X 之间存在线性关系，且参数 β 是线性的。误差项 ε 以相加的形式进入模型。
- 含义：X 变化一个单位，Y 平均变化 β 个单位，与 X 本身的取值无关。
- 违反后果：模型无法捕捉真实关系，导致预测偏差。例如，真实关系是二次的，用线性模型会系统性地高估或低估。
- 检验与处理：绘制 Y 与每个 X 的散点图；检查残差与拟合值的图。若违反，可对 X 或 Y 进行变量变换（如取对数、平方），或使用多项式回归。

假设 2：随机抽样

- 内容：样本数据是从总体中随机抽样得到的。
- 含义：样本能代表总体，避免选择偏差。
- 违反后果：样本估计无法推广到总体（外部效度低）。例如，只用大学生数据研究全民收入。

- 检验与处理：无法从数据本身直接诊断，需了解抽样过程。处理方法是改进抽样设计。

假设 3：严格外生性

- 内容：给定所有自变量 X 的值，误差项 ε 的条件期望为零。即 $E(\varepsilon|X) = 0$。
- 含义：模型中没有遗漏重要的解释变量，且所有包含的 X 都与误差项 ε 不相关。换句话说，X 是 “外生” 的。
- 违反后果（最严重之一）：导致 “内生性” 问题，使参数估计 β 有偏且不一致。无论样本多大，偏差都不会消失。
- 常见原因：遗漏变量、测量误差、互为因果（反向因果）。
- 检验与处理：难以直接检验。通常基于理论推理。处理需用工具变量法、面板数据模型、断点回归等高级方法。

假设 4：无完全多重共线性

- 内容：自变量之间不存在完全的线性关系。
- 含义：没有一个自变量可以精确地由其他自变量的线性组合表示。样本数据中，各 X 提供独立信息。* 违反后果：OLS 无法唯一求解参数估计值（矩阵不可逆）。
- 检验：查看相关系数矩阵；计算方差膨胀因子和条件数。
- 处理：删除高度相关的变量之一；使用主成分回归或岭回归等降维/正则化方法。

假设 5：球形误差（同方差与无自相关）这是两个子假设的合称：

5a) 同方差性：给定所有 X，误差项 ε 的条件方差为常数。即 $Var(\varepsilon|X) = \sigma^2$。

- 违反后果：OLS 估计量仍无偏，但不再是最有效的（方差不是最小）；标准误估计有偏，导致假设检验失效。
- 检验：绘制残差与拟合值的散点图，看是否呈漏斗形、扇形等。正式检验有 Breusch-Pagan 检验、White 检验等。
- 处理：使用稳健标准误（最常见且简单）、加权最小二乘法、对变量进行变换。

5b) 无自相关：对于不同观测 i 和 j，其误差项互不相关。即 $Cov(\varepsilon_i, \varepsilon_j|X) = 0$ (i ≠ j)。

- 违反后果：与异方差类似，效率降低，标准误估计有偏（通常偏低），导致 t 统计量虚高，易得出显著结论。
- 常见于：时间序列数据（本期误差受上期影响）、空间数据。
- 检验：绘制残差与时间/顺序的图；使用 Durbin-Watson 检验（针对一阶自相关）。

- 处理：时间序列模型（如 ARIMA）、使用异方差自相关稳健标准误、广义最小二乘法。

假设 6：误差项的正态性（可选但重要）* 内容：误差项 ε 服从正态分布。即 $\varepsilon|X\ N(0,\sigma^2 I)$。

- 含义：在样本量不大时，此假设是进行精确统计推断（如 t 检验、F 检验）的基础。
- 违反后果：在小样本下，β 的分布不是精确的 t 分布，检验和置信区间可能不准确。但在大样本下（中心极限定理），OLS 估计量近似正态，此假设可放松。
- 检验：绘制残差的正态 Q-Q 图；使用 Shapiro-Wilk、Kolmogorov-Smirnov 等检验。
- 处理：增大样本量；使用自助法进行推断；考虑对 Y 进行变换。

这六个假设可以按目标归类：

目标	核心假设	关键点
估计无偏	1. 线性关系 3. 严格外生性	模型设定正确，无遗漏变量和反向因果。
估计可算	4. 无完全多重共线性	自变量提供独立信息，矩阵可逆。
估计有效（BLUE）	5. 球形误差（同方差 + 无自相关）	OLS 估计量的方差最小。
推断可靠	6. 误差正态性（小样本）+ 以上所有	可用于假设检验和置信区间。

2.2 参数估计方法

2.2.1 普通最小二乘估计

基本原理：寻找参数估计值 $\hat{\boldsymbol{\beta}}$，使得残差平方和最小。

残差平方和：

$$SSE = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

最小二乘估计量：通过对 SSE 关于 $\boldsymbol{\beta}$ 求导并令导数为零，可得正规方程组：

$$\mathbf{X}'\mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{y}$$

当 $\mathbf{X}'\mathbf{X}$ 可逆时，OLS 估计量为：

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

OLS 估计量的性质（在经典假设下）：

1. 无偏性：$E(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$

2. 方差-协方差矩阵：$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$

3. 高斯-马尔可夫定理：OLS 估计量是最佳线性无偏估计

2.2.2 极大似然估计

基本思想：在正态误差的假设下，选择使样本观测值出现概率最大的参数值。

似然函数：假设 $\varepsilon_i \sim N(0, \sigma^2)$，则 $y_i \sim N(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2)$，似然函数为：

$$L(\boldsymbol{\beta}, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2}{2\sigma^2}\right)$$

对数似然函数：

$$\ell(\boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2$$

极大似然估计量：- 回归系数：$\hat{\boldsymbol{\beta}}_{MLE} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$（与 OLS 相同）- 误差方差：$\hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}})^2$

注意：$\hat{\sigma}^2_{MLE}$ 是有偏估计，通常使用无偏估计：

$$s^2 = \frac{1}{n-p-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2.2.3 点估计与区间估计

系数估计：- 点估计：$\hat{\beta}_j$（OLS 或 MLE 估计值）- 标准误：$SE(\hat{\beta}_j) = s\sqrt{[(\mathbf{X}'\mathbf{X})^{-1}]_{jj}}$

系数的置信区间（置信水平 $1-\alpha$）：

$$\hat{\beta}_j \pm t_{1-\alpha/2}(n-p-1) \cdot SE(\hat{\beta}_j)$$

误差方差的置信区间：

$$\left[\frac{(n-p-1)s^2}{\chi^2_{1-\alpha/2}(n-p-1)}, \frac{(n-p-1)s^2}{\chi^2_{\alpha/2}(n-p-1)}\right]$$

最小二乘法实现

```
set.seed(123)
n <- 100
x <- seq(1, 10, length.out = n)
y <- 2 + 1.5 * x + rnorm(n, 0, 1.5)

# 手动实现 OLS
manual_ols <- function(x, y) {
  x_mean <- mean(x)
  y_mean <- mean(y)
  slope <- sum((x - x_mean) * (y - y_mean)) / sum((x - x_mean)^2)
  intercept <- y_mean - slope * x_mean
  return(c(intercept = intercept, slope = slope))
}

manual_coef <- manual_ols(x, y)
lm_coef <- coef(lm(y ~ x))

cat(" 参数估计比较:\n")
```

参数估计比较:

```
print(data.frame(
  方法 = c(" 手动 OLS", "lm 函数"),
  截距 = c(manual_coef[1], lm_coef[1]),
  斜率 = c(manual_coef[2], lm_coef[2])
))
```

```
                方法      截距      斜率
intercept    手动OLS 1.907728 1.541433
(Intercept)  lm函数 1.907728 1.541433
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.datasets import fetch_openml
```

```
# 设置样式
plt.style.use('default')
sns.set_palette("husl")

# 生成数据
np.random.seed(123)
n = 100
x = np.linspace(1, 10, n)
y = 2 + 1.5 * x + np.random.normal(0, 1.5, n)

def manual_ols(x, y):
    """ 手动实现 OLS"""
    x_mean, y_mean = np.mean(x), np.mean(y)
    slope = np.sum((x - x_mean) * (y - y_mean)) / np.sum((x - x_mean)**2)
    intercept = y_mean - slope * x_mean
    return intercept, slope

# 比较结果
manual_intercept, manual_slope = manual_ols(x, y)
lr = LinearRegression()
lr.fit(x.reshape(-1, 1), y)
```

```
LinearRegression()
```

```
print(" 参数估计比较:")
```

```
参数估计比较:
```

```
print(f"{'方法':<10} {'截距':<8} {'斜率':<8}")
```

```
方法          截距        斜率
```

```
print(f"{'手动 OLS':<10} {manual_intercept:.4f}  {manual_slope:.4f}")
```

```
手动OLS       2.0149  1.5047
```

```
print(f"{'Sklearn':<10} {lr.intercept_:.4f}  {lr.coef_[0]:.4f}")
```

```
Sklearn     2.0149  1.5047
```

2.3 拟合优度

2.3.1 总平方和分解

总平方和：$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2$ 回归平方和：$SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$ 残差平方和：$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

平方和分解公式：

$$SST = SSR + SSE$$

2.3.2 决定系数

定义：

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

性质：- $0 \leq R^2 \leq 1$ - R^2 衡量模型对因变量变异的解释比例 - R^2 随自变量增加而增加，可能过拟合

2.3.3 调整的决定系数

定义：

$$R^2_{adj} = 1 - \frac{SSE/(n-p-1)}{SST/(n-1)} = 1 - (1-R^2)\frac{n-1}{n-p-1}$$

优点：对模型复杂度施加惩罚，更适合模型比较

2.4 假设检验

2.4.1 模型整体显著性检验（F 检验）

原假设：$H_0 : \beta_1 = \beta_2 = \cdots = \beta_p = 0$

F 统计量：

$$F = \frac{SSR/p}{SSE/(n-p-1)} = \frac{MSR}{MSE} \sim F(p, n-p-1)$$

决策规则：如果 $F > F_{1-\alpha}(p, n-p-1)$，拒绝原假设，认为模型整体显著

2.4.2 单个系数显著性检验（t 检验）

原假设：$H_0 : \beta_j = 0$

t 统计量：

$$t = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \sim t(n-p-1)$$

决策规则：如果 $|t| > t_{1-\alpha/2}(n-p-1)$，拒绝原假设，认为该自变量对因变量有显著影响

交互演示：t 检验与 p 值

```
# 2.3.2 交互式假设检验演示
interactive_hypothesis_test <- function() {
  if (!require(manipulate)) {
    stop(" 请安装 manipulate 包: install.packages('manipulate')")
  }

  set.seed(123)

  manipulate({
    # 生成数据
    n <- sample_size
    x <- runif(n, 0, 10)
    true_slope <- 0  # 零假设为真
    y <- 1 + true_slope * x + rnorm(n, 0, noise_sd)

    data <- data.frame(x = x, y = y)
    model <- lm(y ~ x, data = data)
    summary_model <- summary(model)

    # 提取 t 统计量和 p 值
    t_stat <- summary_model$coefficients[2, "t value"]
    p_value <- summary_model$coefficients[2, "Pr(>|t|)"]

    # 绘制数据点和回归线
    par(mfrow = c(1, 2))

    # 左图：散点图和回归线
    plot(x, y, pch = 16, col = "blue",
```

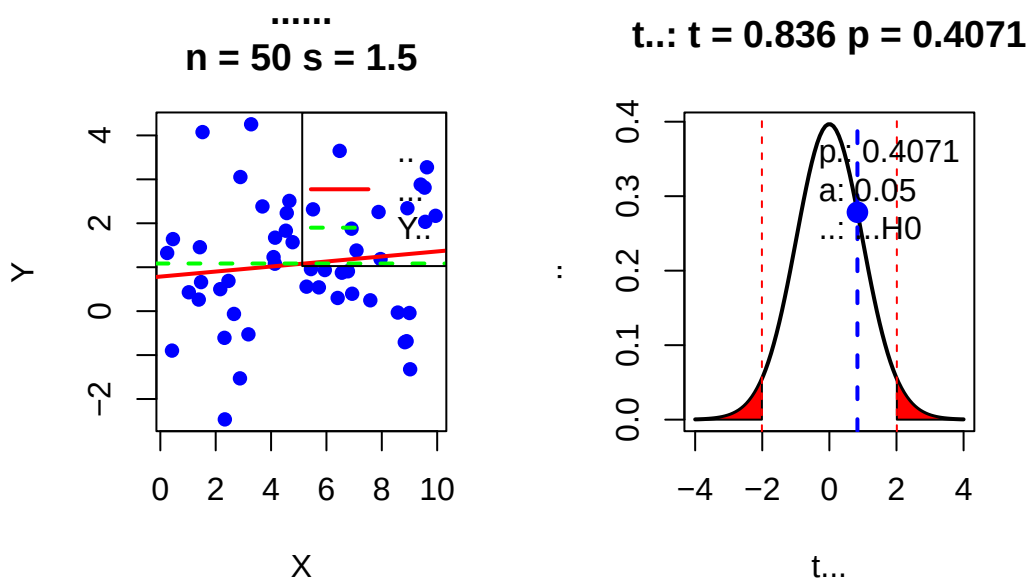
```
     main = paste(" 数据与回归线\nn =", n, "σ =", noise_sd),
     xlab = "X", ylab = "Y",
     xlim = c(0, 10), ylim = c(1 - 3*noise_sd, 1 + 3*noise_sd))
abline(model, col = "red", lwd = 2)
abline(h = mean(y), col = "green", lwd = 2, lty = 2)
legend("topright",
       legend = c(" 数据", " 回归线", "Y 均值"),
       col = c("blue", "red", "green"),
       lwd = c(NA, 2, 2), pch = c(16, NA, NA),
       lty = c(NA, 1, 2))

# 右图: t 分布和 p 值
x_t <- seq(-4, 4, length.out = 100)
y_t <- dt(x_t, df = n - 2)

plot(x_t, y_t, type = "l", lwd = 2, col = "black",
     main = paste("t 检验: t =", round(t_stat, 3),
                  "p =", round(p_value, 4)),
     xlab = "t 统计量", ylab = " 密度")

# 着色拒绝域
alpha <- 0.05
critical_value <- qt(1 - alpha/2, df = n - 2)

# 左侧拒绝域
x_left <- seq(-4, -critical_value, length.out = 50)
y_left <- dt(x_left, df = n - 2)
polygon(c(-4, x_left, -critical_value), c(0, y_left, 0),
        col = "red", density = 20, angle = 45)

# 右侧拒绝域
x_right <- seq(critical_value, 4, length.out = 50)
y_right <- dt(x_right, df = n - 2)
polygon(c(critical_value, x_right, 4), c(0, y_right, 0),
        col = "red", density = 20, angle = 45)

# 标记 t 统计量
abline(v = t_stat, col = "blue", lwd = 2, lty = 2)
```

```
    points(t_stat, dt(t_stat, df = n - 2), pch = 16, col = "blue", cex = 1.5
    text(t_stat, dt(t_stat, df = n - 2) + 0.02,
         paste("t =", round(t_stat, 3)), pos = 3, col = "blue")

    # 标记临界值
    abline(v = c(-critical_value, critical_value),
           col = "red", lwd = 1, lty = 2)
    text(critical_value, 0.3, paste(" 临界值 =", round(critical_value, 3)),
         pos = 4, col = "red")

    # 添加决策
    decision <- ifelse(p_value < alpha, " 拒绝 H0", " 不拒绝 H0")
    legend("topright",
           legend = c(paste("p 值:", round(p_value, 4)),
                      paste("α:", alpha),
                      paste(" 决策:", decision)),
           bty = "n", cex = 0.8)

  },
  sample_size = slider(10, 200, initial = 30, step = 10, label = " 样本量 n"),
  noise_sd = slider(0.5, 5, initial = 2, step = 0.5, label = " 噪声标准差 σ")
  )
}

# 运行交互演示（需要 RStudio）
# interactive_hypothesis_test()

# 静态版本
static_hypothesis_test <- function(n = 30, noise_sd = 2) {
  set.seed(123)

  # 生成数据
  x <- runif(n, 0, 10)
  true_slope <- 0  # 零假设为真
  y <- 1 + true_slope * x + rnorm(n, 0, noise_sd)

  data <- data.frame(x = x, y = y)
  model <- lm(y ~ x, data = data)
```

```r
summary_model <- summary(model)

# 提取统计量
t_stat <- summary_model$coefficients[2, "t value"]
p_value <- summary_model$coefficients[2, "Pr(>|t|)"]
conf_int <- confint(model)[2, ]

# 绘制图形
par(mfrow = c(1, 2))

# 左图：数据与回归线
plot(x, y, pch = 16, col = "blue",
     main = paste(" 数据与回归线\nn =", n, "σ =", noise_sd),
     xlab = "X", ylab = "Y")
abline(model, col = "red", lwd = 2)
abline(h = mean(y), col = "green", lwd = 2, lty = 2)
legend("topright",
       legend = c(" 数据", " 回归线", "Y 均值"),
       col = c("blue", "red", "green"),
       lwd = c(NA, 2, 2), pch = c(16, NA, NA),
       lty = c(NA, 1, 2))

# 右图：t 分布
x_t <- seq(-4, 4, length.out = 100)
y_t <- dt(x_t, df = n - 2)

plot(x_t, y_t, type = "l", lwd = 2, col = "black",
     main = paste("t 检验: t =", round(t_stat, 3),
                  "p =", round(p_value, 4)),
     xlab = "t 统计量", ylab = " 密度")

alpha <- 0.05
critical_value <- qt(1 - alpha/2, df = n - 2)

# 着色拒绝域
x_left <- seq(-4, -critical_value, length.out = 50)
y_left <- dt(x_left, df = n - 2)
polygon(c(-4, x_left, -critical_value), c(0, y_left, 0),
```

```
          col = "red", alpha = 0.3)

  x_right <- seq(critical_value, 4, length.out = 50)
  y_right <- dt(x_right, df = n - 2)
  polygon(c(critical_value, x_right, 4), c(0, y_right, 0),
          col = "red", alpha = 0.3)

  # 标记统计量
  abline(v = t_stat, col = "blue", lwd = 2, lty = 2)
  points(t_stat, dt(t_stat, df = n - 2), pch = 16, col = "blue", cex = 1.5)

  # 标记临界值
  abline(v = c(-critical_value, critical_value),
         col = "red", lwd = 1, lty = 2)

  # 决策
  decision <- ifelse(p_value < alpha, " 拒绝 H0", " 不拒绝 H0")
  legend("topright",
         legend = c(paste("p 值:", round(p_value, 4)),
                    paste("α:", alpha),
                    paste(" 决策:", decision)),
         bty = "n")

  par(mfrow = c(1, 1))

  cat(" 假设检验结果:\n")
  cat("t 统计量:", round(t_stat, 4), "\n")
  cat("p 值:", round(p_value, 4), "\n")
  cat("95% 置信区间: [", round(conf_int[1], 4), ",", round(conf_int[2], 4), "]
  cat(" 决策:", decision, "\n")

  return(list(
    model = model,
    t_stat = t_stat,
    p_value = p_value,
    decision = decision
  ))
}
```

```
# 运行静态检验演示
static_test <- static_hypothesis_test(n = 50, noise_sd = 1.5)
```




```
假设检验结果:
t统计量: 0.8363
p值: 0.4071
95%置信区间: [ -0.0804 , 0.1949 ]
决策: 不拒绝H0
```

2.4.3 系数线性组合的检验

原假设：

$H_0 : \mathbf{C}\boldsymbol{\beta} = \mathbf{d}$

F 统计量：

$$F = \frac{(\mathbf{C}\hat{\boldsymbol{\beta}} - \mathbf{d})'[\mathbf{C}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{C}']^{-1}(\mathbf{C}\hat{\boldsymbol{\beta}} - \mathbf{d})}{q \cdot MSE} \sim F(q, n-p-1)$$

其中 $\mathbf{C}$ 是 $q \times (p+1)$ 的约束矩阵，$\mathbf{d}$ 是 $q \times 1$ 的常数向量

模型显著性检验

```
model <- lm(y ~ x)
summary_model <- summary(model)

cat(" 模型显著性检验:\n")
```

模型显著性检验:

```
cat("F 统计量:", round(summary_model$fstatistic[1], 3), "\n")
```

F统计量: 869.473

```
cat("p 值:", format.pval(pf(summary_model$fstatistic[1],
                            summary_model$fstatistic[2],
                            summary_model$fstatistic[3],
                            lower.tail = FALSE)), "\n")
```

p值: < 2.22e-16

```
cat("\n系数检验:\n")
```

系数检验:

```
print(round(summary_model$coefficients, 4))
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    1.9077     0.3186  5.9885        0
x              1.5414     0.0523 29.4868        0
```

2.5 预测

2.5.1 点预测

对于新的自变量观测 $\mathbf{x}_0 = (1, x_{01}, x_{02}, \cdots, x_{0p})'$，因变量的点预测为：

$$\hat{y}_0 = \mathbf{x}_0' \hat{\boldsymbol{\beta}}$$

2.5.2 区间预测

均值的置信区间（估计条件均值的置信区间）：

$$\hat{y}_0 \pm t_{1-\alpha/2}(n-p-1) \cdot s\sqrt{\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0}$$

个别值的预测区间（预测单个新观测值的区间）：

$$\hat{y}_0 \pm t_{1-\alpha/2}(n-p-1) \cdot s\sqrt{1+\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0}$$

重要区别：- 均值置信区间：估计的是 $E(y|\mathbf{x}_0)$ 的置信区间 - 个别值预测区间：预测的是单个 y_0 的区间，包含额外的误差方差项

2.5.3 预测精度的影响因素

1. 样本量 n：样本量越大，预测越精确
2. 自变量变异：$\mathbf{X}'\mathbf{X}$ 的行列式值越大，预测越精确
3. 杠杆值：$\mathbf{x}_0$ 与样本中心的距离影响预测方差
4. 误差方差：σ^2 越小，预测越精确

2.6 案例分析

本章总结

核心公式回顾

1. OLS 估计：$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$
2. 系数方差：$\text{Var}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}$
3. 拟合优度：$R^2 = 1 - \frac{SSE}{SST}$
4. F 检验：$F = \frac{MSR}{MSE} \sim F(p, n-p-1)$
5. t 检验：$t = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \sim t(n-p-1)$
6. 预测区间：$\hat{y}_0 \pm t_{\alpha/2} \cdot s\sqrt{1+\mathbf{x}_0'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_0}$

关键概念

概念	定义	应用
OLS 估计	最小化残差平方和	参数估计的主要方法
MLE 估计	最大化似然函数	在正态假设下与 OLS 等价
R^2	模型解释的变异比例	拟合优度初步评价
调整 R^2	对复杂度惩罚的 R^2	模型比较
F 检验	模型整体显著性检验	判断模型是否有意义
t 检验	单个系数显著性检验	变量选择依据
预测区间	个别值的预测范围	实际预测应用

线性回归为后续所有预测建模方法提供了理论基础和分析框架。理解本章内容对于掌握现代预测建模技术至关重要。

3 模型诊断

本章导读

在建立线性回归模型后，我们绝不能将其视为终极真理。模型诊断如同一次严谨的" 体检"，旨在评估模型的基本假设是否成立，数据中是否存在异常情况。本章将系统学习如何使用残差分析这一核心工具，检验模型的线性性、正态性及同方差性假设，并深入探讨异常值和高杠杆点的识别方法。更重要的是，我们将建立完整的问题诊断与处理框架，为构建更可靠的预测模型奠定基础。

3.1 残差分析

3.1.1 残差的基本概念与理论基础

残差的定义：对于线性回归模型

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i$$

第 i 个观测值的残差定义为：

$$e_i = y_i - \hat{y}_i$$

其中 $\hat{y}_i$ 是模型的预测值。

经典线性回归的基本假设：

（1）. 线性关系：$E(\varepsilon_i) = 0$

（2）. 同方差性：$\text{Var}(\varepsilon_i) = \sigma^2$（常数）

（3）. 无自相关：$\text{Cov}(\varepsilon_i, \varepsilon_j) = 0$（当 $i \neq j$）

（4）. 正态性：$\varepsilon_i \sim N(0, \sigma^2)$ 5. 无多重共线性：自变量之间不存在完全线性关系

3.1.2 标准化残差与学生化残差

标准化残差：

$$z_i = \frac{e_i}{\hat{\sigma}} = \frac{e_i}{\sqrt{\frac{1}{n-p-1}\sum_{i=1}^{n} e_i^2}}$$

学生化残差（内部学生化）：

$$r_i = \frac{e_i}{\hat{\sigma}\sqrt{1-h_{ii}}}$$

外部学生化残差：

$$t_i = \frac{e_i}{\hat{\sigma}_{(i)}\sqrt{1-h_{ii}}} \sim t(n-p-2)$$

其中 $\hat{\sigma}_{(i)}$ 是剔除第 i 个观测值后重新估计的标准差：

$$\hat{\sigma}^2_{(i)} = \frac{(n-p-1)\hat{\sigma}^2 - e_i^2/(1-h_{ii})}{n-p-2}$$

3.1.3 残差图的系统分析

残差 vs 拟合值图

这是最重要的诊断图，用于检验：- 线性性：点应随机分布在水平带 around 0 - 同方差性：点的散布范围应恒定

异常模式识别：- 喇叭口形：Var$(e_i) \propto \hat{y}_i^k$，提示异方差 - 曲线模式：提示非线性关系，需考虑 $E(y|x) = f(x'\beta)$

正态 Q-Q 图

检验残差的正态性假设。理论分位数与样本分位数应大致在直线上：

理论分位数：$z_{(i)} = \Phi^{-1}\left(\frac{i-0.375}{n+0.25}\right)$

偏离模式：

- - S 形曲线：分布有偏
- - 尾部偏离：重尾或轻尾分布

残差 vs 自变量图

用于识别特定自变量的非线性关系：

$$e_i \text{ vs } x_{ij}$$

3.2 异常值与高杠杆点

3.2.1 杠杆值分析

杠杆值的定义：

$$h_{ii} = \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i$$

其中 $\mathbf{x}_i = (1, x_{i1}, x_{i2}, \cdots, x_{ip})'$

杠杆值的性质：

- $0 \leq h_{ii} \leq 1$ - $\sum_{i=1}^{n} h_{ii} = p + 1$

- 平均杠杆值：$\bar{h} = \frac{p+1}{n}$

高杠杆点的识别：

- - 经验法则：$h_{ii} > 2\bar{h} = \frac{2(p+1)}{n}$
- - 严格标准：$h_{ii} > 3\bar{h} = \frac{3(p+1)}{n}$

3.2.2 异常值检测

基于学生化残差的准则：

$$|t_i| > t_{1-\alpha/2}(n-p-2)$$

异常值的数学定义：观测值 $(y_i, \mathbf{x}_i)$ 被称为异常值，如果

$$|y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}| \gg \sigma$$

3.2.3 影响点的综合度量

Cook 距离

定义：

$$D_i = \frac{(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)})'(\mathbf{X}'\mathbf{X})(\hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\beta}}_{(i)})}{(p+1)\hat{\sigma}^2}$$

计算公式：

$$D_i = \frac{e_i^2}{(p+1)\hat{\sigma}^2} \times \frac{h_{ii}}{(1-h_{ii})^2}$$

判断准则：

- $D_i > 1$：绝对需要检查

- $D_i > \frac{4}{n}$：建议检查

DFFITS 统计量

定义：衡量第 i 个观测对自身拟合值的影响

$$\text{DFFITS}_i = \frac{\hat{y}_i - \hat{y}_{i(i)}}{\hat{\sigma}_{(i)}\sqrt{h_{ii}}}$$

计算公式：

$$\text{DFFITS}_i = t_i \times \sqrt{\frac{h_{ii}}{1-h_{ii}}}$$

判断准则：$|\text{DFFITS}_i| > 2\sqrt{\frac{p+1}{n}}$

DFBETAS 统计量

定义：衡量第 i 个观测对每个系数估计的影响

$$\text{DFBETAS}_{j(i)} = \frac{\hat{\beta}_j - \hat{\beta}_{j(i)}}{\hat{\sigma}_{(i)}\sqrt{[(\mathbf{X}'\mathbf{X})^{-1}]_{jj}}}$$

判断准则：$|\text{DFBETAS}_{j(i)}| > \frac{2}{\sqrt{n}}$

3.2.4 影响点的处理策略

诊断流程：1. 计算杠杆值 h_{ii}，识别高杠杆点 2. 计算学生化残差 t_i，识别异常值 3. 计算 Cook 距离 D_i，识别强影响点 4. 分析 DFFITS 和 DFBETAS，了解具体影响

处理方案：1. 数据核查：检查是否存在录入错误 2. 稳健回归：如 M 估计，最小化 $\sum_{i=1}^{n} \rho(e_i)$ 3. 变量变换：降低异常值的影响 4. 分段建模：对异常区域单独建立模型

3.3 异方差及处理

3.3.1 异方差的数学表述与影响

异方差的定义：

$$\text{Var}(\varepsilon_i) = \sigma_i^2 \neq \text{常数}$$

异方差对 OLS 估计的影响：

- 无偏性：$E(\hat{\beta}) = \beta$（仍然成立）

- 有效性：Var$(\hat{\beta})$ 不是最小方差

- 推断可靠性：标准误估计有偏，t 检验和 F 检验失效

方差-协方差矩阵：

$$\text{Var}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\boldsymbol{\Omega}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

其中 $\boldsymbol{\Omega} = \text{diag}(\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2)$

3.3.2 异方差的统计检验

Breusch-Pagan 检验

辅助回归模型：

$$e_i^2 = \alpha_0 + \alpha_1 x_{i1} + \cdots + \alpha_p x_{ip} + u_i$$

检验统计量：

$$\text{LM} = nR^2 \sim \chi^2(p)$$

其中 R^2 是辅助回归的决定系数。

White 检验

辅助回归：

$$e_i^2 = \alpha_0 + \sum_{j=1}^{p} \alpha_j x_{ij} + \sum_{j=1}^{p} \sum_{k \geq j}^{p} \alpha_{jk} x_{ij} x_{ik} + u_i$$

检验统计量：

$$\text{LM} = nR^2 \sim \chi^2(q)$$

其中 q 是辅助回归中自变量的个数（不包括常数项）。

3.4 异方差的修正

3.4.1 稳健标准误（Eicker-Huber-White 标准误）

HC0 估计量：

$$\text{Var}_{HC0}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\hat{\boldsymbol{\Omega}}_{HC0}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

其中 $\hat{\boldsymbol{\Omega}}_{HC0} = \text{diag}(e_1^2, e_2^2, \cdots, e_n^2)$

HC3 估计量（小样本更优）：

$$\hat{\boldsymbol{\Omega}}_{HC3} = \text{diag}\left(\frac{e_1^2}{(1-h_{11})^2}, \frac{e_2^2}{(1-h_{22})^2}, \cdots, \frac{e_n^2}{(1-h_{nn})^2}\right)$$

3.4.2 加权最小二乘法

权重选择：$w_i = 1/\sigma_i^2$

WLS 估计量：

$$\hat{\boldsymbol{\beta}}_{WLS} = (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{y}$$

其中 $\mathbf{W} = \text{diag}(w_1, w_2, \cdots, w_n)$

可行 GLS：当 σ_i^2 未知时，使用 $\hat{w}_i = 1/\hat{\sigma}_i^2$

其他方法：变量变换法

对数变换：$y^* = \ln(y)$，当 $\text{Var}(\varepsilon) \propto [E(y)]^2$ 时适用

平方根变换：$y^* = \sqrt{y}$，当 $\text{Var}(\varepsilon) \propto E(y)$ 时适用

3.5 案例分析

本章总结

核心公式回顾

1. 学生化残差：$t_i = \frac{e_i}{\hat{\sigma}_{(i)}\sqrt{1-h_{ii}}}$

2. 杠杆值：$h_{ii} = \mathbf{x}_i'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i$
3. Cook 距离：$D_i = \frac{e_i^2}{(p+1)\hat{\sigma}^2} \times \frac{h_{ii}}{(1-h_{ii})^2}$
4. 稳健标准误：$\text{Var}_{HC3}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\hat{\boldsymbol{\Omega}}_{HC3}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$
5. DFFITS：$\text{DFFITS}_i = t_i \times \sqrt{\frac{h_{ii}}{1-h_{ii}}}$

诊断决策框架

问题类型	诊断工具	判断标准	处理方案
异方差	残差 vs 拟合值图、BP 检验	残差方差非常数、BP 检验显著	稳健标准误、WLS、变量变换
非线性	残差 vs 拟合值图	残差呈现曲线模式	多项式项、样条回归、变量变换
高杠杆点	杠杆值	$h_{ii} > 2(p+1)/n$	数据核查、增加样本量
异常值	学生化残差	$\|t_i\| > t_{1-\alpha/2}(n-p-2)$	数据核查、稳健回归
强影响点	Cook 距离	$D_i > 4/n$	综合分析、谨慎处理

诊断流程体系

1. 初步筛查：残差 vs 拟合值图、Q-Q 图
2. 异方差检验：Breusch-Pagan 检验、White 检验
3. 影响分析：杠杆值、学生化残差、Cook 距离
4. 深入诊断：DFFITS、DFBETAS 分析
5. 处理实施：根据诊断结果选择适当修正方案

模型诊断是建立可靠预测模型的关键环节。通过系统性的诊断分析，我们能够识别模型缺陷，采取针对性改进措施，最终获得更加稳健、可靠的预测结果。

4 时间序列分析初步

本章导读

时间序列数据在回归分析框架下面临着独特的挑战，其中自相关问题尤为突出。本章将从回归分析的视角重新审视时间序列数据，系统介绍如何将自相关问题转化为回归模型的方差假设检验问题。通过学习，您将掌握在回归框架下诊断和处理自相关的方法，为时间序列数据建立更可靠的预测模型。

4.1 时间序列回归模型的基本框架

4.1.1 时间序列回归模型设定

基本回归模型：

$$Y_t = \beta_0 + \beta_1 X_{t1} + \beta_2 X_{t2} + \cdots + \beta_p X_{tp} + \varepsilon_t$$

其中 ε_t 是误差项，在时间序列背景下可能具有自相关性。

经典回归假设的违背：- 独立性假设：Cov$(\varepsilon_t, \varepsilon_s) = 0$（当 $t \neq s$）可能不成立 - 同方差性假设：Var$(\varepsilon_t) = \sigma^2$ 可能随时间变化

4.1.2 自相关的回归解释

一阶自回归误差结构：

$$\varepsilon_t = \rho \varepsilon_{t-1} + u_t$$

其中 $u_t \sim \text{WN}(0, \sigma_u^2)$，且 $|\rho| < 1$

方差-协方差矩阵：在自相关情况下，误差项的方差-协方差矩阵不再是标量矩阵：

$$\text{Var}(\varepsilon) = \sigma^2\Omega = \sigma^2 \begin{pmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{n-1} \\ \rho & 1 & \rho & \cdots & \rho^{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \cdots & 1 \end{pmatrix}$$

4.2 自相关的检验方法

4.2.1 基于残差的图形检验法

残差序列图：绘制残差 e_t 对时间 t 的散点图，观察是否存在：- 系统性模式 - 周期波动 - 趋势性变化

残差自相关图：计算残差的样本自相关函数：

$$r_k = \frac{\sum_{t=k+1}^{n} e_t e_{t-k}}{\sum_{t=1}^{n} e_t^2}$$

4.2.2 Durbin-Watson 检验

检验统计量：

$$d = \frac{\sum_{t=2}^{n} (e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$$

统计量的性质：- $d \approx 2(1 - r_1)$，其中 r_1 是一阶自相关系数 - $d \to 2$：无自相关 - $d \to 0$：正自相关 - $d \to 4$：负自相关

假设检验：- $H_0 : \rho = 0$（无自相关）- 通过 Durbin-Watson 表查找临界值

4.2.3 Breusch-Godfrey 检验（LM 检验）

辅助回归模型：

$$e_t = \alpha_0 + \alpha_1 X_{t1} + \cdots + \alpha_p X_{tp} + \phi_1 e_{t-1} + \cdots + \phi_m e_{t-m} + u_t$$

检验统计量：

$$\text{LM} = (n - m)R^2 \sim \chi^2(m)$$

其中 R^2 是辅助回归的决定系数。

检验步骤：1. 用 OLS 估计原模型，得到残差 e_t 2. 建立包含滞后残差的辅助回归 3. 计算 LM 统计量并进行检验

4.2.4 回归系数的有效性检验

OLS 估计量的方差偏误：在自相关情况下，OLS 估计量的真实方差为：

$$\text{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\Omega\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}$$

传统标准误的偏误：当存在正自相关时，传统标准误通常被低估，导致：- t 统计量被高估 - 置信区间过窄 - 假设检验的第一类错误率增加

4.3 自相关的处理方法

4.3.1 广义最小二乘法（**GLS**）

基本思想：通过变换将自相关误差转化为独立同分布误差。

变换矩阵：寻找矩阵 **P**，使得：

$$\mathbf{P}\Omega\mathbf{P}' = \mathbf{I}$$

GLS 估计量：

$$\hat{\beta}_{GLS} = (\mathbf{X}'\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}'\Omega^{-1}\mathbf{Y}$$

4.3.2 Cochrane-Orcutt 迭代法

算法步骤：

1. 初始估计：用 OLS 估计原模型，得到残差 e_t
2. 估计自相关系数：
$$\hat{\rho} = \frac{\sum_{t=2}^{n} e_t e_{t-1}}{\sum_{t=2}^{n} e_{t-1}^2}$$
3. 数据变换：
$$Y_t^* = Y_t - \hat{\rho}Y_{t-1}, \quad X_{tj}^* = X_{tj} - \hat{\rho}X_{t-1,j}$$
4. 重新估计：用 OLS 估计变换后的模型

5. 迭代：重复步骤 2-4 直到收敛

4.3.3 Prais-Winsten 估计

改进的 Cochrane-Orcutt 方法：保留第一个观测的变换：

$$Y_1^* = \sqrt{1-\hat{\rho}^2}Y_1, \quad X_{1j}^* = \sqrt{1-\hat{\rho}^2}X_{1j}$$

其他观测变换同 Cochrane-Orcutt 方法。

4.3.4 Newey-West 异方差自相关一致标准误

HAC 标准误估计量：

$$\text{Var}_{HAC}(\hat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{S}(\mathbf{X}'\mathbf{X})^{-1}$$

其中：

$$\mathbf{S} = \sum_{t=1}^{n} e_t^2 \mathbf{x}_t \mathbf{x}_t' + \sum_{l=1}^{m} w_l \sum_{t=l+1}^{n} (e_t e_{t-l} \mathbf{x}_t \mathbf{x}_{t-l}' + e_t e_{t-l} \mathbf{x}_{t-l} \mathbf{x}_t')$$

权重函数（Bartlett 核）：

$$w_l = 1 - \frac{l}{m+1}$$

带宽选择：

$$m = \lfloor 4(n/100)^{2/9} \rfloor$$

4.4 动态回归模型

4.4.1 分布滞后模型

有限分布滞后模型：

$$Y_t = \alpha + \sum_{i=0}^{k} \beta_i X_{t-i} + \varepsilon_t$$

无限分布滞后模型（Koyck 几何滞后）：

$$Y_t = \alpha + \beta \sum_{i=0}^{\infty} \lambda^i X_{t-i} + \varepsilon_t$$

4.4.2 自回归分布滞后模型（**ARDL**）

ARDL(p,q) 模型：

$$Y_t = \alpha + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=0}^{q} \beta_j X_{t-j} + \varepsilon_t$$

误差修正形式：

$$\Delta Y_t = \alpha + \gamma(Y_{t-1} - \theta X_{t-1}) + \sum_{i=1}^{p-1} \phi_i^* \Delta Y_{t-i} + \sum_{j=0}^{q-1} \beta_j^* \Delta X_{t-j} + \varepsilon_t$$

4.4.3 模型选择与诊断

信息准则：- AIC：$AIC = \ln(\hat{\sigma}^2) + \frac{2k}{n}$ - BIC：$BIC = \ln(\hat{\sigma}^2) + \frac{k \ln(n)}{n}$

残差诊断：- Durbin-Watson 检验 - Breusch-Godfrey 检验 - 残差自相关图分析

4.5 案例分析

本章总结

核心公式回顾

1. Durbin-Watson 统计量：$d = \frac{\sum_{t=2}^{n}(e_t - e_{t-1})^2}{\sum_{t=1}^{n} e_t^2}$
2. Breusch-Godfrey 检验：$LM = (n-m)R^2 \sim \chi^2(m)$
3. Cochrane-Orcutt 变换：$Y_t^* = Y_t - \hat{\rho} Y_{t-1}$
4. Newey-West 方差：$\text{Var}_{HAC}(\hat{\beta}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{S}(\mathbf{X}'\mathbf{X})^{-1}$
5. ARDL 模型：$Y_t = \alpha + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{j=0}^{q} \beta_j X_{t-j} + \varepsilon_t$

自相关检验方法比较

检验方法	适用场景	优点	局限性
Durbin-Watson	一阶自相关检验	计算简单	不适用于高阶自相关
Breusch-Godfrey	高阶自相关检验	适用性广	需要选择滞后阶数
残差图分析	初步诊断	直观易懂	主观性强
ACF/PACF 图	自相关结构识别	提供详细信息	需要经验判断

处理方法的决策框架

问题类型	推荐方法	适用条件
已知自相关结构	GLS 估计	Ω 已知或可估计
一阶自相关	Cochrane-Orcutt	中等样本量，一阶 AR 结构
大样本推断	Newey-West 标准误	任意形式的自相关和异方差
动态关系建模	ARDL 模型	存在理论支持的动态结构
预测导向	动态回归模型	关注预测精度而非因果推断

重要概念体系

1. 自相关检验：诊断回归模型方差假设的违背
2. 一致标准误：在违背经典假设下的有效推断
3. 可行 GLS：基于数据估计的广义最小二乘法
4. 动态设定：包含滞后项的扩展回归模型
5. 模型稳健性：对假设违背不敏感的估计方法

通过将时间序列问题纳入回归分析框架，我们能够利用成熟的回归技术来处理自相关问题，同时保持模型的解释性和易用性。这种方法为实际应用提供了实用的解决方案。

II 扩展——线性分析方法

我们将从经典的线性回归分析进行方法和应用上的扩展，以涵盖更广泛的统计建模技术。本章将介绍以下内容：- 方法的改进：逐步回归、岭回归和套索回归等技术，以提高模型的预测性能和解释能力。正则化和降维方法 - 应用的扩展：进入分类模型（如逻辑回归和判别分析）- 非线性模型的样条回归

5 降维

本章导读

随着大数据时代的到来，高维数据问题日益普遍。当自变量数目 p 接近甚至超过样本量 n 时，传统的最小二乘法面临严重挑战。本章将系统介绍正则化方法和降维技术这两类处理高维问题的主流方法，为您提供在 “维数灾难” 下建立稳定预测模型的有效工具。

5.1 多重共线性

5.1.1 高维问题

高维回归模型：

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

其中 $\mathbf{X} \in \mathbb{R}^{n \times p}$，当 $p \gg n$ 或 $p \approx n$ 时出现高维问题。

OLS 的失效：

- 当 $p > n$ 时，$\mathbf{X}'\mathbf{X}$ 不可逆
- 当 $p \approx n$ 时，估计方差极大
- 预测性能下降，模型过拟合

稀疏性假设

精确稀疏性：

$$||\boldsymbol{\beta}||_0 = \#\{j : \beta_j \neq 0\} \ll p$$

近似稀疏性：大多数系数的绝对值很小，只有少数系数较大。

5.1.2 多重共线性原因与表现

考虑线性回归模型：

$$\mathbf{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

定义 **1**（完全多重共线性）：如果存在不全为零的常数 $c_1, c_2, \ldots, c_p$，使得：

$$c_1 x_1 + c_2 x_2 + \cdots + c_p x_p = 0 \quad \text{（几乎处处成立）}$$

则称自变量间存在完全多重共线性（perfect multicollinearity）。

在矩阵形式下，设设计矩阵 $\mathbf{X} = [\mathbf{1}, \mathbf{x}_1, \ldots, \mathbf{x}_p] \in \mathbb{R}^{n \times (p+1)}$，完全多重共线性意味着：

$$\text{rank}(\mathbf{X}) < p + 1$$

即 $\mathbf{X}^\top \mathbf{X}$ 是奇异的，不存在唯一的最小二乘解。

定义 **2**（近似多重共线性）：如果存在不全为零的常数 $c_1, c_2, \ldots, c_p$，使得：

$$c_1 x_1 + c_2 x_2 + \cdots + c_p x_p \approx 0$$

即自变量间存在高度但不完全的线性关系，则称为近似多重共线性。

多重共线系的原因：

- 变量之间的共同变化趋势
- 时间序列数据趋势，滞后变量
- 经济指标关系约束或者虚拟变量陷阱
- 样本量不足
- 截面数据的规模效应

多重共线性的表现

在回归结果中：

系数估计异常

- 符号异常：系数符号与理论预期相反
- 数值异常大：系数绝对值异常大
- 统计不显著但联合显著：

- 单个 t 检验：$H_0 : \beta_j = 0$ 不拒绝
- 整体 F 检验：$H_0 : \beta_1 = \cdots = \beta_p = 0$ 拒绝

稳定性检验

- 删除观测影响：删除个别观测导致系数估计剧烈变化
- 添加/删除变量：添加或删除其他变量导致系数符号或大小显著变化

5.1.3 多重共线性的影响：

- 参数估计方面：参数估计仍然是无偏的，但是参数估计方差会很大
- 假设检验方面：F 检验会显著，但是 t 检验会失效
- 预测：预测变得不稳定
- 参数估计的值不稳定，正负符号可能不符合预期

多重共线性的主要影响在于参数估计量的方程膨胀，下面给出简单的推导过程：

这里只考虑考虑复共线性，即设计矩阵 X 的列向量之间存在近似线性相关关系。

数学表述为：

存在非零向量 $c = (c_1, c_2, \dots, c_p)' \neq 0$，使得：

$$c_1 X_1 + c_2 X_2 + \cdots + c_p X_p \approx 0$$

其中 $X_j \in \mathbb{R}^n$ 表示第 j 个自变量的观测向量。

考虑 Gram 矩阵 $S = X'X$，其特征值问题为 $Sv = \lambda v$。利用 Rayleigh 商的极值性质：

$$\lambda_{min} = \min_{v \neq 0} \frac{v'Sv}{v'v} = \min_{v \neq 0} \frac{\|Xv\|^2}{\|v\|^2}$$

当存在多重共线性时，取 $v = c$（近似线性相关的系数向量），则有：

$$\lambda_{min} \leq \frac{\|Xc\|^2}{\|c\|^2} \approx 0$$

Gram 矩阵的谱分解为：

$$S = X'X = \sum_{j=1}^{p} \lambda_j v_j v_j'$$

其逆矩阵相应为：

$$(X'X)^{-1} = \sum_{j=1}^{p} \frac{1}{\lambda_j} v_j v_j'$$

这时，逆矩阵中每个特征方向的贡献被 $\frac{1}{\lambda_j}$ 放大。当某些 $\lambda_j \to 0$ 时，对应项 $\frac{1}{\lambda_j} \to \infty$。

普通最小二乘估计的方差-协方差矩阵为：

$$\text{Var}(\hat{\beta}) = \sigma^2 (X'X)^{-1} = \sigma^2 \sum_{j=1}^{p} \frac{1}{\lambda_j} v_j v_j'$$

第 j 个参数估计的方差为：

$$\text{Var}(\hat{\beta}_j) = \sigma^2 [(X'X)^{-1}]_{jj} = \sigma^2 \sum_{k=1}^{p} \frac{v_{kj}^2}{\lambda_k}$$

每个参数估计的方差可分解为在各特征向量方向上的贡献之和，权重为 $\frac{v_{kj}^2}{\lambda_k}$。

设 $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0$，当存在严重多重共线性时，假设前 m 个特征值显著大于零，后 $p-m$ 个特征值接近零：

$$\text{Var}(\hat{\beta}_j) = \sigma^2 \left(\sum_{k=1}^{m} \frac{v_{kj}^2}{\lambda_k} + \sum_{k=m+1}^{p} \frac{v_{kj}^2}{\lambda_k} \right)$$

其中第二项由于 $\lambda_k \approx 0$ 而急剧增大：

$$\sum_{k=m+1}^{p} \frac{v_{kj}^2}{\lambda_k} \to \infty$$

小特征值对应的特征方向在方差表达式中产生巨大贡献。

在多元线性回归模型 $Y = X\beta + \varepsilon$ 中，第 j 个参数估计 $\hat{\beta}_j$ 的方差为：

$$\text{Var}(\hat{\beta}_j) = \frac{\sigma^2}{(1 - R_j^2) \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2}$$

将方差重写为：

$$\text{Var}(\widehat{\beta}_j) = \underbrace{\frac{\sigma^2}{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}}_{\text{简单回归方差}} \times \underbrace{\frac{1}{1-R_j^2}}_{VIF_j}$$

方差膨胀因子 **(Variance Inflation Factor, VIF)** 是衡量多重共线性严重程度的核心指标，定义为：

$$VIF_j = \frac{1}{1-R_j^2}$$

其中：

- R_j^2 是第 j 个自变量 X_j 对其他所有自变量 $X_1, X_2, \ldots, X_{j-1}, X_{j+1}, \ldots, X_p$ 进行回归所得的决定系数
- VIF_j 度量了由于与其他自变量相关而导致的第 j 个参数估计方差的膨胀倍数
- 诊断标准：- $\text{VIF}_j > 10$：存在严重多重共线性 - $\text{VIF}_j > 5$：存在中度多重共线性

这清晰地展示了方差膨胀因子的含义：参数估计方差相对于无多重共线性情况下的膨胀倍数。

将方差代入：

$$VIF_j = \frac{1}{1-R_j^2} = \frac{\sum_{i=1}^{n}(x_{ij}-\bar{x}_j)^2}{s^2} \cdot [(X'X)^{-1}]_{jj}$$

可知：

$$VIF_j \propto \sum_{k=1}^{p} \frac{v_{kj}^2}{\lambda_k}$$

方差膨胀因子本质上度量了参数方差在各特征方向上的累积放大效应。

矩阵 $X'X$ 的条件数定义为：

$$\kappa(X'X) = \frac{\lambda_{max}}{\lambda_{min}}$$

多重共线性下 $\lambda_{min} \to 0$，导致：

- $\kappa(X'X) \to \infty$
- 矩阵求逆数值不稳定
- 估计量对数据扰动敏感

设计矩阵的病态性：条件数 $\kappa(\mathbf{X}'\mathbf{X}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$ 过大，其中 λ 是特征值。

5.1.4 多重共线性的解决方法

- 删除变量
- 合并变量
- 变量变换
- 增加样本量
- 正则化方法
- 降维方法

5.2 自变量选择

5.2.1 子集回归模型

考虑线性回归模型：

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

其中：

- $\mathbf{y} \in \mathbb{R}^n$：响应变量
- $\mathbf{X} \in \mathbb{R}^{n \times p}$：设计矩阵（包含截距项）
- $\boldsymbol{\beta} \in \mathbb{R}^p$：回归系数
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n$：误差项，$\boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$

子集模型

设 $\mathcal{M}$ 表示一个候选模型，包含 k 个自变量（不包括截距），则：

$$\mathbf{y} = \mathbf{X}_{\mathcal{M}}\boldsymbol{\beta}_{\mathcal{M}} + \boldsymbol{\varepsilon}_{\mathcal{M}}$$

其中 $k = |\mathcal{M}|$。

- 残差平方和：

  $$\text{RSS}_{\mathcal{M}} = \|\mathbf{y} - \hat{\mathbf{y}}_{\mathcal{M}}\|^2 = \mathbf{y}^\top (\mathbf{I} - \mathbf{P}_{\mathcal{M}})\mathbf{y}$$

  其中 $\mathbf{P}_{\mathcal{M}} = \mathbf{X}_{\mathcal{M}}(\mathbf{X}_{\mathcal{M}}^\top \mathbf{X}_{\mathcal{M}})^{-1}\mathbf{X}_{\mathcal{M}}^\top$ 是投影矩阵。

- 最大似然估计：在正态假设下，似然函数为：

$$L(\boldsymbol{\beta}_{\mathcal{M}}, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left(-\frac{\text{RSS}_{\mathcal{M}} + \|\mathbf{X}_{\mathcal{M}}\boldsymbol{\beta}_{\mathcal{M}} - \mathbf{X}_{\mathcal{M}}\hat{\boldsymbol{\beta}}_{\mathcal{M}}\|^2}{2\sigma^2}\right)$$

  最大似然估计为：

$$\hat{\boldsymbol{\beta}}_{\mathcal{M}} = (\mathbf{X}_{\mathcal{M}}^\top \mathbf{X}_{\mathcal{M}})^{-1}\mathbf{X}_{\mathcal{M}}^\top \mathbf{y}, \quad \hat{\sigma}^2_{\mathcal{M}} = \frac{\text{RSS}_{\mathcal{M}}}{n}$$

- 对数似然：

$$\ell(\mathcal{M}) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\hat{\sigma}^2_{\mathcal{M}}) - \frac{n}{2}$$

5.2.2 自变量选择标准

在线性回归模型中，当有大量潜在自变量时，我们需要选择最优的子集。自变量选择的目标是：

- 提高预测精度：减少过拟合，降低预测方差
- 增强模型解释性：识别真正重要的变量
- 简化模型：减少数据收集和计算成本

1 决定系数 R^2

$$R^2_{\mathcal{M}} = 1 - \frac{\text{RSS}_{\mathcal{M}}}{\text{TSS}}$$

其中 $\text{TSS} = \sum_{i=1}^n (y_i - \bar{y})^2$ 是总平方和。

问题：R^2 随自变量增加而单调增加，倾向于选择大模型。

为了惩罚模型复杂度：

$$R^2_{\text{adj},\mathcal{M}} = 1 - \frac{\text{RSS}_{\mathcal{M}}/(n-k-1)}{\text{TSS}/(n-1)} = 1 - \frac{n-1}{n-k-1}(1 - R^2_{\mathcal{M}})$$

性质：- 仅当新变量的 t 统计量绝对值大于 1 时，R^2_{adj} 才会增加 - 但仍不是一致模型选择准则

2 Akaike 信息准则 (AIC)

AIC 基于 Kullback-Leibler 散度最小化。设真实分布为 $g(y)$，模型分布为 $f(y|\theta)$，K-L 散度为：

$$I(g; f) = \int g(y)\log g(y)dy - \int g(y)\log f(y|\theta)dy$$

最小化 K-L 散度等价于最大化 $\mathbb{E}_g[\log f(y|\theta)]$。AIC 估计：

$$\text{AIC} = -2\log L(\hat{\theta}) + 2k$$

对于线性回归：

$$\text{AIC} = n\log\left(\frac{\text{RSS}_{\mathcal{M}}}{n}\right) + 2(k+1) + n[\log(2\pi)+1]$$

去掉常数项，常用简化形式：

$$\text{AIC} = n\log(\hat{\sigma}^2_{\mathcal{M}}) + 2k$$

或等价的：

$$\text{AIC} = \frac{\text{RSS}_{\mathcal{M}}}{n}e^{2k/n}$$

版本差异：

- AIC：$n\log(\text{RSS}/n) + 2k$
- AICc（小样本校正）：$n\log(\text{RSS}/n) + 2k + \frac{2k(k+1)}{n-k-1}$
- AICu：$n\log[\text{RSS}/(n-k)] + 2k$

3 贝叶斯信息准则 (BIC)

BIC 源于贝叶斯因子的 Laplace 近似。模型 $\mathcal{M}$ 的后验概率：

$$P(\mathcal{M}|\mathbf{y}) \propto P(\mathbf{y}|\mathcal{M})P(\mathcal{M})$$

边缘似然：

$$P(\mathbf{y}|\mathcal{M}) = \int L(\boldsymbol{\beta}_{\mathcal{M}}, \sigma^2)\pi(\boldsymbol{\beta}_{\mathcal{M}}, \sigma^2)d\boldsymbol{\beta}_{\mathcal{M}}d\sigma^2$$

使用 Jeffrey 无信息先验 $\pi(\boldsymbol{\beta}, \sigma^2) \propto 1/\sigma^2$，可得：

$$-2\log P(\mathbf{y}|\mathcal{M}) \approx n\log(\hat{\sigma}^2_{\mathcal{M}}) + k\log n$$

BIC 公式

$$\text{BIC} = n\log\left(\frac{\text{RSS}_{\mathcal{M}}}{n}\right) + k\log n$$

或等价地：

$$\text{BIC} = \frac{\text{RSS}_{\mathcal{M}}}{n} n^{k/n}$$

AIC 与 BIC 比较

准则	惩罚项	目标	渐近性质	模型倾向
AIC	$2k$	预测最优	非一致	偏大模型
BIC	$k \log n$	真实模型	一致	偏小模型

一致性：当 $n \to \infty$ 时，BIC 以概率 1 选择真实模型（如果真实模型在候选集中），而 AIC 可能选择过参数化模型。

效率：在错误设定下，AIC 选择的模型可能有更好的预测性能。

- AIC：渐近有效（当真实模型不在候选集中时，选择的模型预测误差接近最优）
- BIC：渐近一致但非有效

模拟研究显示：1. 当信噪比低、样本量小时，AIC 倾向于选择过大的模型 2. BIC 在样本量足够大时表现更好 3. C_p 与 AIC 渐近等价（$C_p = \text{AIC}/n + \text{常数}$）高维情况 ($p > n$)

传统准则失效，需要改进：- 扩展 BIC(EBIC)：$\text{BIC} + \gamma \log \binom{p}{k}$，$\gamma \in [0, 1]$ - 高维 AIC：使用有效自由度代替 k

4 Mallow's C_p 统计量

考虑标准化预测误差：

$$\Gamma_p(\mathcal{M}) = \frac{1}{\sigma^2} \mathbb{E}[\|\mathbf{X}\hat{\boldsymbol{\beta}}_{\mathcal{M}} - \mathbf{X}\boldsymbol{\beta}\|^2]$$

可以证明：

$$\mathbb{E}[\text{RSS}_{\mathcal{M}}] = (n-k)\sigma^2 + \|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}_{\mathcal{M}}\boldsymbol{\beta}^*_{\mathcal{M}}\|^2$$

其中 $\boldsymbol{\beta}^*_{\mathcal{M}}$ 是 $\boldsymbol{\beta}$ 在子空间上的投影。因此：

$$\mathbb{E}[C_p(\mathcal{M})] \approx \frac{1}{\sigma^2}\|\mathbf{X}\boldsymbol{\beta} - \mathbf{X}_{\mathcal{M}}\boldsymbol{\beta}^*_{\mathcal{M}}\|^2 + k$$

假设全模型（包含所有 p 个变量）是正确的，则：

$$C_p(\mathcal{M}) = \frac{\text{RSS}_{\mathcal{M}}}{\hat{\sigma}^2_{\text{full}}} + 2k - n$$

其中 $\hat{\sigma}^2_{\text{full}} = \text{RSS}_{\text{full}}/(n-p)$。

可以证明 C_p 是 $\Gamma_p(k)$ 的无偏估计

解释 - $C_p \approx k$：模型无偏 - $C_p > k$：模型有偏 - $C_p < k$：可能过拟合或随机波动

选择 C_p 小且接近 k 的模型。

准则选择指南

- 预测导向：用 AIC、C_p 或交叉验证
- 解释导向：用 BIC 或调整 R^2
- 大数据：BIC 更可靠（$n \geq 1000$）
- 小数据：AICc 或交叉验证

自变量选择需要权衡偏差-方差、解释性与预测精度。没有单一最优准则，实际应用中应：1. 考虑研究目标（预测 vs 解释）2. 结合多种准则 3. 进行模型诊断和验证 4. 考虑计算成本和可解释性

核心公式总结：

- $R^2_{\text{adj}} = 1 - \frac{n-1}{n-k-1}(1-R^2)$
- $\text{AIC} = n\log(\text{RSS}/n) + 2k$
- $\text{BIC} = n\log(\text{RSS}/n) + k\log n$
- $C_p = \text{RSS}/\hat{\sigma}^2_{\text{full}} + 2k - n$

选择准则本身也有不确定性，模型平均和集成方法可以进一步改善预测性能。

最优子集回归

最优子集选择：对于每个 $k = 1, 2, \cdots, p$，选择使 RSS 最小的包含 k 个自变量的模型。

计算挑战：需要评估 2^p 个模型，计算不可行。

逐步回归：- 前向选择 - 后向剔除 - 逐步选择

局限性：- 路径依赖 - 多重检验问题 - 标准误低估

5.2.3 最优子集回归：

对于给定的模型大小 k，最优子集问题是组合优化问题：

$$\mathcal{M}_k^* = \arg\min_{\mathcal{M}:|\mathcal{M}|=k} \text{RSS}(\mathcal{M})$$

总搜索空间大小为 $\sum_{k=0}^{p}\binom{p}{k} = 2^p$，是 NP-hard 问题。

5.2.4 逐步回归

前向选择

前向选择等价于求解以下序列优化问题：

$$j_t = \arg\max_{j \notin \mathcal{M}_{t-1}} \frac{|\mathbf{x}_j^\top \mathbf{r}_{t-1}|}{\|\mathbf{x}_j\|}$$

其中 $\mathbf{r}_{t-1}$ 是第 $t-1$ 步的残差向量。

与正交匹配追踪（OMP）的关系

前向选择是 OMP 的特例。在正交化版本中：1. 计算当前残差与所有预测变量的相关系数 2. 选择最大相关系数对应的变量 3. 正交化更新残差

数学表达：

$$\mathbf{q}_t = \frac{(\mathbf{I} - \mathbf{Q}_{t-1}\mathbf{Q}_{t-1}^\top)\mathbf{x}_{j_t}}{\|(\mathbf{I} - \mathbf{Q}_{t-1}\mathbf{Q}_{t-1}^\top)\mathbf{x}_{j_t}\|}$$

其中 $\mathbf{Q}_{t-1} = [\mathbf{q}_1, \dots, \mathbf{q}_{t-1}]$ 为正交基。

后向删除的

后向删除基于假设检验：

$$H_0 : \beta_j = 0 \quad \text{vs} \quad H_1 : \beta_j \neq 0$$

使用 F 统计量：

$$F_j = \frac{(\text{RSS}_{-j} - \text{RSS})/1}{\text{RSS}/(n-k)}$$

其中 RSS_{-j} 是删除变量 j 后的 RSS。

双向逐步回归

算法描述

结合前向和后向步骤：1. 初始化：$\mathcal{M} = \emptyset$ 2. 迭代直到收敛：- 前向步：尝试添加一个变量 - 后向步：尝试删除一个变量 - 每次选择使 RSS 减少最多（或增加最少）的操作 - 通常基于统计显著性（如 p 值）决定

基于 F 检验的版本

常用的停止准则是 F 统计量：- 添加变量：计算偏 F 统计量

$$F_{\text{add}} = \frac{(\text{RSS}_{\text{old}} - \text{RSS}_{\text{new}})/1}{\text{RSS}_{\text{new}}/(n - |\mathcal{M}| - 1)}$$

如果 $F_{\text{add}} > F_{\text{in}}$（进入临界值），则添加该变量 - 删除变量：计算 F 统计量

$$F_{\text{drop}} = \frac{(\text{RSS}_{\text{without}} - \text{RSS}_{\text{with}})/1}{\text{RSS}_{\text{with}}/(n - |\mathcal{M}|)}$$

如果 $F_{\text{drop}} < F_{\text{out}}$（删除临界值），则删除该变量

** 与 LAR 算法联系 **

最小角回归（Least Angle Regression, LAR）可以看作逐步回归的连续版本：1. 开始时，所有系数为 0 2. 找到与当前残差最相关的变量 3. 沿该方向移动系数，直到另一个变量与残差的相关性相等 4. 沿这两个变量的角平分线方向移动 5. 重复直到所有变量都进入模型

总结与比较

方法	优点	缺点	计算复杂度	适用场景
最优子集	全局最优	计算不可行（$p > 40$）	$O(2^p)$	$p \leq 20$，精确解重要
前向选择	计算快，可处理 $p > n$	贪心，可能不是最优	$O(p^2)$	高维数据，计算资源有限
后向删除	考虑变量间交互	需要 $n > p$	$O(p^3)$	p 中等，全模型可拟合
双向逐步	更灵活	可能过拟合	$O(p^3)$	中等维度，平衡探索利用

实际建议

- 当 $p \leq 20$ 时，考虑最优子集（或所有子集）
- 当 $p > n$ 时，使用前向选择或 LASSO
- 当关心解释性时，逐步回归可能优于黑箱方法
- 总是使用交叉验证选择最终模型大小

数学洞察

逐步回归本质上是在逐步优化的基础上进行的坐标下降。设目标函数：

$$J(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_0$$

其中 $\|\boldsymbol{\beta}\|_0 = \sum_{j=1}^{p} I(\beta_j \neq 0)$。这是 NP 难问题，逐步回归提供了近似解。

现代发展表明，凸松弛（如 LASSO）在理论和计算上都有优势，但逐步回归因其简单直观，仍在实践中广泛使用。

5.2.6 调参策略

网格搜索：在 λ（和 α）的网格上计算交叉验证误差。

信息准则：

- AIC：$AIC = n\ln(\hat{\sigma}^2) + 2k$
- BIC：$BIC = n\ln(\hat{\sigma}^2) + k\ln(n)$

一倍标准误准则：选择调优参数，使得交叉验证误差在最小误差的一倍标准误范围内。

5.3 主成分回归

5.3.1 主成分回归原理

主成分回归（Principal Component Regression, PCR）是多元线性回归与主成分分析（PCA）的结合。当自变量存在多重共线性时，普通最小二乘法（OLS）估计的方差会变得很大，PCR 通过先对自变量进行主成分分析，然后选取部分主成分作为新的自变量进行回归，从而改善估计的稳定性。

5.3.2 理论推导过程

设线性回归模型：

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

其中：

- $\mathbf{y} \in \mathbb{R}^n$ 是因变量观测向量
- $\mathbf{X} \in \mathbb{R}^{n\times p}$ 是自变量设计矩阵（已中心化）
- $\boldsymbol{\beta} \in \mathbb{R}^p$ 是回归系数向量
- $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ 是误差向量，满足 $\mathbb{E}(\boldsymbol{\varepsilon}) = \mathbf{0}$, $\mathrm{Var}(\boldsymbol{\varepsilon}) = \sigma^2\mathbf{I}_n$

假设 $\mathbf{X}$ 已中心化（即每列均值为 0），则无需考虑截距项。

则 $\mathbf{X}$ 的样本协方差矩阵为：

$$\mathbf{S} = \frac{1}{n-1}\mathbf{X}^\top\mathbf{X}$$

对 $\mathbf{S}$ 进行特征分解：

$$\mathbf{S} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$$

其中：

- $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_p)$，$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0$ 是特征值
- $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_p]$ 是正交特征向量矩阵，满足 $\mathbf{V}^\top \mathbf{V} = \mathbf{I}_p$

主成分得分矩阵为：

$$\mathbf{Z} = \mathbf{X}\mathbf{V}$$

其中 $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_p]$，第 j 个主成分 $\mathbf{z}_j = \mathbf{X}\mathbf{v}_j$。

原模型 $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ 可写为：

$$\mathbf{y} = \mathbf{X}\mathbf{V}\mathbf{V}^\top\boldsymbol{\beta} + \boldsymbol{\varepsilon} = \mathbf{Z}\boldsymbol{\alpha} + \boldsymbol{\varepsilon}$$

其中 $\boldsymbol{\alpha} = \mathbf{V}^\top\boldsymbol{\beta}$ 是主成分回归中的系数向量。

选择前 k ($k \leq p$) 个主成分，得到约简模型：

$$\mathbf{y} = \mathbf{Z}_k\boldsymbol{\alpha}_k + \boldsymbol{\varepsilon}^*$$

其中：

- $\mathbf{Z}_k = \mathbf{X}\mathbf{V}_k$，$\mathbf{V}_k = [\mathbf{v}_1, \dots, \mathbf{v}_k]$
- $\boldsymbol{\alpha}_k = (\alpha_1, \dots, \alpha_k)^\top$

对约简模型应用最小二乘法：

$$\hat{\boldsymbol{\alpha}}_k = (\mathbf{Z}_k^\top\mathbf{Z}_k)^{-1}\mathbf{Z}_k^\top\mathbf{y}$$

由于 $\mathbf{Z}_k^\top\mathbf{Z}_k = (n-1)\mathbf{\Lambda}_k$，其中 $\mathbf{\Lambda}_k = \text{diag}(\lambda_1, \dots, \lambda_k)$，可得：

$$\hat{\boldsymbol{\alpha}}_k = \frac{1}{n-1}\mathbf{\Lambda}_k^{-1}\mathbf{Z}_k^\top\mathbf{y}$$

分量形式为：

$$\hat{\alpha}_j = \frac{\mathbf{z}_j^\top\mathbf{y}}{(n-1)\lambda_j}, \quad j = 1, \dots, k$$

原始参数 $\boldsymbol{\beta}$ 的 PCR 估计为：

$$\hat{\boldsymbol{\beta}}_{\text{PCR}} = \mathbf{V}_k\hat{\boldsymbol{\alpha}}_k = \sum_{j=1}^{k}\mathbf{v}_j\hat{\alpha}_j$$

代入 $\hat{\alpha}_j$ 表达式：

$$\hat{\boldsymbol{\beta}}_{\text{PCR}} = \sum_{j=1}^{k} \mathbf{v}_j \frac{\mathbf{z}_j^\top \mathbf{y}}{(n-1)\lambda_j} = \sum_{j=1}^{k} \mathbf{v}_j \frac{\mathbf{v}_j^\top \mathbf{X}^\top \mathbf{y}}{(n-1)\lambda_j}$$

算法步骤总结

- 中心化 **X** 和 **y**
- 计算 **X** 的协方差矩阵 **S**
- 对 **S** 进行特征分解，得到 **V** 和 $\boldsymbol{\Lambda}$
- 选择主成分个数 k
- 计算主成分得分 $\mathbf{Z}_k = \mathbf{X}\mathbf{V}_k$
- 用 OLS 估计 **y** 对 $\mathbf{Z}_k$ 的回归系数 $\hat{\boldsymbol{\alpha}}_k$
- 转换回原始空间：$\hat{\boldsymbol{\beta}}_{\text{PCR}} = \mathbf{V}_k \hat{\boldsymbol{\alpha}}_k$

选择主成分个数 k 的常用方法：

1. 累积方差贡献率：选择最小的 k 使得

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \geq \tau$$

   通常取 $\tau = 0.7 \sim 0.9$

2. 交叉验证：通过最小化预测误差的交叉验证估计来选择 k
3. 特征值大于 **1** 准则（适用于标准化数据）

5.3.3 估计量的性质

1. 偏差

PCR 估计是有偏估计（除非 $k = p$）：

$$\mathbb{E}[\hat{\boldsymbol{\beta}}_{\text{PCR}}] = \mathbf{V}_k \mathbf{V}_k^\top \boldsymbol{\beta}$$

偏差为：

$$\text{Bias}(\hat{\boldsymbol{\beta}}_{\text{PCR}}) = (\mathbf{V}_k \mathbf{V}_k^\top - \mathbf{I})\boldsymbol{\beta} = -\sum_{j=k+1}^{p} \mathbf{v}_j \mathbf{v}_j^\top \boldsymbol{\beta}$$

2. 方差

$$\text{Var}(\hat{\boldsymbol{\beta}}_{\text{PCR}}) = \sigma^2 \mathbf{V}_k \boldsymbol{\Lambda}_k^{-1} \mathbf{V}_k^\top = \sigma^2 \sum_{j=1}^{k} \frac{\mathbf{v}_j \mathbf{v}_j^\top}{(n-1)\lambda_j}$$

3. 均方误差

$$\text{MSE}(\hat{\boldsymbol{\beta}}_{\text{PCR}}) = \text{Var}(\hat{\boldsymbol{\beta}}_{\text{PCR}}) + \|\text{Bias}(\hat{\boldsymbol{\beta}}_{\text{PCR}})\|^2$$

当存在小特征值时，PCR 通过牺牲偏差来显著降低方差，从而可能获得比 OLS 更小的均方误差。

5.4 偏最小二乘回归

5.4.1 PLS 的基本思想

潜变量构建：寻找潜变量 $\mathbf{t} = \mathbf{X}\mathbf{w}$，使得：1. $\mathbf{t}$ 的方差尽可能大 2. $\mathbf{t}$ 与 $\mathbf{y}$ 的相关性尽可能大

优化问题：

$$\begin{aligned} \max_{\mathbf{w}} \quad & \text{Cov}(\mathbf{X}\mathbf{w}, \mathbf{y}) \\ \text{s.t.} \quad & \|\mathbf{w}\|_2 = 1 \end{aligned}$$

5.4.2 PLS 算法（NIPALS）

初始化

令：

$$E_0 = X - \bar{X}, \quad f_0 = y - \bar{y}$$

其中 E_0 和 f_0 是中心化后的数据。

第 h 个成分的提取

步骤 1：计算权重向量 w_h

$$w_h = \frac{E_{h-1}^T f_{h-1}}{\|E_{h-1}^T f_{h-1}\|}$$

解释：找到 X 中与当前 y 残差最相关的方向。

步骤 2：计算得分向量 t_h

$$t_h = E_{h-1} w_h$$

解释：X 在权重方向上的投影，即潜变量。

步骤 3：计算 X 载荷向量 p_h

$$p_h = \frac{E_{h-1}^T t_h}{t_h^T t_h}$$

解释：t_h 对原始 X 变量的解释程度。

步骤 4：计算 y 载荷 c_h

$$c_h = \frac{f_{h-1}^T t_h}{t_h^T t_h}$$

解释：t_h 对 y 的解释程度（标量）。

步骤 5：更新残差

$$E_h = E_{h-1} - t_h p_h^T$$
$$f_h = f_{h-1} - c_h t_h$$

5.5 案例分析

本章总结

核心公式回顾 1. 主成分：$\mathbf{Z} = \mathbf{XV}$，其中 $\mathbf{S} = \mathbf{V\Lambda V}'$ 2. 坐标下降：$\beta_j \leftarrow S\left(\frac{1}{n}\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\right)$

方法选择指南

方法	核心思想	优点	缺点	适用场景
最优子集	穷举搜索	全局最优解	计算量大，不可行	变量数较少 (<20)

方法	核心思想	优点	缺点	适用场景
逐步回归	逐步筛选	计算高效，自动选择	局部最优，结果不稳定	初步变量筛选
主成分回归	正交变换	消除共线性，稳定	可解释性差	强共线性，预测为主
偏最小二乘	协同降维	利用 Y 信息，预测好	计算复杂，理论深	高维数据，预测精度要求高

实践建议

1. 数据预处理：标准化自变量，确保惩罚的公平性
2. 模型评估：使用交叉验证，避免过拟合
3. 结果验证：在测试集上评估预测性能
4. 稳健性检查：尝试不同方法，比较结果一致性
5. 领域知识：结合实际问题背景选择合适方法

“传统变量选择方法如逐步回归，虽然应用广泛，但存在选择结果不稳定、忽略变量间理论关系等局限性。而降维方法如主成分回归，虽然解决了共线性问题，却损失了原始变量的可解释性。20 世纪 70 年代以来，一种基于惩罚似然的新思路——正则化方法逐渐发展起来，它通过在损失函数中加入惩罚项，同时实现参数估计和变量选择。下一章将系统介绍岭回归、Lasso 及其变体，这些方法不仅能够处理多重共线性，还能产生稀疏解，在高维数据分析中表现出显著优势。”

6 正则化

本章导读

随着大数据时代的到来，高维数据问题日益普遍。当自变量数目 p 接近甚至超过样本量 n 时，传统的最小二乘法面临严重挑战。本章将系统介绍正则化方法和降维技术这两类处理高维问题的主流方法，为您提供在 “维数灾难” 下建立稳定预测模型的有效工具。

6.1 正则化方法理论基础

6.1.1 正则化的一般框架

复习前面普通最小二乘法的目标函数为：

$$\min_{\beta} \|y - X\beta\|_2^2$$

即：

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$$

解为：

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

当 $X^T X$ 接近奇异时（多重共线性），OLS 估计变得不稳定，方差很大。

下面通过添加正则化项（惩罚项）来解决这个问题：

惩罚最小二乘：

$$\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$$
$$\text{s.t.} \quad P(\beta) \leq t$$

即：

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \{||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda P(\boldsymbol{\beta})\}$$

其中 $P(\boldsymbol{\beta})$ 是惩罚函数，$\lambda \geq 0$ 是调节参数。

6.1.2 偏差-方差权衡的数学表达

期望预测误差分解：

$$\mathbb{E}[(y_0 - \mathbf{x}_0'\hat{\boldsymbol{\beta}})^2] = \text{Var}(\mathbf{x}_0'\hat{\boldsymbol{\beta}}) + [\text{Bias}(\mathbf{x}_0'\hat{\boldsymbol{\beta}})]^2 + \sigma^2$$

正则化的作用：通过引入偏差来降低方差，从而减少期望预测误差。

6.2 岭回归

6.2.1 岭估计的定义与性质

L2 正则化：

$$\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$$
$$\text{s.t.} \quad ||\boldsymbol{\beta}||_2^2 \leq t$$

即：

$$\min_{\beta} \{\|y - X\beta\|_2^2 + \lambda\|\beta\|_2^2\}$$

其中 $\lambda > 0$ 是正则化参数。将目标函数展开

$$J(\beta) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$$

展开第一项：

$$(y - X\beta)^T(y - X\beta) = y^Ty - 2y^TX\beta + \beta^TX^TX\beta$$

因此：

$$J(\beta) = y^T y - 2y^T X\beta + \beta^T X^T X\beta + \lambda\beta^T\beta$$

对 β 求梯度：

$$\nabla J(\beta) = -2X^T y + 2X^T X\beta + 2\lambda\beta$$

令梯度为零：

$$-2X^T y + 2X^T X\beta + 2\lambda\beta = 0$$

整理得：

$$(X^T X + \lambda I)\beta = X^T y$$

解得岭回归估计：

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T y$$

岭估计量：

$$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$$

理论性质：

- 有偏性：$\mathbb{E}[\hat{\boldsymbol{\beta}}_{\text{ridge}}] = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta}$
- 方差：$\text{Var}(\hat{\boldsymbol{\beta}}_{\text{ridge}}) = \sigma^2(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}$

6.2.2 岭参数的选择

交叉验证：

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\hat{\boldsymbol{\beta}}^{(-i)}(\lambda))^2$$

广义交叉验证 (GCV)：

$$GCV(\lambda) = \frac{\frac{1}{n}||(\mathbf{I} - \mathbf{S}(\lambda))\mathbf{y}||_2^2}{[1 - \frac{1}{n}\text{tr}(\mathbf{S}(\lambda))]^2}$$

其中 $\mathbf{S}(\lambda) = \mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'$

6.3 LASSO 回归

6.3.1 LASSO 的定义与特性

L1 正则化项：

$$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \right\}$$

约束形式：

$$\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2$$
$$\text{s.t.} \quad ||\boldsymbol{\beta}||_1 \leq t$$

6.3.2 LASSO 的变量选择性质

软阈值：在正交设计情况下，LASSO 解有显式表达式：

$$\hat{\beta}_j^{\text{lasso}} = \text{sign}(\hat{\beta}_j^{\text{OLS}})(|\hat{\beta}_j^{\text{OLS}}| - \lambda)_+$$

变量选择一致性：在适当条件下，LASSO 能以概率 1 选择出真实模型。

6.3.3 求解算法

坐标下降法：对每个 j，固定其他系数，更新：

$$\beta_j \leftarrow S\left(\frac{1}{n} \sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda \right)$$

其中 $S(z, \lambda) = \text{sign}(z)(|z| - \lambda)_+$ 是软阈值函数。

LARS 算法：通过分段线性路径高效计算整个解路径。

6.4 弹性网

6.5 正则化路径比较

弹性网：结合岭回归和 LASSO 的优点：

$$\hat{\boldsymbol{\beta}}_{\text{enet}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2n} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda[\alpha||\boldsymbol{\beta}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2] \right\}$$

方法比较：

方法	变量选择	组效应	高维适用性
岭回归	否	是	是
LASSO	是	否	是
弹性网	是	是	是
PCR	否	是	是
PLS	否	是	是

调参策略

网格搜索：在 λ（和 α）的网格上计算交叉验证误差。

信息准则：

- AIC：$AIC = n\ln(\hat{\sigma}^2) + 2k$
- BIC：$BIC = n\ln(\hat{\sigma}^2) + k\ln(n)$

一倍标准误准则：选择调优参数，使得交叉验证误差在最小误差的一倍标准误范围内。

6.6 案例分析

本章总结

核心公式回顾

1. 岭回归：$\hat{\boldsymbol{\beta}}_{\text{ridge}} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$
2. LASSO：$\hat{\boldsymbol{\beta}}_{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \{\frac{1}{2n}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_1\}$
3. 弹性网：$P_\alpha(\boldsymbol{\beta}) = \alpha||\boldsymbol{\beta}||_1 + \frac{1}{2}(1-\alpha)||\boldsymbol{\beta}||_2^2$

方法选择指南

问题场景	推荐方法	理由
多重共线性严重	岭回归、PCR	稳定估计，降低方差
变量选择需求	LASSO、弹性网	自动特征选择
预测精度优先	弹性网、PLS	平衡偏差与方差
可解释性重要	LASSO	稀疏解，易于解释
计算效率要求	坐标下降	高效处理高维问题

实践建议

1. 数据预处理：标准化自变量，确保惩罚的公平性
2. 模型评估：使用交叉验证，避免过拟合
3. 结果验证：在测试集上评估预测性能
4. 稳健性检查：尝试不同方法，比较结果一致性
5. 领域知识：结合实际问题背景选择合适方法

高维数据分析是现代统计学习的重要组成部分。掌握正则化和降维技术，能够帮助您在面对复杂数据时建立更加稳健和可解释的预测模型。

7 广义线性模型

本章导读

在实际应用中，响应变量并不总是连续型的。当面临分类问题或计数数据时，传统线性回归模型的局限性凸显。本章将系统介绍广义线性模型的统一框架，重点讲解逻辑斯谛回归、Probit 回归和线性判别分析等经典分类方法，为您提供处理分类问题的完整理论体系和实用工具。

7.1 虚拟变量

7.1.1 虚拟变量的本质与构建

虚拟变量（Dummy Variables），又称指示变量（Indicator Variables），是将分类自变量转化为数值变量的基本技术。其核心思想是用二进制编码表示类别成员身份。

7.1.2 虚拟变量的构建原则

对于一个具有 k 个类别的分类变量，通常需要 $k-1$ 个虚拟变量，以避免虚拟变量陷阱（完全多重共线性）。这种编码方式称为参照组编码或基线编码。

$$D_1 = \begin{cases} 1 & \text{如果观测属于类别 1} \\ 0 & \text{否则} \end{cases}, \quad D_2 = \begin{cases} 1 & \text{如果观测属于类别 2} \\ 0 & \text{否则} \end{cases}, \quad \cdots, \quad D_{k-1} = \begin{cases} 1 & \text{如果观测属于类别 } k \\ 0 & \text{否则} \end{cases}$$

第 k 个类别作为参照组，当所有 $D_i = 0$ 时表示属于参照组。

7.1.3 包含虚拟变量的回归模型

考虑一个简单的回归模型，包含一个连续自变量 X 和一个具有三个类别的分类变量 C：

$$Y = \beta_0 + \beta_1 X + \beta_2 D_1 + \beta_3 D_2 + \beta_4 (X \times D_1) + \beta_5 (X \times D_2) + \varepsilon$$

其中：- β_0：参照组的截距 - β_1：参照组中 X 对 Y 的效应 - β_2, β_3：类别 1 和类别 2 相对于参照组的截距差异 - β_4, β_5：类别 1 和类别 2 中 X 效应的差异（交互作用）

7.1.4 虚拟变量的统计检验

- 单个虚拟变量的 t 检验：检验特定类别与参照组是否存在显著差异
- 联合 F 检验：检验所有虚拟变量（即整个分类变量）是否显著

$$F = \frac{(SSR_{\text{完整}} - SSR_{\text{简化}})/(k-1)}{SSR_{\text{完整}}/(n-k-p)}$$

其中 p 为其他自变量的数量

多项式虚拟变量

对于有序分类变量，可以检验线性、二次等趋势：

$$Y = \beta_0 + \beta_1 X + \gamma_1 L + \gamma_2 Q + \varepsilon$$

其中 L 和 Q 分别表示线性项和二次项编码。

虚拟变量与结构变化检验

虚拟变量可用于检验不同子样本（如不同时期、不同群体）的模型结构是否相同，这是 Chow 检验的基础。

7.2 广义线性模型理论

7.2.1 指数族分布

指数族的概率密度函数：

$$f(y|\theta, \phi) = \exp\left\{ \frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi) \right\}$$

其中：

- θ：自然参数（典则参数）
- ϕ：离散参数
- $a(\cdot), b(\cdot), c(\cdot)$ 已知函数

常见分布的指数族形式：

分布	θ	ϕ	$b(\theta)$	$a(\phi)$
正态	μ	σ^2	$\frac{\theta^2}{2}$	ϕ
泊松	$\ln \mu$	1	e^θ	1
二项	$\ln\left(\frac{\pi}{1-\pi}\right)$	1	$\ln(1+e^\theta)$	1
Gamma	$-\frac{1}{\mu}$	ν	$-\ln(-\theta)$	$\frac{1}{\nu}$

7.2.2 GLM 的三组成部分

随机成分：响应变量 Y 来自指数族分布：

$$Y \sim f(y|\theta, \phi)$$

系统成分：线性预测器：

$$\eta = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = \mathbf{X}\boldsymbol{\beta}$$

连接函数：连接随机成分和系统成分：

$$g(\mu) = \eta$$

其中 $\mu = E(Y) = b'(\theta)$

7.2.3 典则连接函数

定义：当 $g(\mu) = \theta$ 时，称为典则连接函数。

性质：- 简化计算和理论推导 - 保证 $\mathbf{X}'\mathbf{y}$ 是充分统计量 - 在多数情况下具有良好的统计性质

7.3 逻辑斯谛回归

7.3.1 二项分布的 GLM 框架

伯努利分布：

$$P(Y=1)=\pi,\quad P(Y=0)=1-\pi$$

典则连接函数（logit 函数）：

$$\eta=\ln\left(\frac{\pi}{1-\pi}\right)=\mathbf{X}\boldsymbol{\beta}$$

逆连接函数（sigmoid 函数）：

$$\pi=\frac{e^{\mathbf{X}\boldsymbol{\beta}}}{1+e^{\mathbf{X}\boldsymbol{\beta}}}=\frac{1}{1+e^{-\mathbf{X}\boldsymbol{\beta}}}$$

7.3.2 参数估计：极大似然法

似然函数：

$$L(\boldsymbol{\beta})=\prod_{i=1}^{n}\pi_i^{y_i}(1-\pi_i)^{1-y_i}$$

对数似然函数：

$$\ell(\boldsymbol{\beta})=\sum_{i=1}^{n}\left[y_i\ln\pi_i+(1-y_i)\ln(1-\pi_i)\right]$$

得分函数：

$$\frac{\partial\ell}{\partial\boldsymbol{\beta}}=\sum_{i=1}^{n}(y_i-\pi_i)\mathbf{x}_i=\mathbf{0}$$

7.3.3 迭代加权最小二乘算法

权重矩阵：

$$\mathbf{W}=\text{diag}\left[\pi_i(1-\pi_i)\right]$$

工作响应变量：

$$z_i=\mathbf{x}_i'\boldsymbol{\beta}+\frac{y_i-\pi_i}{\pi_i(1-\pi_i)}$$

迭代更新：

$$\boldsymbol{\beta}^{(t+1)}=(\mathbf{X}'\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{(t)}\mathbf{z}^{(t)}$$

7.3.4 统计推断与解释

优势比解释：对于二值自变量 X_j：

$$\text{OR}_j = e^{\beta_j} = \frac{\pi|X_j = 1/(1-\pi|X_j = 1)}{\pi|X_j = 0/(1-\pi|X_j = 0)}$$

Wald 检验：

$$z_j = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \sim N(0,1)$$

似然比检验：

$$G = 2[\ell(\hat{\boldsymbol{\beta}}) - \ell(\hat{\boldsymbol{\beta}}_0)] \sim \chi^2(p - p_0)$$

7.3.5 多项逻辑斯谛回归

基线类别 Logit 模型

对于 J 个类别的响应变量，以第 J 类为参照：

$$\log\left(\frac{P(Y_i = j|\mathbf{x}_i)}{P(Y_i = J|\mathbf{x}_i)}\right) = \beta_{j0} + \beta_{j1}x_{i1} + \cdots + \beta_{jp}x_{ip}, \quad j = 1, \ldots, J-1$$

类别概率计算

$$P(Y_i = j|\mathbf{x}_i) = \frac{\exp(\eta_{ij})}{1 + \sum_{m=1}^{J-1}\exp(\eta_{im})}, \quad j = 1, \ldots, J-1$$

$$P(Y_i = J|\mathbf{x}_i) = \frac{1}{1 + \sum_{m=1}^{J-1}\exp(\eta_{im})}$$

其中 $\eta_{ij} = \beta_{j0} + \beta_{j1}x_{i1} + \cdots + \beta_{jp}x_{ip}$。

7.3.6 有序逻辑斯谛回归

对于有序响应变量，使用比例优势模型：

$$\log\left(\frac{P(Y_i \le j|\mathbf{x}_i)}{P(Y_i > j|\mathbf{x}_i)}\right) = \alpha_j - (\beta_1 x_{i1} + \cdots + \beta_p x_{ip}), \quad j = 1, \ldots, J-1$$

其中 $\alpha_1 < \alpha_2 < \cdots < \alpha_{J-1}$ 为切点参数，负号使得 β 的正值表示更可能属于更高类别。

7.4 Probit 回归

7.4.1 Probit 模型设定

Probit 连接函数：

$$\Phi^{-1}(\pi) = \mathbf{X}\boldsymbol{\beta}$$

其中 $\Phi(\cdot)$ 是标准正态分布的累积分布函数。

概率形式：

$$\pi = \Phi(\mathbf{X}\boldsymbol{\beta})$$

7.4.2 潜在变量解释

潜变量模型：假设存在连续潜变量 Y^*：

$$Y^* = \mathbf{X}\boldsymbol{\beta} + \varepsilon, \quad \varepsilon \sim N(0,1)$$

观测响应：

$$Y = \begin{cases} 1 & \text{if } Y^* > 0 \\ 0 & \text{otherwise} \end{cases}$$

概率推导：

$$P(Y=1|\mathbf{X}) = P(Y^* > 0) = P(\varepsilon > -\mathbf{X}\boldsymbol{\beta}) = \Phi(\mathbf{X}\boldsymbol{\beta})$$

7.4.3 与 Logit 模型的比较

连接函数形状：- Logit：$g(p) = \ln\left(\frac{p}{1-p}\right)$ - Probit：$g(p) = \Phi^{-1}(p)$

尾部行为：- Logit 函数有更重的尾部 - 在实践中两者通常给出相似的结果

系数转换：

$$\beta_{\text{logit}} \approx 1.6 \times \beta_{\text{probit}}$$

7.5 模型比较与选择

7.5.1 分类性能评估

混淆矩阵：

	预测正类	预测负类
实际正类	TP	FN
实际负类	FP	TN

常用指标：- 准确率：$\frac{TP+TN}{n}$ - 精确率：$\frac{TP}{TP+FP}$ - 召回率：$\frac{TP}{TP+FN}$ - F1 分数：$\frac{2\times\text{精确率}\times\text{召回率}}{\text{精确率}+\text{召回率}}$

7.5.2 ROC 曲线与 AUC

ROC 曲线：以假正率为横轴，真正率为纵轴的曲线。

AUC 解释：- AUC = 0.5：随机猜测 - AUC = 1.0：完美分类 - AUC > 0.8：通常认为模型表现良好

7.5.3 方法比较

方法	类型	假设	优点	局限性
逻辑回归	判别式	线性决策边界	概率输出，可解释性强	对特征相关性敏感
Probit 回归	判别式	线性决策边界	有潜在变量解释	计算稍复杂
LDA	生成式	正态分布，等协方差	小样本表现好，多分类自然	假设较强

7.5.4 模型诊断

离群值检测：- Pearson 残差：$r_i = \frac{y_i-\hat{\pi}_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)}}$ - Deviance 残差：$d_i = \text{sign}(y_i-\hat{\pi}_i)\sqrt{-2[y_i \ln \hat{\pi}_i + (1-y_i)\ln(1-\hat{\pi}_i)]}$

拟合优度检验：- Hosmer-Lemeshow 检验 - 皮尔逊卡方检验

过度离散检验：

$$\frac{\sum_{i=1}^{n}(y_i-\hat{\pi}_i)^2}{\hat{\pi}_i(1-\hat{\pi}_i)} \sim \chi^2(n-p-1)$$

7.6 案例分析

本章总结

核心公式回顾

1. GLM 连接函数：$g(\mu) = \mathbf{X}\boldsymbol{\beta}$
2. Logit 模型：$\ln\left(\frac{\pi}{1-\pi}\right) = \mathbf{X}\boldsymbol{\beta}$
3. Probit 模型：$\Phi^{-1}(\pi) = \mathbf{X}\boldsymbol{\beta}$
4. LDA 判别函数：$\delta_k(\mathbf{x}) = \mathbf{x}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \pi_k$
5. IWLS 更新：$\boldsymbol{\beta}^{(t+1)} = (\mathbf{X}'\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{(t)}\mathbf{z}^{(t)}$

方法选择指南

问题场景	推荐方法	理由
需要概率输出	逻辑回归、Probit 回归	直接建模条件概率
小样本问题	LDA	参数更少，估计更稳定
多分类问题	LDA、多项逻辑回归	天然处理多类别
可解释性重要	逻辑回归	优势比有明确解释
理论一致性	Probit 回归	有潜在变量理论基础

实践建议

1. 数据预处理：检查类别平衡性，必要时重采样
2. 特征工程：考虑交互项和非线性变换
3. 模型验证：使用交叉验证评估泛化能力
4. 结果解释：结合领域知识理解系数含义
5. 稳健性分析：尝试不同方法，比较结果一致性

广义线性模型为处理非正态响应变量提供了统一的框架，而线性分类方法则是机器学习中最基础且重要的工具。掌握这些方法为进一步学习更复杂的非线性模型奠定了坚实基础。

8 线性分类模型

本章导读

把线性分析进行扩展

8.1 线性判别分析与二次判别分析

判别分析（Discriminant Analysis）是一种经典的统计分类方法，主要用于解决分类问题和降维问题。其核心思想是寻找能够最佳区分不同类别的特征组合。

假设我们有：- K 个类别：$C_1, C_2, \ldots, C_K$ - p 维特征向量：$\mathbf{x} = (x_1, x_2, \ldots, x_p)^\top$ - 训练数据集：$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，其中 $y_i \in \{1, 2, \ldots, K\}$

判别分析的目标是基于贝叶斯定理构建分类规则：

$$P(Y = k|\mathbf{x}) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{l=1}^K \pi_l f_l(\mathbf{x})}$$

其中：- $\pi_k = P(Y = k)$：类别 k 的先验概率 - $f_k(\mathbf{x})$：类别 k 中 $\mathbf{x}$ 的概率密度函数

基本假设

LDA 基于以下关键假设：

1. 多元正态性：每个类别的特征向量服从多元正态分布

$$\mathbf{x}|Y = k \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

2. 同方差性：所有类别共享相同的协方差矩阵 $\boldsymbol{\Sigma}$

3. 协方差矩阵非奇异：$\boldsymbol{\Sigma}$ 是可逆的

判别函数推导

根据多元正态分布的概率密度函数：

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

代入贝叶斯公式，忽略常数项，得到判别函数：

$$\begin{aligned}\delta_k(\mathbf{x}) &= \log(\pi_k) + \log(f_k(\mathbf{x})) \\ &= \log(\pi_k) - \frac{1}{2}\boldsymbol{\mu}_k^\top \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k + \mathbf{x}^\top \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\mathbf{x}^\top \mathbf{\Sigma}^{-1}\mathbf{x}\end{aligned}$$

由于 $\mathbf{x}^\top \mathbf{\Sigma}^{-1}\mathbf{x}$ 与 k 无关，可以简化为线性判别函数：

$$\delta_k(\mathbf{x}) = \mathbf{x}^\top \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^\top \mathbf{\Sigma}^{-1}\boldsymbol{\mu}_k + \log(\pi_k)$$

分类规则

将观测 $\mathbf{x}$ 分配到使判别函数最大的类别：

$$\hat{y}(\mathbf{x}) = \arg\max_{k=1,\ldots,K} \delta_k(\mathbf{x})$$

决策边界是线性的，因为判别函数是 $\mathbf{x}$ 的线性函数。

参数估计

从训练数据中估计参数：

1. 先验概率：
$$\hat{\pi}_k = \frac{n_k}{n}, \quad n_k = \sum_{i=1}^{n} I(y_i = k)$$

2. 均值向量：
$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i:y_i=k} \mathbf{x}_i$$

3. 协方差矩阵：
$$\hat{\mathbf{\Sigma}} = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

   这是合并协方差矩阵（pooled covariance matrix）。

8.1.2 LDA 的几何解释

马氏距离

LDA 的分类规则等价于将 **x** 分配到具有最小马氏距离（Mahalanobis distance）的类别：

$$D_k^2(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)$$

考虑先验概率调整后的距离：

$$D_k^2(\mathbf{x}) - 2\log(\pi_k)$$

降维视角

LDA 可以视为寻找最佳投影方向以最大化类间散度与类内散度的比值：

$$J(\mathbf{w}) = \frac{\mathbf{w}^\top \mathbf{S}_B \mathbf{w}}{\mathbf{w}^\top \mathbf{S}_W \mathbf{w}}$$

其中：- 类间散度矩阵：$\mathbf{S}_B = \sum_{k=1}^{K} n_k (\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})(\boldsymbol{\mu}_k - \bar{\boldsymbol{\mu}})^\top$ - 类内散度矩阵：$\mathbf{S}_W = \sum_{k=1}^{K} \sum_{i:y_i=k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top$ - 总体均值：$\bar{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$

最优投影方向是 $\mathbf{S}_W^{-1}\mathbf{S}_B$ 的特征向量，最多有 $\min(p, K-1)$ 个有效判别方向。

8.2 二次判别分析（QDA）

基本假设

QDA 放宽了 LDA 的同方差性假设：

1. 多元正态性：每个类别的特征向量服从多元正态分布

$$\mathbf{x} | Y = k \sim N_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

2. 异方差性：每个类别有自己的协方差矩阵 $\boldsymbol{\Sigma}_k$

判别函数推导

对于多元正态分布，类别 k 的概率密度函数为：

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

取对数并忽略常数项，得到二次判别函数：

$$\begin{aligned}\delta_k(\mathbf{x}) = \log(\pi_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| \\ - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\end{aligned}$$

分类规则

同样，将观测 $\mathbf{x}$ 分配到使判别函数最大的类别：

$$\hat{y}(\mathbf{x}) = \arg\max_{k=1,\dots,K} \delta_k(\mathbf{x})$$

由于判别函数包含 $\mathbf{x}$ 的二次项，决策边界是二次曲面（椭圆、双曲线或抛物线）。

参数估计

从训练数据中估计参数：

1. 先验概率：
$$\hat{\pi}_k = \frac{n_k}{n}$$
2. 均值向量：
$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i:y_i=k} \mathbf{x}_i$$
3. 协方差矩阵：
$$\hat{\mathbf{\Sigma}}_k = \frac{1}{n_k - 1}\sum_{i:y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top$$

QDA 的二次项展开

将 QDA 的判别函数展开，可以更清楚地看到其二次本质：

$$\begin{aligned}\delta_k(\mathbf{x}) = \log(\pi_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}\boldsymbol{\mu}_k^\top \mathbf{\Sigma}_k^{-1}\boldsymbol{\mu}_k \\ + \mathbf{x}^\top \mathbf{\Sigma}_k^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\mathbf{x}^\top \mathbf{\Sigma}_k^{-1}\mathbf{x}\end{aligned}$$

令：- $C_k = \log(\pi_k) - \frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}\boldsymbol{\mu}_k^\top \mathbf{\Sigma}_k^{-1}\boldsymbol{\mu}_k$（常数项）- $\boldsymbol{\beta}_k = \mathbf{\Sigma}_k^{-1}\boldsymbol{\mu}_k$（线性系数）- $\mathbf{\Omega}_k = -\frac{1}{2}\mathbf{\Sigma}_k^{-1}$（二次系数）

则判别函数可写为：

$$\delta_k(\mathbf{x}) = C_k + \mathbf{x}^\top \boldsymbol{\beta}_k + \mathbf{x}^\top \mathbf{\Omega}_k \mathbf{x}$$

8.2.2 LDA 与 QDA 的比较

理论比较

特性	LDA	QDA
协方差矩阵	$\mathbf{\Sigma}_k = \mathbf{\Sigma}$（相同）	$\mathbf{\Sigma}_k$（不同）
判别函数	线性函数	二次函数
决策边界	线性超平面	二次曲面
参数数量	$Kp + p(p+1)/2 + (K-1)$	$Kp + K \cdot p(p+1)/2 + (K-1)$
假设强度	较强	较弱

参数复杂度分析

LDA 的参数数量：- 均值向量：$K \times p$ 个参数 - 协方差矩阵：$p(p+1)/2$ 个参数 - 先验概率：$K-1$ 个参数（因为 $\sum_{k=1}^{K} \pi_k = 1$）

总计：$Kp + \frac{p(p+1)}{2} + (K-1)$

QDA 的参数数量：- 均值向量：$K \times p$ 个参数 - 协方差矩阵：$K \times p(p+1)/2$ 个参数 - 先验概率：$K-1$ 个参数

总计：$Kp + K \cdot \frac{p(p+1)}{2} + (K-1)$

偏差-方差权衡

LDA 的优势：1. 参数更少，方差更小 2. 在样本量较小时更稳定 3. 当同方差性假设成立时，分类效果更优

QDA 的优势：1. 更灵活，可以捕捉类别间的协方差差异 2. 当异方差性明显时，分类精度更高 3. 对模型假设的依赖较小

样本量要求

QDA 需要更大的样本量来准确估计每个类别的协方差矩阵。经验法则：- LDA：每个类别至少需要 $p+1$ 个样本 - QDA：每个类别至少需要 $p(p+3)/2+1$ 个样本

8.2.4 正则化判别分析（RDA）

RDA 的基本思想

RDA 是 LDA 和 QDA 的折中方案，通过正则化协方差矩阵来平衡偏差和方差：

$$\hat{\mathbf{\Sigma}}_k(\alpha,\gamma)=\alpha\hat{\mathbf{\Sigma}}_k+(1-\alpha)\hat{\mathbf{\Sigma}}$$

其中：- $\hat{\mathbf{\Sigma}}_k$：第 k 类的样本协方差矩阵 - $\hat{\mathbf{\Sigma}}$：合并协方差矩阵 - $\alpha\in[0,1]$：控制个体协方差与合并协方差的混合比例

进一步进行特征值收缩：

$$\hat{\mathbf{\Sigma}}_k(\alpha,\gamma)=\gamma\hat{\mathbf{\Sigma}}_k(\alpha)+(1-\gamma)\frac{\text{tr}(\hat{\mathbf{\Sigma}}_k(\alpha))}{p}\mathbf{I}_p$$

其中 $\gamma\in[0,1]$ 控制收缩强度。

RDA 的参数选择

通过交叉验证选择最优的 (α,γ) 组合：

1. 在训练集上拟合不同 (α,γ) 组合的 RDA 模型
2. 在验证集上评估分类准确率
3. 选择使验证集准确率最高的参数组合

与逻辑回归的比较

方面	LDA/QDA	逻辑回归
假设	特征服从多元正态分布	无分布假设
建模对象	联合分布 $P(X,Y)$	条件分布 $P(Y\|X)$
参数估计	有解析解	迭代最大似然估计
样本效率	当假设成立时更高效	更稳健
多分类	直接扩展	需要多项逻辑回归

8.3 朴素贝叶斯分类器

朴素贝叶斯基本思想

朴素贝叶斯（Naïve Bayes）是一种基于贝叶斯定理的简单而高效的分类算法。其” 朴素”（naïve）之处在于假设特征之间相互条件独立，即给定类别时，各个特征之间没有依赖关系。尽管这一假设在现实中很少完全成立，但朴素贝叶斯在许多实际应用中仍表现出色。

贝叶斯定理是朴素贝叶斯分类器的理论基础：

$$P(Y = k|\mathbf{x}) = \frac{P(Y = k) \cdot P(\mathbf{x}|Y = k)}{P(\mathbf{x})}$$

其中：- $P(Y = k|\mathbf{x})$：后验概率（给定特征 **x** 时属于类别 k 的概率）- $P(Y = k)$：先验概率（类别 k 在训练集中的比例）- $P(\mathbf{x}|Y = k)$：似然（类别 k 中观察到特征 **x** 的概率）- $P(\mathbf{x})$：证据（边际概率，作为归一化常数）

朴素贝叶斯假设

对于 p 维特征向量 $\mathbf{x} = (x_1, x_2, \dots, x_p)^\top$，朴素贝叶斯假设：

$$P(\mathbf{x}|Y = k) = P(x_1, x_2, \dots, x_p|Y = k) = \prod_{j=1}^{p} P(x_j|Y = k)$$

即给定类别 k 时，各个特征 $x_1, x_2, \dots, x_p$ 相互条件独立。

分类规则

将观测 **x** 分配到后验概率最大的类别：

$$\hat{y}(\mathbf{x}) = \arg \max_{k=1,\dots,K} P(Y = k|\mathbf{x})$$

根据贝叶斯定理和朴素假设：

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \arg\max_k P(Y = k) \cdot P(\mathbf{x}|Y = k) \\ &= \arg\max_k P(Y = k) \cdot \prod_{j=1}^{p} P(x_j|Y = k) \end{aligned}$$

由于 $P(\mathbf{x})$ 对所有类别相同，可以省略。

对数形式

在实际计算中，使用对数避免数值下溢：

$$\hat{y}(\mathbf{x}) = \arg\max_k \left[\log P(Y=k) + \sum_{j=1}^{p} \log P(x_j|Y=k) \right]$$

8.3.2 不同特征类型的概率估计

朴素贝叶斯可以根据特征类型选择不同的概率估计方法。

分类特征（伯努利朴素贝叶斯）

对于二值特征或类别特征，使用伯努利分布：

$$P(x_j = v|Y=k) = \theta_{kjv}$$

其中 $v \in \{0,1\}$ 或 $v \in \{1,2,\dots,m_j\}$。

参数估计

最大似然估计：

$$\hat{\theta}_{kjv} = \frac{N_{kjv} + \alpha}{N_k + \alpha m_j}$$

其中：- N_{kjv}：类别 k 中特征 j 取值为 v 的样本数 - N_k：类别 k 的总样本数 - α：平滑参数（拉普拉斯平滑）- m_j：特征 j 的可能取值数

拉普拉斯平滑

当某个特征值在训练集中未出现时，最大似然估计为 0，导致整个概率为 0。拉普拉斯平滑（Laplace smoothing）或加一平滑解决此问题：

$$\hat{\theta}_{kjv} = \frac{N_{kjv} + 1}{N_k + m_j}$$

这是 $\alpha = 1$ 时的特例。

连续特征（高斯朴素贝叶斯）

对于连续特征，假设服从高斯分布：

$$P(x_j|Y=k) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} \exp\left(-\frac{(x_j-\mu_{kj})^2}{2\sigma_{kj}^2}\right)$$

参数估计

$$\hat{\mu}_{kj} = \frac{1}{N_k} \sum_{i:y_i=k} x_{ij}$$

$$\hat{\sigma}_{kj}^2 = \frac{1}{N_k} \sum_{i:y_i=k} (x_{ij}-\hat{\mu}_{kj})^2$$

对数似然

对于高斯朴素贝叶斯，对数似然为：

$$\log P(x_j|Y=k) = -\frac{1}{2}\log(2\pi\sigma_{kj}^2) - \frac{(x_j-\mu_{kj})^2}{2\sigma_{kj}^2}$$

（多项式朴素贝叶斯）

对于表示频次或计数的特征，使用多项式分布：

$$P(\mathbf{x}|Y=k) = \frac{(\sum_{j=1}^p x_j)!}{\prod_{j=1}^p x_j!} \prod_{j=1}^p \theta_{kj}^{x_j}$$

参数估计

$$\hat{\theta}_{kj} = \frac{N_{kj}+\alpha}{\sum_{j=1}^p N_{kj} + \alpha p}$$

其中 $N_{kj} = \sum_{i:y_i=k} x_{ij}$ 是类别 k 中特征 j 的总计数。

混合特征类型

当数据包含不同类型特征时，可以组合不同分布：

$$P(\mathbf{x}|Y=k) = \prod_{j\in C} P_{\text{cat}}(x_j|Y=k) \cdot \prod_{j\in R} P_{\text{cont}}(x_j|Y=k) \cdot \prod_{j\in N} P_{\text{count}}(x_j|Y=k)$$

其中 C、R、N 分别表示分类、连续、计数特征的索引集。

8.3.3 算法实现

训练阶段

输入：训练集 $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$，其中 $\mathbf{x}_i \in \mathbb{R}^p$，$y_i \in \{1, 2, \ldots, K\}$

步骤：1. 估计先验概率：$\hat{\pi}_k = \frac{N_k + \alpha}{n + \alpha K}$ 2. 对于每个类别 k 和每个特征 j：- 如果特征 j 是分类的：估计 $\hat{\theta}_{kjv}$ 对于所有 v - 如果特征 j 是连续的：估计 $\hat{\mu}_{kj}$ 和 $\hat{\sigma}^2_{kj}$ - 如果特征 j 是计数的：估计 $\hat{\theta}_{kj}$

输出：所有估计参数

预测阶段

输入：新样本 $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_p^*)$，训练得到的参数

步骤：1. 对于每个类别 $k = 1, \ldots, K$：

$$\text{score}_k = \log \hat{\pi}_k + \sum_{j=1}^{p} \log P(x_j^* | Y = k; \hat{\theta}_{kj})$$

2. 预测类别：$\hat{y} = \arg\max_k \text{score}_k$ 3. （可选）计算后验概率：

$$P(Y = k | \mathbf{x}^*) = \frac{\exp(\text{score}_k)}{\sum_{l=1}^{K} \exp(\text{score}_l)}$$

输出：预测类别 $\hat{y}$ 和后验概率

8.3.4 与判别分析的比较

与 LDA/QDA 的关系

朴素贝叶斯与判别分析都基于贝叶斯定理，但有重要区别：

方面	朴素贝叶斯	LDA/QDA
特征独立性	条件独立假设	考虑特征相关性
分布假设	可灵活选择	必须多元正态
参数数量	$O(Kp)$	LDA: $O(p^2)$, QDA: $O(Kp^2)$
决策边界	可能是非线性	LDA 线性，QDA 二次

等价性条件

当以下条件成立时，高斯朴素贝叶斯等价于 LDA：1. 所有特征服从多元正态分布 2. 特征条件独立（协方差矩阵为对角矩阵）3. 各类别的对角协方差矩阵相等

在这种情况下，朴素贝叶斯的决策边界也是线性的。

偏差-方差权衡

朴素贝叶斯引入条件独立假设：- 增加偏差：当特征相关时，模型假设错误 - 减少方差：参数更少，估计更稳定

在高维小样本情况下，朴素贝叶斯通常优于更复杂的模型。

模型变体

伯努利朴素贝叶斯

适用于二值特征，如文本分类中的词袋模型（出现/不出现）。

概率模型：

$$P(x_j|Y=k) = \theta_{kj}^{x_j}(1-\theta_{kj})^{1-x_j}$$

多项式朴素贝叶斯

适用于计数数据，如文本分类中的词频。

概率模型：

$$P(\mathbf{x}|Y=k) = \frac{(\sum_j x_j)!}{\prod_j x_j!}\prod_j \theta_{kj}^{x_j}$$

高斯朴素贝叶斯

适用于连续特征，假设正态分布。

补充朴素贝叶斯

处理不平衡数据，调整先验概率：

$$\pi_k^{\text{complement}} = \frac{\sum_{i \notin \text{class } k}\sum_j x_{ij}}{\text{总计数}}$$

贝叶斯信念网络

放宽条件独立假设，允许部分特征相关：

$$P(\mathbf{x}|Y=k)=\prod_{j=1}^{p}P(x_j|\text{Pa}(x_j),Y=k)$$

其中 $\text{Pa}(x_j)$ 是 x_j 的父节点集合。

参数估计与正则化

最大似然估计

对于参数 θ，最大似然估计为：

$$\hat{\theta}_{\text{MLE}}=\arg\max_{\theta}\prod_{i=1}^{n}P(\mathbf{x}_i,y_i|\theta)$$

最大后验估计

考虑参数先验，最大后验估计：

$$\hat{\theta}_{\text{MAP}}=\arg\max_{\theta}P(\theta)\prod_{i=1}^{n}P(\mathbf{x}_i,y_i|\theta)$$

共轭先验

分布	共轭先验	后验分布
伯努利	Beta 分布	Beta 分布
多项式	Dirichlet 分布	Dirichlet 分布
高斯（已知方差）	高斯分布	高斯分布
高斯（已知均值）	Gamma 分布	Gamma 分布

经验贝叶斯

从数据中估计先验分布的超参数：

$$\hat{\alpha}=\arg\max_{\alpha}\int P(D|\theta)P(\theta|\alpha)d\theta$$

实际应用

文本分类

朴素贝叶斯是文本分类的经典算法，如：- 垃圾邮件检测 - 情感分析 - 新闻分类

特征工程：- 词袋模型（Bag-of-Words）- TF-IDF 加权 - N-gram 特征

推荐系统

基于内容的推荐：

$$P(\text{喜欢}|\text{物品特征}) \propto P(\text{喜欢}) \prod_j P(\text{特征}_j|\text{喜欢})$$

医学诊断

基于症状预测疾病：

$$P(\text{疾病}|\text{症状}) \propto P(\text{疾病}) \prod_j P(\text{症状}_j|\text{疾病})$$

异常检测

建模正常行为，检测低概率事件：

$$P(\mathbf{x}) = \sum_{k=1}^{K} P(Y=k) \prod_{j=1}^{p} P(x_j|Y=k)$$

将 $P(\mathbf{x}) < \tau$ 的样本标记为异常。

8.5 模型比较与选择

8.5.1 分类性能评估

混淆矩阵：

	预测正类	预测负类
实际正类	TP	FN
实际负类	FP	TN

常用指标：- 准确率：$\frac{TP+TN}{n}$ - 精确率：$\frac{TP}{TP+FP}$ - 召回率：$\frac{TP}{TP+FN}$ - F1 分数：$\frac{2\times\text{精确率}\times\text{召回率}}{\text{精确率}+\text{召回率}}$

8.5.2 ROC 曲线与 AUC

ROC 曲线：以假正率为横轴，真正率为纵轴的曲线。

AUC 解释：- AUC = 0.5：随机猜测 - AUC = 1.0：完美分类 - AUC > 0.8：通常认为模型表现良好

8.5.3 方法比较

方法	类型	假设	优点	局限性
逻辑回归	判别式	线性决策边界	概率输出，可解释性强	对特征相关性敏感
Probit 回归	判别式	线性决策边界	有潜在变量解释	计算稍复杂
LDA	生成式	正态分布，等协方差	小样本表现好，多分类自然	假设较强

8.5.4 模型诊断

离群值检测：- Pearson 残差：$r_i = \frac{y_i - \hat{\pi}_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)}}$ - Deviance 残差：$d_i = \text{sign}(y_i - \hat{\pi}_i)\sqrt{-2[y_i \ln \hat{\pi}_i + (1-y_i)\ln(1-\hat{\pi}_i)]}$

拟合优度检验：- Hosmer-Lemeshow 检验 - 皮尔逊卡方检验

过度离散检验：

$$\frac{\sum_{i=1}^{n}(y_i - \hat{\pi}_i)^2}{\hat{\pi}_i(1-\hat{\pi}_i)} \sim \chi^2(n-p-1)$$

8.6 案例分析

本章总结

核心公式回顾

1. GLM 连接函数：$g(\mu) = \mathbf{X}\boldsymbol{\beta}$
2. Logit 模型：$\ln\left(\frac{\pi}{1-\pi}\right) = \mathbf{X}\boldsymbol{\beta}$
3. Probit 模型：$\Phi^{-1}(\pi) = \mathbf{X}\boldsymbol{\beta}$
4. LDA 判别函数：$\delta_k(\mathbf{x}) = \mathbf{x}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \ln \pi_k$
5. IWLS 更新：$\boldsymbol{\beta}^{(t+1)} = (\mathbf{X}'\mathbf{W}^{(t)}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}^{(t)}\mathbf{z}^{(t)}$

方法选择指南

问题场景	推荐方法	理由
需要概率输出	逻辑回归、Probit 回归	直接建模条件概率

问题场景	推荐方法	理由
小样本问题	LDA	参数更少，估计更稳定
多分类问题	LDA、多项逻辑回归	天然处理多类别
可解释性重要	逻辑回归	优势比有明确解释
理论一致性	Probit 回归	有潜在变量理论基础

实践建议

1. 数据预处理：检查类别平衡性，必要时重采样
2. 特征工程：考虑交互项和非线性变换
3. 模型验证：使用交叉验证评估泛化能力
4. 结果解释：结合领域知识理解系数含义
5. 稳健性分析：尝试不同方法，比较结果一致性

广义线性模型为处理非正态响应变量提供了统一的框架，而线性分类方法则是机器学习中最基础且重要的工具。掌握这些方法为进一步学习更复杂的非线性模型奠定了坚实基础。

9 非线性回归与样条回归

本章导读

在实际科学研究与数据分析中，变量间的关系往往并非简单的直线形式。经济增长的 S 型轨迹、药物反应的剂量-效应曲线、温度对化学反应的加速效应——这些现象都揭示了一个基本事实：现实世界的关系本质上是非线性的。线性回归模型虽然简洁优雅，但当它面对这些复杂的曲线关系时，便会显得力不从心。

本章将带领读者从线性回归的舒适区走向非线性模型的广阔天地。我们将首先认识到线性模型的局限性，然后系统学习四种主流的非线性建模方法：可线性化的变换模型、灵活的多项式回归、高维的多元非线性模型，以及平滑灵活的样条回归。每种方法都有其独特的数学原理、适用场景和解释方式。

特别值得关注的是，非线性模型的参数估计不再有闭合解，我们需要借助数值优化方法。本章将深入浅出地介绍非线性最小二乘法和梯度下降法的原理与实现逻辑，帮助读者理解计算机是如何”学习” 这些复杂模型参数的。

通过本章的学习，读者将掌握从识别非线性模式、选择合适模型、估计模型参数到解释模型结果的全流程能力。非线性回归不仅是统计方法的扩展，更是认识复杂世界的有力工具。让我们开始这段探索曲线关系的旅程。

9.1 非线性的定义

线性回归模型假设因变量与自变量之间的关系是线性的：

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$$

然而在实际应用中，许多关系本质上是非线性的：- 经济增长的 S 型曲线 - 药物浓度与反应的饱和曲线 - 温度与化学反应速率的指数关系

定义：非线性回归模型是指参数相对于自变量是非线性的模型：

$$Y = f(\mathbf{X}, \boldsymbol{\beta}) + \varepsilon$$

其中 f 是关于参数 $\boldsymbol{\beta}$ 的非线性函数。

非线性模型的分类

类型	特点	示例
可线性化模型	通过变换可转为线性模型	$Y = \beta_0 e^{\beta_1 X}$
本质非线性模型	无法通过变换线性化	$Y = \beta_0 + e^{\beta_1 X}$
分段回归模型	不同区间用不同模型	样条回归、阈值模型

9.2 可线性化的非线性模型

指数模型

模型形式：

$$Y = \beta_0 e^{\beta_1 X} \varepsilon$$

其中 ε 为乘法误差项，通常假设 $\ln \varepsilon \sim N(0, \sigma^2)$。

线性化方法：取自然对数

$$\ln Y = \ln \beta_0 + \beta_1 X + \ln \varepsilon$$

令 $Y' = \ln Y$，$\beta_0' = \ln \beta_0$，$\varepsilon' = \ln \varepsilon$，得：

$$Y' = \beta_0' + \beta_1 X + \varepsilon'$$

幂函数模型

模型形式：

$$Y = \beta_0 X^{\beta_1} \varepsilon$$

线性化方法：两边取对数

$$\ln Y = \ln \beta_0 + \beta_1 \ln X + \ln \varepsilon$$

令 $Y' = \ln Y$，$X' = \ln X$，$\beta_0' = \ln \beta_0$，得：

$$Y' = \beta_0' + \beta_1 X' + \varepsilon'$$

对数模型

模型形式：

$$Y = \beta_0 + \beta_1 \ln X + \varepsilon$$

该模型直接是线性的形式，只需对自变量进行对数变换。

双曲线模型

模型形式：

$$\frac{1}{Y} = \beta_0 + \beta_1 \frac{1}{X} + \varepsilon$$

令 $Y' = 1/Y$，$X' = 1/X$，得线性形式：

$$Y' = \beta_0 + \beta_1 X' + \varepsilon$$

可线性化方法的局限

1. 误差结构的改变：变换可能改变误差项的分布假设
2. 可解释性：变换后参数的解释与原始模型不同
3. 预测的逆变换偏倚：对变换后的预测值进行逆变换可能产生系统偏倚

9.3 多项式回归

9.3.1 一元多项式回归

模型形式：

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p + \varepsilon$$

矩阵形式：

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

其中设计矩阵 $\mathbf{X} = [\mathbf{1}, \mathbf{X}, \mathbf{X}^2, \cdots, \mathbf{X}^p]$。

正交多项式：为减少多重共线性，可使用正交多项式：

$$Y = \alpha_0 + \alpha_1 \phi_1(X) + \alpha_2 \phi_2(X) + \cdots + \alpha_p \phi_p(X) + \varepsilon$$

其中 $\phi_j(X)$ 是 j 次正交多项式。

阶数选择

1. 逐步回归法：向前选择、向后删除
2. 信息准则：AIC、BIC

$$\text{AIC} = n\ln(\text{RSS}/n) + 2(p+1)$$

$$\text{BIC} = n\ln(\text{RSS}/n) + (p+1)\ln n$$

3. 交叉验证：k 折交叉验证的均方误差最小化

9.3.2 多元多项式回归

二次模型（含交互项）：

$$Y = \beta_0 + \sum_{i=1}^{k}\beta_i X_i + \sum_{i=1}^{k}\sum_{j\geq i}^{k}\beta_{ij}X_i X_j + \varepsilon$$

完整二次模型：

$$Y = \beta_0 + \sum_{i=1}^{k}\beta_i X_i + \sum_{i=1}^{k}\beta_{ii}X_i^2 + \sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\beta_{ij}X_i X_j + \varepsilon$$

多项式回归的注意事项

过拟合风险：高阶多项式可能过度拟合噪声外推不可靠：多项式在外推时可能产生不合理预测多重共线性：高次项之间高度相关

9.4 样条回归

9.4.1 分段多项式回归

将定义域划分为 $K+1$ 个区间，每个区间使用不同的多项式：

$$f(X) = \begin{cases} \beta_{00} + \beta_{10}X + \cdots + \beta_{p0}X^p & \text{if } X \leq \xi_1 \\ \beta_{01} + \beta_{11}X + \cdots + \beta_{p1}X^p & \text{if } \xi_1 < X \leq \xi_2 \\ \vdots & \vdots \\ \beta_{0K} + \beta_{1K}X + \cdots + \beta_{pK}X^p & \text{if } X > \xi_K \end{cases}$$

其中 $\xi_1, \xi_2, \ldots, \xi_K$ 为节点。

9.4.2 回归样条

为保证分段多项式在节点处光滑，添加连续性约束。m 次样条要求在节点处函数值及前 $m-1$ 阶导数连续。

三次样条（最常用）：在每个区间内为三次多项式，在节点处函数值、一阶导数、二阶导数连续。

基函数表示：三次样条可表示为：

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3$$

其中 $(X-\xi_k)_+^3 = \max(0, X-\xi_k)^3$。

9.4.3 自然样条

在边界节点外添加线性约束，使函数在边界区域更稳定：

$$f''(X) = 0 \quad \text{当} X \le \xi_1 \text{ 或} X \ge \xi_K$$

9.4.4 光滑样条

通过惩罚复杂度来寻找最优拟合函数：

$$\min_f \left\{ \sum_{i=1}^{n} [Y_i - f(X_i)]^2 + \lambda \int [f''(t)]^2 dt \right\}$$

其中 λ 为光滑参数，控制拟合优度与光滑度的权衡。

9.5 非线性回归的估计方法

9.5.1 非线性最小二乘法

目标函数：

$$\min_{\boldsymbol{\beta}} S(\boldsymbol{\beta}) = \sum_{i=1}^{n} [Y_i - f(\mathbf{X}_i, \boldsymbol{\beta})]^2$$

高斯-牛顿法：迭代求解，在每次迭代处对 f 进行一阶泰勒展开：

$$f(\mathbf{X}_i, \boldsymbol{\beta}) \approx f(\mathbf{X}_i, \boldsymbol{\beta}^{(t)}) + \mathbf{J}_i(\boldsymbol{\beta}^{(t)})(\boldsymbol{\beta} - \boldsymbol{\beta}^{(t)})$$

其中 $\mathbf{J}_i(\boldsymbol{\beta}) = \frac{\partial f(\mathbf{X}_i,\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$ 为雅可比矩阵。

迭代更新公式：

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + (\mathbf{J}^\top \mathbf{J})^{-1}\mathbf{J}^\top(\mathbf{Y} - \mathbf{f}(\boldsymbol{\beta}^{(t)}))$$

9.5.2 梯度下降法

基本思想：沿目标函数梯度反方向迭代更新参数。

更新公式：

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \eta \nabla S(\boldsymbol{\beta}^{(t)})$$

其中 η 为学习率，$\nabla S(\boldsymbol{\beta})$ 为梯度向量。

梯度计算：

$$\frac{\partial S}{\partial \beta_j} = -2\sum_{i=1}^{n}[Y_i - f(\mathbf{X}_i,\boldsymbol{\beta})]\frac{\partial f(\mathbf{X}_i,\boldsymbol{\beta})}{\partial \beta_j}$$

变体：批量梯度下降：使用全部样本计算梯度随机梯度下降：每次随机使用一个样本小批量梯度下降：每次使用一小批样本

9.5.3 牛顿法与拟牛顿法

牛顿法：

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \mathbf{H}^{-1}(\boldsymbol{\beta}^{(t)})\nabla S(\boldsymbol{\beta}^{(t)})$$

其中 $\mathbf{H}$ 为海森矩阵（二阶导数矩阵）。

拟牛顿法（BFGS）：用近似矩阵代替海森矩阵，减少计算量。

9.5.4 参数初始值与收敛准则

初始值选择：1. 基于物理意义或先验知识 2. 网格搜索法 3. 线性化模型的估计值

收敛准则：1. 参数变化量：$\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\| < \epsilon$ 2. 目标函数变化量：$|S(\boldsymbol{\beta}^{(t+1)}) - S(\boldsymbol{\beta}^{(t)})| < \epsilon$ 3. 梯度范数：$\|\nabla S(\boldsymbol{\beta}^{(t)})\| < \epsilon$

9.6 模型比较与诊断

9.6.1 拟合优度度量

残差平方和：

$$\text{RSS} = \sum_{i=1}^{n} [Y_i - \hat{f}(\mathbf{X}_i)]^2$$

调整后 R^2：

$$R^2_{\text{adj}} = 1 - \frac{\text{RSS}/(n-p)}{\text{TSS}/(n-1)}$$

信息准则：

$$\text{AIC} = n \ln(\text{RSS}/n) + 2p$$

$$\text{BIC} = n \ln(\text{RSS}/n) + p \ln n$$

9.6.2 残差分析

标准化残差：

$$r_i = \frac{Y_i - \hat{f}(\mathbf{X}_i)}{\hat{\sigma}}$$

诊断图形：

- 残差与拟合值图：检查方差齐性
- 残差与自变量图：检查模型设定
- 正态 Q-Q 图：检查正态性假设

9.7 变量变换方法

9.7.1 Box-Cox 变换

Box-Cox 变换通过对响应变量进行幂变换，使其更接近正态分布并稳定方差。

变换族定义：

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda} & \lambda \neq 0 \\ \ln y & \lambda = 0 \end{cases}$$

参数选择：通过最大化轮廓似然函数选择 λ：

$$L_{\max}(\lambda) = -\frac{n}{2}\ln(RSS(\lambda)/n) + (\lambda - 1)\sum_{i=1}^{n}\ln y_i$$

9.7.2 其他常用变换

对数变换：适用于右偏分布和方差随均值增大的情况：

$$y^* = \ln(y)$$

平方根变换：适用于泊松分布的计数数据：

$$y^* = \sqrt{y}$$

逻辑变换：适用于比例数据：

$$y^* = \ln\left(\frac{y}{1-y}\right)$$

9.8 案例分析

本章总结

非线性回归分析为我们打开了认识复杂世界关系的新窗口。通过本章的学习，我们掌握了从简单变换模型到复杂样条回归的完整方法体系，理解了如何根据数据特征和问题背景选择适当的非线性模型。

首先，对于可以通过数学变换线性化的模型（如指数、幂函数、对数模型），我们获得了简洁的解决方案。这类方法虽然实用，但需要警惕变换对误差结构和参数解释的影响。

多项式回归以其数学简洁性和高度灵活性成为非线性建模的基础工具。通过控制多项式阶数和使用正交多项式，我们可以在拟合能力与模型简洁性之间取得平衡。多元多项式回归进一步扩展了这一思想，能够捕捉变量间的交互效应和曲面关系。

样条回归代表了非线性建模的艺术高度。通过分段多项式和光滑性约束，样条模型既能捕捉局部特征，又能保证整体光滑，特别适用于没有明确函数形式但需要灵活拟合的情形。自然样条和光滑样条更是提供了边界稳定性和自动光滑参数选择的优雅解决方案。

在估计方法层面，我们认识到非线性模型参数估计的本质是数值优化问题。非线性最小二乘法、梯度下降法及其变体构成了求解这类问题的工具箱。每种方法都有其适用场景：高斯-牛顿法在接近

最优解时收敛迅速，梯度下降法对初始值不敏感但可能收敛较慢，而拟牛顿法则在计算效率与收敛性之间取得了良好平衡。

最后，模型比较与诊断提醒我们，无论模型多么复杂，都需要严格的统计验证。残差分析、交叉验证、信息准则等工具帮助我们避免过拟合，确保模型的稳健性和泛化能力。

非线性回归不仅是一套统计方法，更是一种建模哲学：它教导我们在尊重数据复杂性的同时，保持模型的简约性；在追求拟合优度的同时，警惕过度参数化的陷阱。掌握这些方法后，读者将能够更自信地面对现实世界中的曲线关系，从数据中提取更深刻的洞察。

核心公式回顾

1. 多项式回归：$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \varepsilon_i$
2. 正交多项式条件：$\sum_{i=1}^{n} Q_j(x_i) Q_k(x_i) = 0$（$j \neq k$）
3. Box-Cox 变换：$y^{(\lambda)} = \frac{y^\lambda - 1}{\lambda}$（$\lambda \neq 0$）
4. 样条光滑条件：$f_j^{(m)}(\xi_j) = f_{j+1}^{(m)}(\xi_j)$（$m = 0, 1, 2$）
5. 平滑样条目标函数：$\min_f \left\{ \sum (y_i - f(x_i))^2 + \lambda \int [f''(x)]^2 dx \right\}$

方法比较与选择指南

方法类型	适用场景	优势	局限性
多项式回归	光滑的全局趋势	简单直观，易于解释	不够灵活，外推性能差
正交多项式	高阶多项式拟合	数值稳定，系数独立	需要专门的基函数构造
样条回归	局部变化复杂	灵活性高，适应性强	需要选择节点位置
平滑样条	自动光滑度控制	无需选择节点，理论性质好	计算量较大，解释性稍差

重要概念体系

1. 基函数展开：用基函数的线性组合逼近复杂函数关系
2. 正交性原理：通过正交化改善数值稳定性和统计性质
3. 分段多项式：在不同区间使用不同的多项式逼近
4. 光滑性条件：在分段连接处施加连续性约束
5. 惩罚回归：通过惩罚项控制模型复杂度
6. 偏差-方差权衡：在模型复杂度和预测精度间寻求平衡

实践指导原则

1. 从简到繁：先尝试低阶多项式，再考虑更复杂的方法
2. 模型验证：使用交叉验证评估模型泛化能力

3. 数值稳定性：高阶多项式务必使用正交多项式形式
4. 领域知识：结合实际问题背景选择合适的方法
5. 可视化诊断：通过残差图和拟合曲线评估模型 adequacy

多项式回归和样条回归为处理非线性关系提供了系统的框架，它们在线性模型的基础上通过基函数扩展来捕捉复杂的数据模式。理解这些方法的原理和适用条件，对于建立有效的预测模型具有重要意义。

III 进阶——超越线性的机器学习

在前面我们介绍了线性分析方法，包括回归分析和线性分类模型。本章我们将扩展到更复杂的机器学习，涵盖以下内容：- 决策树 - 集成方法（随机森林和梯度提升）- 支持向量机 - 神经网络 - 深度学习

这些方法在处理非线性关系和高维数据时表现出色，广泛应用于分类和回归任务。我们将通过理论讲解和实际案例，帮助读者理解这些技术的基本原理和应用场景。此外，我们还将探讨模型评估与选择的方法，如交叉验证、超参数调优等，确保读者能够构建出性能优良的机器学习模型。通过本章的学习，读者将掌握现代机器学习的核心技术，为后续更深入的研究打下坚实基础。本章内容安排如下：1. 决策树基础 2. 集成方法详解 3. 支持向量机原理 4. 神经网络与深度学习简介

10 决策树与集成学习

本章导读

在前面的学习中，我们分别建立了回归分析框架（针对连续型响应变量）和分类分析框架（针对类别型响应变量）。决策树方法为这两类问题提供了统一的建模视角，它通过直观的树状结构将复杂的预测问题分解为一系列简单的决策规则。集成学习则进一步通过模型组合的策略，显著提升了单一模型的预测性能。本章将从回归和分类的双重角度，系统阐述决策树方法的理论基础和实践应用。

10.1 决策树的基本框架

10.1.1 统一视角下的树模型

决策树的核心思想是通过递归地划分特征空间来构建预测模型，这种思想既适用于回归问题，也适用于分类问题。

树结构的通用组成：- 根节点：包含完整的训练数据集 - 内部节点：基于某个特征的决策规则 - 分支：决策规则的可能结果 - 叶节点：最终的预测输出

关键区别：- 回归树：叶节点输出连续数值预测 - 分类树：叶节点输出类别标签或概率

10.1.2 特征空间的划分策略

决策树通过轴平行分割将特征空间划分为矩形区域，每个区域对应一个叶节点。

数学表述：对于 p 维特征空间，决策树构建了一个划分：

$$R_1, R_2, \ldots, R_M \quad \text{满足} \quad \bigcup_{m=1}^{M} R_m = \mathbb{R}^p$$

其中每个区域 R_m 对应树的一个叶节点。

10.2 回归树：连续响应的建模

10.2.1 回归树的预测机制

叶节点预测值：对于落入区域 R_m 的观测，回归树给出该区域内所有训练样本响应值的均值作为预测：

$$\hat{y}_{R_m} = \frac{1}{N_m} \sum_{x_i \in R_m} y_i = \bar{y}_{R_m}$$

其中 N_m 是区域 R_m 中包含的训练样本数。

模型表示：回归树模型可以表示为：

$$f(x) = \sum_{m=1}^{M} \bar{y}_{R_m} \cdot I(x \in R_m)$$

10.2.2 回归树的分裂准则

目标函数：选择分裂特征 j 和分裂点 s 来最小化加权平方误差：

$$\min_{j,s} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

其中：- $R_1(j,s) = \{X | X_j \leq s\}$（左子节点）- $R_2(j,s) = \{X | X_j > s\}$（右子节点）- c_1, c_2 分别是左右子节点的最优预测值（即样本均值）

分裂增益：分裂带来的平方误差减少量为：

$$\Delta SSE = SSE_{\text{parent}} - (SSE_{\text{left}} + SSE_{\text{right}})$$

其中 $SSE = \sum (y_i - \bar{y})^2$

10.2.3 回归树的复杂度控制

预剪枝策略：- 最大树深度 - 叶节点最小样本数 - 分裂所需最小改进量

后剪枝策略：代价复杂度剪枝，最小化：

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m \cdot MSE(R_m) + \alpha|T|$$

其中 $MSE(R_m) = \frac{1}{N_m}\sum_{x_i \in R_m}(y_i - \bar{y}_{R_m})^2$

10.3 分类树：类别响应的建模

10.3.1 分类树的预测机制

叶节点预测：对于落入区域 R_m 的观测，分类树给出该区域内最频繁的类别：

$$\hat{y}_{R_m} = \arg\max_k \left(\frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \right)$$

或者输出类别概率：

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

10.3.2 分类树的不纯度度量

不纯度度量用于评估节点内类别的混杂程度，理想的不纯度函数应满足：1. 在均匀分布时取得最大值 2. 在纯节点时取得最小值（0）3. 是对称的凹函数

Gini 不纯度：

$$\text{Gini}(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

- 直观解释：从节点中随机抽取两个样本，它们属于不同类别的概率
- 取值范围：[0, 1-1/K]
- 对类别分布变化敏感

信息熵：

$$\text{Entropy}(p) = -\sum_{k=1}^{K} p_k \log p_k$$

- 基于信息论的概念
- 在机器学习中广泛使用
- 计算相对复杂

误分类误差：

$$\text{Error}(p) = 1 - \max_k p_k$$

- 直观但不够敏感
- 不适合作为分裂准则，常用于剪枝

10.3.3 分类树的分裂准则

不纯度减少量：选择使子节点不纯度加权和减少最多的分裂：

$$\Delta I = I(\text{parent}) - \left(\frac{N_{\text{left}}}{N} I(\text{left}) + \frac{N_{\text{right}}}{N} I(\text{right}) \right)$$

其中 $I(\cdot)$ 是选择的不纯度函数。

信息增益（使用熵不纯度时）：

$$\text{Information Gain} = \text{Entropy}(p) - \sum_{j=1}^{2} \frac{N_j}{N} \text{Entropy}(p_j)$$

10.4 树模型的比较与诊断

10.4.1 回归树与分类树的统一视角

共同特点：1. 非参数方法：不对数据分布做强假设 2. 特征选择：自动选择重要特征 3. 处理混合类型数据：能同时处理数值型和类别型特征 4. 稳健性：对异常值和缺失值相对稳健

主要差异：

方面	回归树	分类树
响应变量	连续型	类别型
叶节点预测	均值	众数或概率
分裂准则	平方误差减少	不纯度减少
模型评估	MSE, R^2	准确率, AUC
复杂度控制	基于 MSE 的代价复杂度	基于不纯度的代价复杂度

10.4.2 树模型的优势与局限

主要优势：1. 直观易懂：决策规则可解释性强 2. 无需预处理：对特征缩放不敏感 3. 处理非线性：自动捕捉变量间的交互效应 4. 变量重要性：提供特征重要性的自然度量

主要局限：1. 不稳定性：数据微小变化可能导致树结构巨大改变 2. 贪婪性：每一步选择局部最优，未必全局最优 3. 预测面不光滑：分段常数预测 4. 外推能力差：无法预测训练数据范围外的值

10.5 集成学习理论基础

10.5.1 集成学习的统计原理

集成学习通过组合多个基学习器来改善预测性能，其理论基础可以从偏差-方差分解的角度理解。

回归问题的偏差-方差分解：

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}^2(\hat{f}(x)) + \text{Var}(\hat{f}(x)) + \sigma_\varepsilon^2$$

分类问题的误差分解：虽然分类问题没有精确的偏差-方差分解，但类似概念仍然适用：- 偏差：模型系统性错误 - 方差：模型对训练数据变化的敏感性

10.5.2 集成方法的分类

根据基学习器的生成方式和组合策略，集成方法主要分为：

1. 并行方法：基学习器独立生成，如 Bagging、随机森林
2. 序列方法：基学习器顺序生成，如 Boosting
3. 异质集成：组合不同类型的学习器，如 Stacking

10.6 Bagging 与随机森林

10.6.1 Bagging 的回归应用

Bootstrap 采样：从训练集中有放回地抽取 B 个自助样本，每个样本大小与原始训练集相同。

回归预测聚合：

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

方差减少机制：假设基学习器方差为 σ^2，相关系数为 ρ，则 Bagging 集成方差为：

$$\text{Var}(\hat{f}_{\text{bag}}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

10.6.2 Bagging 的分类应用

多数投票：

$$\hat{y}_{\text{bag}}(x) = \arg\max_k \sum_{b=1}^{B} I(\hat{f}^{*b}(x) = k)$$

概率平均：

$$\hat{p}_k(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{p}_k^{*b}(x)$$

10.6.3 随机森林的改进

随机森林在 Bagging 的基础上引入特征随机选择，进一步降低基学习器间的相关性。

特征采样策略：- 分类问题：$m \approx \sqrt{p}$ - 回归问题：$m \approx p/3$

特征重要性：基于不纯度减少的平均值：

$$\text{Importance}(X_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{t \in T_b} \Delta I(t, X_j)$$

10.7 Boosting 方法

10.7.1 回归问题的梯度提升

加性模型：

$$f_M(x) = \sum_{m=1}^{M} \beta_m h_m(x)$$

梯度下降视角：在函数空间中进行梯度下降，每一步拟合当前模型的负梯度：

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

对于平方损失：

$$r_{im} = y_i - f_{m-1}(x_i)$$

算法步骤：1. 初始化 $f_0(x) = \bar{y}$ 2. 对于 $m = 1$ 到 M：- 计算伪残差 $r_{im} = y_i - f_{m-1}(x_i)$ - 用决策树拟合 $\{(x_i, r_{im})\}$ 得到 $h_m(x)$ - 更新 $f_m(x) = f_{m-1}(x) + \nu \cdot h_m(x)$

其中 ν 是学习率。

10.7.2 分类问题的 **AdaBoost**

AdaBoost 通过调整样本权重来关注难以分类的样本。

指数损失函数：

$$L(y, f(x)) = \exp(-yf(x))$$

其中 $y \in \{-1, +1\}$

算法流程：1. 初始化权重 $w_i = 1/N$ 2. 对于 $m = 1$ 到 M：- 训练弱分类器 $G_m(x)$ 最小化加权误差 - 计算误差率 $\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$ - 计算分类器权重 $\alpha_m = \frac{1}{2}\log\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$ - 更新样本权重 $w_i \leftarrow w_i \exp(-\alpha_m y_i G_m(x_i))$ - 权重归一化 3. 输出最终分类器 $\text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$

10.10 集成学习的高级应用

10.10.1 回归与分类的性能差异

回归问题的集成：

- - 主要降低方差
- - 对基学习器的准确性要求相对较低
- – 容易实现预测区间估计

分类问题的集成：

- – 同时影响偏差和方差
- – 对基学习器的多样性要求更高
- – 类别不平衡时需要特殊处理

10.10.2 模型解释性工具

特征重要性：无论回归还是分类，都可以通过特征在集成中的总不纯度减少来评估重要性。

部分依赖图：显示某个特征对预测值的边际效应，适用于回归和概率预测。

SHAP 值：统一框架下的特征贡献度分析，为每个预测提供可解释的分解。

10.11 案例分析

本章总结

核心公式回顾

1. 回归树预测：$\hat{y}_{R_m} = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$
2. 分类树 Gini 不纯度：$\text{Gini}(p) = 1 - \sum_{k=1}^{K} p_k^2$
3. Bagging 回归：$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$
4. 梯度提升伪残差：$r_{im} = y_i - f_{m-1}(x_i)$
5. AdaBoost 分类器权重：$\alpha_m = \frac{1}{2} \log\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$

回归与分类的对比总结

方面	回归树	分类树
响应变量	连续数值	离散类别
预测输出	叶节点均值	叶节点众数或概率
损失函数	平方误差	Gini/熵不纯度
集成目标	主要降低方差	平衡偏差与方差
评估指标	MSE, RMSE, R^2	准确率, 精确率, 召回率

方法选择指南

问题类型	推荐方法	关键考虑
需要强解释性	单棵决策树	深度限制，剪枝强度
回归预测精度	梯度提升树	学习率，树深度，迭代次数
分类平衡数据	随机森林	树数量，特征采样比例
类别不平衡	AdaBoost	关注困难样本的能力
计算效率优先	Bagging	可并行化，内存需求

重要概念体系

1. 递归划分：通过特征测试构建层次决策结构
2. 不纯度度量：量化节点内响应变量的混杂程度
3. 模型集成：通过组合多个基学习器提升预测性能
4. 偏差-方差权衡：在不同集成方法中的差异化表现
5. 特征重要性：基于树模型的特征贡献度评估
6. 稳健预测：通过模型平均降低方差和过拟合风险

决策树和集成学习方法为回归和分类问题提供了强大而灵活的解决方案。从单一树的直观解释到集成方法的高精度预测，这一方法体系涵盖了从探索性分析到生产部署的全流程需求。理解这些方法在回归和分类背景下的共性与差异，有助于在实际问题中选择最适合的技术路线。

11 支持向量机

本章导读

支持向量机（Support Vector Machine, SVM）是一类强大的监督学习算法，在线性分类和回归问题中表现出色。本章将从几何直观和数学优化两个角度，系统介绍 SVM 的基本原理、核方法以及在实际问题中的应用。SVM 的核心思想是通过寻找最优分离超平面来实现分类，并通过核技巧处理非线性问题。

11.1 线性支持向量分类器

11.1.1 最大间隔分类器

基本概念：对于线性可分的二分类问题，存在无数个超平面可以将两类样本分开。最大间隔分类器选择那个具有最大间隔的超平面。

数学表述：考虑训练数据 $\{(x_i, y_i)\}_{i=1}^n$，其中 $y_i \in \{-1, +1\}$。分离超平面可以表示为：

$$w^T x + b = 0$$

函数间隔：样本点 (x_i, y_i) 到超平面的函数间隔定义为：

$$\hat{\gamma}_i = y_i(w^T x_i + b)$$

几何间隔：几何间隔是函数间隔除以权重向量的范数：

$$\gamma_i = \frac{y_i(w^T x_i + b)}{\|w\|} = \frac{\hat{\gamma}_i}{\|w\|}$$

11.1.2 优化问题 formulation

最大间隔优化问题：寻找超平面使得最小几何间隔最大化：

$$\begin{aligned}&\max_{w,b} \gamma \\ &\text{满足} \frac{y_i(w^T x_i + b)}{\|w\|} \geq \gamma, \quad i = 1, \ldots, n\end{aligned}$$

等价形式：通过缩放约束，可以得到等价的凸优化问题：

$$\begin{aligned}&\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{满足} y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n\end{aligned}$$

11.1.3 支持向量的概念

支持向量：是那些恰好满足 $y_i(w^T x_i + b) = 1$ 的样本点，它们定义了间隔的边界，并完全决定了最优超平面。

关键性质：- 支持向量是距离分离超平面最近的样本点 - 移除非支持向量不会影响模型 - 支持向量的数量通常很少，使得 SVM 具有稀疏性

11.2 软间隔支持向量机

11.2.1 线性不可分情况

在实际问题中，数据往往不是线性可分的。软间隔 SVM 通过引入松弛变量来处理这种情况。

松弛变量：对于每个样本引入 $\xi_i \geq 0$，允许某些样本违反间隔约束：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n$$

优化问题：

$$\begin{aligned}&\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i \\ &\text{满足} y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ &\xi_i \geq 0, \quad i = 1, \ldots, n\end{aligned}$$

11.2.2 参数 **C** 的解释

正则化参数：- C 控制间隔宽度与分类错误之间的权衡 - $C \to \infty$：硬间隔 SVM，不允许任何分类错误 - $C \to 0$：最大化间隔，容忍更多分类错误

实际选择：C 通常通过交叉验证选择，平衡模型的复杂度和训练误差。

11.2.3 损失函数视角

合页损失：软间隔 SVM 等价于最小化合页损失：

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b))$$

与其他方法的比较：- 合页损失对正确分类且置信度高的样本损失为 0 - 与逻辑回归的交叉熵损失形成对比

11.3 对偶问题与核方法

11.3.1 拉格朗日对偶

原始问题：

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$
$$\text{满足} y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

拉格朗日函数：

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i[y_i(w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{n} \mu_i \xi_i$$

对偶问题：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$\text{满足} 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$$
$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

11.3.2 核技巧

特征映射：通过非线性映射 $\phi : \mathcal{X} \to \mathcal{H}$ 将输入空间映射到高维特征空间。

核函数：核函数 $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ 计算特征空间中的内积，而无需显式计算特征映射。

对偶问题的核化形式：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{满足} 0 \leq \alpha_i \leq C, \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

11.3.3 常用核函数

线性核：

$$K(x_i, x_j) = x_i^T x_j$$

多项式核：

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$$

高斯径向基核：

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

Sigmoid 核：

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$$

11.4 支持向量回归

11.4.1 ϵ-不敏感损失

回归问题设定：对于回归问题 $y_i \in \mathbb{R}$，SVM 的扩展称为支持向量回归。

ϵ-不敏感损失：

$$L_\epsilon(y, f(x)) = \begin{cases} 0 & \text{if } |y - f(x)| \leq \epsilon \\ |y - f(x)| - \epsilon & \text{otherwise} \end{cases}$$

11.4.2 SVR 优化问题

原始问题：

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$
$$\text{满足} y_i - (w^T x_i + b) \leq \epsilon + \xi_i$$
$$(w^T x_i + b) - y_i \leq \epsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, n$$

对偶问题：

$$\max_{\alpha,\alpha^*} -\epsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{n} y_i(\alpha_i - \alpha_i^*)$$
$$-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j)$$
$$\text{满足} 0 \leq \alpha_i, \alpha_i^* \leq C, \quad \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$

11.4.3 预测函数

SVR 预测：

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b$$

支持向量：只有那些满足 $|\alpha_i - \alpha_i^*| > 0$ 的样本才是支持向量。

11.5 模型选择与评估

11.5.1 超参数调优

关键超参数：- 正则化参数 C - 核参数（如 RBF 核的 γ）- ϵ（对于 SVR）

网格搜索：在超参数空间中进行系统搜索，使用交叉验证评估性能。

交叉验证策略：- k-折交叉验证 - 分层交叉验证（用于分类）- 时间序列交叉验证（用于时间相关数据）

11.5.2 核函数选择

核选择指南：

数据类型	推荐核函数	理由
线性可分	线性核	简单，计算高效
文本数据	线性核	高维稀疏数据
非线性问题	RBF 核	通用性强，适应各种模式
先验知识	自定义核	利用领域特定知识

11.5.3 模型评估指标

分类问题：- 准确率、精确率、召回率、F1 分数 - ROC 曲线和 AUC - 混淆矩阵

回归问题：- 均方误差（MSE）- 平均绝对误差（MAE）- R^2 决定系数

11.6 SVM 的扩展变体

11.6.1 多类 SVM

一对多方法：为每个类别训练一个二分类 SVM，将该类与其他所有类别区分。

一对一方法：为每对类别训练一个二分类 SVM，然后通过投票决定最终类别。

多类 **SVM** 的统一公式：直接扩展 SVM 到多类情况，使用单一优化问题。

11.6.2 概率输出

Platt 缩放：将 SVM 输出通过 sigmoid 函数转换为概率估计：

$$P(y=1|x) = \frac{1}{1+\exp(Af(x)+B)}$$

其中 A, B 通过最大似然估计得到。

11.6.3 大规模 SVM

挑战：标准 SVM 算法的时间复杂度为 $O(n^3)$，空间复杂度为 $O(n^2)$，难以处理大规模数据。

解决方案：- 序列最小优化（SMO）算法 - 随机梯度下降 - 近似核方法

11.7 SVM 与其他方法的比较

11.7.1 与逻辑回归的比较

相似点：- 都是线性分类器 - 都可以通过核技巧处理非线性问题 - 都有正则化项控制模型复杂度

不同点：- 损失函数：合页损失 vs 交叉熵损失 - 输出：决策函数 vs 概率估计 - 支持向量：稀疏解 vs 稠密解

11.7.2 与神经网络的比较

优势：- 理论基础坚实 - 全局最优解 - 对高维数据有效

劣势：- 核函数选择困难 - 大规模数据计算成本高 - 特征工程相对重要

11.8 案例分析

本章总结

核心公式回顾

1. 最大间隔优化：$\min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $y_i(w^T x_i + b) \geq 1$
2. 软间隔 SVM：$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$
3. 对偶问题：$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
4. RBF 核：$K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$
5. SVR 预测：$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(x_i, x) + b$

方法选择指南

问题类型	推荐方法	关键参数	注意事项
线性可分分类	硬间隔 SVM	无	数据需严格线性可分
一般分类	软间隔 SVM	C, 核参数	通过交叉验证选择参数
非线性分类	核 SVM	C, γ	小心过拟合
回归问题	SVR	C, ϵ, γ	ϵ 控制预测精度

问题类型	推荐方法	关键参数	注意事项
大规模数据	线性 SVM	C	使用随机梯度下降

重要概念体系

1. 最大间隔原理：基于几何间隔最大化的分类准则
2. 支持向量：决定分类边界的关键样本点
3. 核技巧：通过核函数隐式实现非线性映射
4. 对偶理论：将原问题转化为更易求解的对偶问题
5. 软间隔：通过松弛变量处理线性不可分情况
6. ϵ-不敏感损失：SVR 中使用的稳健损失函数

实践指导原则

1. 数据预处理：SVM 对特征缩放敏感，建议标准化数据
2. 核选择：从线性核开始，必要时升级到 RBF 核
3. 参数调优：使用网格搜索和交叉验证系统优化超参数
4. 模型解释：线性 SVM 的权重向量提供特征重要性
5. 计算考虑：对于大规模数据，考虑线性 SVM 或近似方法

支持向量机提供了坚实的理论基础和优秀的实践性能，特别适合中小规模的高维数据问题。理解 SVM 的几何直观和优化理论，有助于在实际应用中更好地使用和解释这一强大工具。

12 神经网络与深度学习基础

本章导读

神经网络是预测建模领域的重要方法，它通过模拟人脑神经元的工作方式，为回归和分类问题提供了统一的解决方案。本章将从预测建模的本质出发，介绍神经网络的基本原理、网络设计、训练方法，并简要探讨深度学习的概念，帮助理解这一强大工具在回归和分类任务中的应用。

12.1 神经网络：回归与分类的统一框架

12.1.1 从线性模型到神经网络

线性模型的回顾：- 线性回归：$y = w^T x + b$（连续输出）- 逻辑回归：$P(y = 1|x) = \sigma(w^T x + b)$（概率输出）

神经网络的扩展思路：单一神经元可以看作一个广义线性模型，通过组合多个神经元并引入非线性激活函数，神经网络能够学习更复杂的模式。

基本神经元模型：

$$z = w^T x + b \quad \text{(线性组合)}$$
$$a = \sigma(z) \quad \text{(非线性变换)}$$

12.1.2 为什么需要神经网络？

传统方法的局限性：- 线性模型只能捕捉线性关系 - 多项式回归需要手动设计特征组合 - 对复杂非线性模式建模能力有限

神经网络的优势：- 自动学习特征组合 - 通过层次化结构学习抽象特征 - 统一处理回归和分类问题

12.2 网络架构设计

12.2.1 输出层设计

回归问题的输出层：- 神经元数量：1 个（单输出）或对应输出维度 - 激活函数：恒等函数（无激活）- 输出解释：连续数值预测

二分类问题的输出层：- 神经元数量：1 个 - 激活函数：sigmoid - 输出解释：属于正类的概率

多分类问题的输出层：- 神经元数量：类别数 K - 激活函数：softmax - 输出解释：每个类别的概率分布

12.2.2 隐藏层设计

隐藏层的作用：- 学习输入特征的中间表示 - 通过非线性变换增强模型表达能力 - 自动进行特征工程

网络深度与宽度：- 浅层网络：1-2 个隐藏层，适合简单问题 - 深层网络：多个隐藏层，适合复杂模式 - 宽度：每层神经元数，影响模型容量

12.3 激活函数与损失函数

12.3.1 常用激活函数

Sigmoid 函数：

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- 将输出压缩到 (0,1) - 适合输出层，直观的概率解释 - 梯度消失问题

ReLU 函数：

$$\text{ReLU}(z) = \max(0, z)$$

- 计算简单，训练高效 - 缓解梯度消失 - 现代神经网络的首选

Tanh 函数：

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

- 输出范围 (-1,1)，零中心化 - 比 sigmoid 有更好的梯度特性

12.3.2 损失函数选择

回归任务：均方误差损失：

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

二分类任务：交叉熵损失：

$$L = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

多分类任务：多类交叉熵损失：

$$L = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{ik}\log(\hat{y}_{ik})$$

12.4 神经网络训练

12.4.1 前向传播

计算过程：从输入层开始，逐层计算直到输出层：

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = \sigma^{[l]}(z^{[l]})$$

预测输出：最终层的激活值即为模型预测。

12.4.2 反向传播

梯度计算：利用链式法则计算损失函数对每个参数的梯度：

$$\frac{\partial L}{\partial W^{[l]}} = \frac{\partial L}{\partial z^{[l]}}(a^{[l-1]})^T$$

参数更新：使用梯度下降算法更新参数：

$$W^{[l]} := W^{[l]} - \alpha\frac{\partial L}{\partial W^{[l]}}$$

12.4.3 优化算法

批量梯度下降：使用全部训练数据计算梯度，收敛稳定但计算量大。

小批量梯度下降：折中方案，兼顾收敛速度和稳定性。

Adam 优化器：自适应学习率，结合动量项，实践中效果良好。

12.5 过拟合与正则化

12.5.1 过拟合问题

神经网络的特点：- 参数众多，模型容量大 - 容易过拟合训练数据 - 需要正则化技术控制复杂度

检测方法：- 训练误差持续下降，验证误差开始上升 - 学习曲线分析 - 早停法

12.5.2 正则化技术

L2 正则化：在损失函数中加入权重惩罚：

$$L_{\text{reg}} = L + \frac{\lambda}{2} \sum \|W\|^2$$

Dropout：训练时随机丢弃部分神经元，提高泛化能力。

数据增强：通过变换生成更多训练样本。

12.6 深度学习简介

深度学习模型可以形式化地定义为：

$$f(\mathbf{x}; \theta) = f_L(f_{L-1}(\cdots f_1(\mathbf{x}; \theta_1) \cdots; \theta_{L-1}); \theta_L)$$

其中：- $\mathbf{x}$：输入向量 - $\theta = \{\theta_1, \theta_2, \ldots, \theta_L\}$：模型参数 - f_l：第 l 层的变换函数 - L：网络总层数（深度）

与浅层学习的对比

特征	浅层学习	深度学习
层数	1-2 层隐藏层	多层（通常 >3 层）
特征工程	需要手动设计	自动学习特征
数据需求	相对较少	大量数据
计算需求	相对较低	高计算需求
可解释性	较好	较差（黑箱问题）

12.6.2 卷积神经网络（**CNN**）

CNN 的基本思想

卷积神经网络专门处理具有网格结构的数据（如图像），通过局部连接和权值共享大幅减少参数数量。

卷积操作

二维离散卷积：

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n)$$

其中 I 为输入图像，K 为卷积核。

卷积层特性

1. 局部连接：每个神经元只连接输入图像的局部区域
2. 权值共享：同一卷积核在输入的不同位置使用相同权重
3. 平移不变性：能够检测输入任何位置的特征

CNN 的基本组件

卷积层

输出特征图大小计算：

$$\text{输出高度} = \frac{H - F_H + 2P}{S} + 1$$

$$\text{输出宽度} = \frac{W - F_W + 2P}{S} + 1$$

其中：- H, W：输入高度和宽度 - F_H, F_W：卷积核高度和宽度 - P：填充（padding）大小 - S：步长（stride）

池化层

1. 最大池化：

$$\text{output} = \max(\text{window})$$

2. 平均池化：

$$\text{output} = \text{mean}(\text{window})$$

池化作用：降维、减少过拟合、增加平移不变性。

全连接层

在 CNN 末端，将特征图展平后连接全连接层进行分类。

经典 CNN 架构

LeNet-5（1998）

- 输入：32×32 灰度图像
- 结构：2 个卷积层 + 2 个池化层 + 3 个全连接层
- 应用：手写数字识别

AlexNet（2012）

- 创新：ReLU 激活函数、Dropout、数据增强
- 结构：5 个卷积层 + 3 个池化层 + 3 个全连接层
- 成就：ImageNet 竞赛冠军

VGGNet（2014）

- 特点：使用小卷积核（3×3），增加网络深度
- VGG16：13 个卷积层 + 3 个全连接层
- VGG19：16 个卷积层 + 3 个全连接层

ResNet（2015）

- 创新：残差连接，解决梯度消失问题
- 残差块：

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

- 深度：ResNet-50, ResNet-101, ResNet-152

12.6.3 循环神经网络（**RNN**）

序列数据处理

循环神经网络用于处理序列数据，具有时间维度上的记忆能力。

RNN 基本结构

$$\mathbf{h}_t = \phi(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h)$$

$$\mathbf{y}_t = \psi(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y)$$

其中：- $\mathbf{h}_t$：时刻 t 的隐藏状态 - $\mathbf{x}_t$：时刻 t 的输入 - $\mathbf{y}_t$：时刻 t 的输出

不同类型的 RNN

1. 一对一：标准神经网络
2. 一对多：图像描述生成
3. 多对一：情感分析
4. 多对多（等长）：词性标注
5. 多对多（不等长）：机器翻译

长短期记忆网络（**LSTM**）

LSTM 通过门控机制解决长期依赖问题。

LSTM 单元结构

1. 遗忘门：

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f)$$

2. 输入门：

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)$$

3. 细胞状态更新：

$$\mathbf{C}_t = \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t$$

4. 输出门：

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

门控循环单元（GRU）

GRU 是 LSTM 的简化版本，合并遗忘门和输入门：

$$\mathbf{z}_t = \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}[\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t])$$

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

12.6.5 注意力机制

缩放点积注意力

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

其中：- Q：查询矩阵 - K：键矩阵 - V：值矩阵 - d_k：键的维度

多头注意力

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Transformer 架构

编码器-解码器结构

编码器（N 层）：1. 多头自注意力 2. 前馈神经网络 3. 残差连接 + 层归一化

解码器（N 层）：1. 掩码多头自注意力（防止看到未来信息）2. 多头编码器-解码器注意力 3. 前馈神经网络 4. 残差连接 + 层归一化

位置编码

为序列添加位置信息：

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d})$$

BERT 与 GPT

BERT（双向编码器表示）

- 预训练任务：掩码语言模型 + 下一句预测
- 特点：双向上下文理解
- 应用：文本分类、问答、命名实体识别

GPT（生成式预训练 Transformer）

- 架构：仅解码器的 Transformer
- 预训练：自回归语言建模
- 特点：强大的生成能力

12.7 案例分析

本章总结

核心概念回顾

1. 统一框架：神经网络为回归和分类提供统一建模框架
2. 架构设计：根据问题类型设计输出层，回归用线性输出，分类用 sigmoid/softmax 输出
3. 激活函数：ReLU 适合隐藏层，根据任务选择输出层激活函数
4. 训练原理：前向传播计算预测，反向传播计算梯度，梯度下降更新参数
5. 正则化：使用 L2 正则化、Dropout 等技术防止过拟合

实践应用指南

数据预处理要求：- 特征标准化：加速收敛，提高数值稳定性 - 数据量要求：神经网络通常需要较多训练数据 - 计算资源：深层网络需要较强的计算能力

与传统方法的选择：

考虑因素	选择传统方法	选择神经网络
数据量	小样本	大样本
问题复杂度	简单线性关系	复杂非线性关系
可解释性	要求高	要求低
计算资源	有限	充足
特征工程	手动设计	自动学习

深度学习扩展

成功应用领域：- 计算机视觉：图像分类、目标检测 - 自然语言处理：文本分类、机器翻译 - 语音识别：语音转文本、声纹识别 - 推荐系统：个性化推荐

实践建议：1. 从基础开始：先掌握浅层神经网络，再学习深度学习 2. 理解原理：不仅要会使用，更要理解背后的数学原理 3. 项目驱动：通过实际项目加深理解 4. 持续学习：深度学习领域发展迅速，需要持续跟进

与前面章节的联系

神经网络不是孤立的方法，而是前面学习内容的自然延伸：

- 单层神经网络 = 广义线性模型
- 多层神经网络 = 多个广义线性模型的堆叠 + 非线性变换
- 深度学习 = 更深层次的特征学习

通过本教材的学习，希望读者能够建立从传统统计方法到现代机器学习的完整知识体系，在实际问题中选择合适的建模方法。

全书总结：从线性回归到神经网络，我们建立了一套完整的预测建模方法体系。每种方法都有其适用场景和局限性，在实际工作中需要根据具体问题选择合适的方法，理解其假设和限制，才能构建出有效的预测模型。

IV 实践——综合案例分析

在前面的章节中，我们介绍了各种回归和分类模型，但是具体的案例分析还没有涉及。接下来的章节将通过一些实际案例来展示如何应用这些模型解决现实中的问题。这些案例将涵盖不同领域，如医疗、金融和市场营销等，帮助读者更好地理解模型的实际应用。

在每个案例中，我们将详细介绍数据的收集与预处理过程，模型的选择与训练方法，以及结果的评估与解释。通过这些实际案例，读者将能够更好地掌握如何将理论知识应用到实践中，从而提升自己的数据分析和建模能力。

特征工程：在实际应用中，数据通常需要经过清洗和转换才能用于建模。特征工程包括处理缺失值、异常值检测、特征缩放和编码等步骤。通过合理的特征工程，可以显著提升模型的性能。模型选择与调优：不同的问题可能适合不同的模型。我们将介绍如何根据数据的特点选择合适的模型，并通过交叉验证和超参数调优等方法优化模型性能。结果解释与可视化：模型训练完成后，理解和解释模型的结果同样重要。我们将介绍一些常用的结果可视化技术，帮助读者更好地理解模型的预测结果和特征的重要性。通过这些实际案例的学习，读者将能够更好地理解如何将回归和分类模型应用到现实问题中，从而提升自己的数据分析和建模能力。

特征工程是机器学习中的核心环节，它通过对原始数据进行转换、组合和提取，构建更能代表预测任务的特征，从而提升模型性能。其内容广泛，可分为以下几个主要方面：

一、数据预处理这是特征工程的基础，旨在处理原始数据中的问题，使其适合建模。1. 缺失值处理：删除（缺失过多的行/列）、填充（均值/中位数/众数、模型预测填充等）。2. 异常值处理：基于统计（如 3σ 原则、IQR 法）、可视化检测，并进行修正、缩尾或删除。3. 数据类型转换：将分类变量编码、将数值变量分箱（离散化）、将文本/日期等转换为模型可处理的格式。

二、特征构造从现有数据中创造新的、更有信息量的特征。1. 领域知识驱动：例如在电商中，从“购买金额” 和 “购买次数” 构造 “客单价”；在时间序列中，提取 “星期几”、“是否节假日” 等。2. 交互特征：通过加减乘除等运算组合现有特征（如 “身高体重指数 BMI= 体重/身高 ^2”）。3. 多项式特征：自动生成特征的高次项和交互项，用于捕捉非线性关系。4. 分解特征：例如将 “交易日期时间” 分解为年、月、日、小时、分钟等独立特征。

三、特征变换通过数学变换改变特征的分布或尺度，使其更符合模型假设或提升效果。1. 标准化/归一化：* 标准化（Z-Score）：使特征均值为 0，方差为 1。适用于许多线性模型（如 SVM、逻辑回归）。* 归一化（Min-Max）：将特征缩放到固定范围（如 [0,1]）。对神经网络、距离类模型（如 KNN）有益。2. 非线性变换：* 对数变换、平方根变换、Box-Cox 变换：用于处理偏态分布，使其

更接近正态分布。* 分箱（离散化）：将连续变量转换为有序的类别，可以捕捉非线性并稳定模型。3. 统计量变换：例如对数值特征进行排序、计算百分位排名。

四、特征编码将非数值特征转换为数值形式。1. 分类变量编码：* 独热编码（One-Hot）：为每个类别创建一个二进制列。适用于无序类别，但维度会膨胀。* 标签编码（Label Encoding）：为每个类别分配一个整数。适用于有序类别或树模型。* 目标编码（Target Encoding）：用目标变量的统计量（如均值）来代表类别。效果强但需小心过拟合。* 频率编码：用类别出现的频率代替类别标签。2. 文本特征编码：* 词袋模型（Bag of Words）、TF-IDF、词向量（Word2Vec, GloVe）等。3. 图像/音频特征提取：使用 SIFT、HOG、MFCC 等传统方法或深度网络中间层输出作为特征。

五、特征选择从大量特征中筛选出最相关、最重要的子集，以降低过拟合风险、减少计算成本并提升模型可解释性。1. 过滤法（Filter）：基于统计指标（如相关系数、卡方检验、互信息）独立评估每个特征与目标的相关性。2. 包装法（Wrapper）：将特征选择看作一个搜索问题，使用模型性能作为评价标准（如递归特征消除 RFE、前向/后向选择）。3. 嵌入法（Embedded）：在模型训练过程中自动进行特征选择（如 LASSO 回归的正则化、决策树/随机森林的特征重要性、XGBoost 的增益）。

六、特征降维当特征过多、高度相关或存在 “维度灾难” 时，通过数学方法将高维特征映射到低维空间，同时尽可能保留信息。1. 线性方法：* 主成分分析（PCA）：寻找方差最大的正交方向进行投影。* 线性判别分析（LDA）：寻找能最好地区分类别的方向进行投影（有监督）。2. 非线性方法：t-SNE、UMAP 等，常用于高维数据的可视化。

七、特征学习（高级）利用模型（尤其是深度学习）自动从原始数据中学习有效的特征表示。1. 自编码器（Autoencoder）2. 深度神经网络：CNN 从图像中学习层次化特征，RNN/LSTM 从序列数据中学习上下文特征。3. 预训练模型：使用 BERT、ResNet 等在大规模数据上预训练的模型进行特征提取或微调。

核心工作流程与原则在实践中，特征工程是一个迭代、探索性的过程：1. 理解数据和业务：这是所有工作的起点。2. 生成候选特征池：通过预处理、构造等方法产生大量特征。3. 评估和筛选：通过选择、降维等方法提炼出高质量特征子集。4. 验证与迭代：将构建的特征输入模型，评估性能，根据反馈（如特征重要性分析、误差分析）进一步调整特征。

核心原则：特征工程的最终目标是让数据更好地 “讲述” 与预测目标相关的故事，为模型提供最大价值的 “燃料”，而非追求复杂的技巧。好的特征工程往往比更换更复杂的模型带来更显著的性能提升。

13 回归任务与基准测试

回归任务导读

回归分析是预测建模中最基础且重要的任务，广泛应用于房价预测、销量预测、风险评估等领域。本章将通过波士顿房价预测案例，完整展示回归问题的解决方案，涵盖数据探索、特征工程、多种回归算法比较、模型评估等关键环节。

数据集说明本案例使用糖尿病数据集（Diabetes Dataset），包含糖尿病患者的生理指标和疾病进展指标。

问题定义 目标：预测糖尿病疾病的进展（target）
任务类型：回归分析
业务价值：为医疗诊断和治疗方案制定提供数据支持

13.1 R 语言实现 (mlr3 框架)

数据加载与探索

```
# 综合案例：糖尿病进展预测 - R 语言实现 (mlr3 框架)

# 加载必要的包
library(mlr3)
library(mlr3verse)
library(mlr3pipelines)
library(mlr3tuning)
library(data.table)
library(ggplot2)
library(corrplot)
library(gridExtra)
```

```
# 加载糖尿病数据集
library(lars)
data(diabetes)
variable_names <- c("age", "sex", "bmi", "map", "tc", "ldl", "hdl", "tch", "
diabetes_data <- as.data.frame(matrix(unlist(diabetes$x), ncol = 10))
colnames(diabetes_data) <- variable_names

str(diabetes_data)
```

```
'data.frame':   442 obs. of  10 variables:
 $ age: num  0.03808 -0.00188 0.0853 -0.08906 0.00538 ...
 $ sex: num  0.0507 -0.0446 0.0507 -0.0446 -0.0446 ...
 $ bmi: num  0.0617 -0.0515 0.0445 -0.0116 -0.0364 ...
 $ map: num  0.02187 -0.02633 -0.00567 -0.03666 0.02187 ...
 $ tc : num  -0.04422 -0.00845 -0.0456 0.01219 0.00393 ...
 $ ldl: num  -0.0348 -0.0192 -0.0342 0.025 0.0156 ...
 $ hdl: num  -0.0434 0.07441 -0.03236 -0.03604 0.00814 ...
 $ tch: num  -0.00259 -0.03949 -0.00259 0.03431 -0.00259 ...
 $ ltg: num  0.01991 -0.06833 0.00286 0.02269 -0.03199 ...
 $ glu: num  -0.01765 -0.0922 -0.02593 -0.00936 -0.04664 ...
```

```
diabetes_data$target <- diabetes$y

# 检查缺失值
cat("\n缺失值检查:\n")
```

缺失值检查:

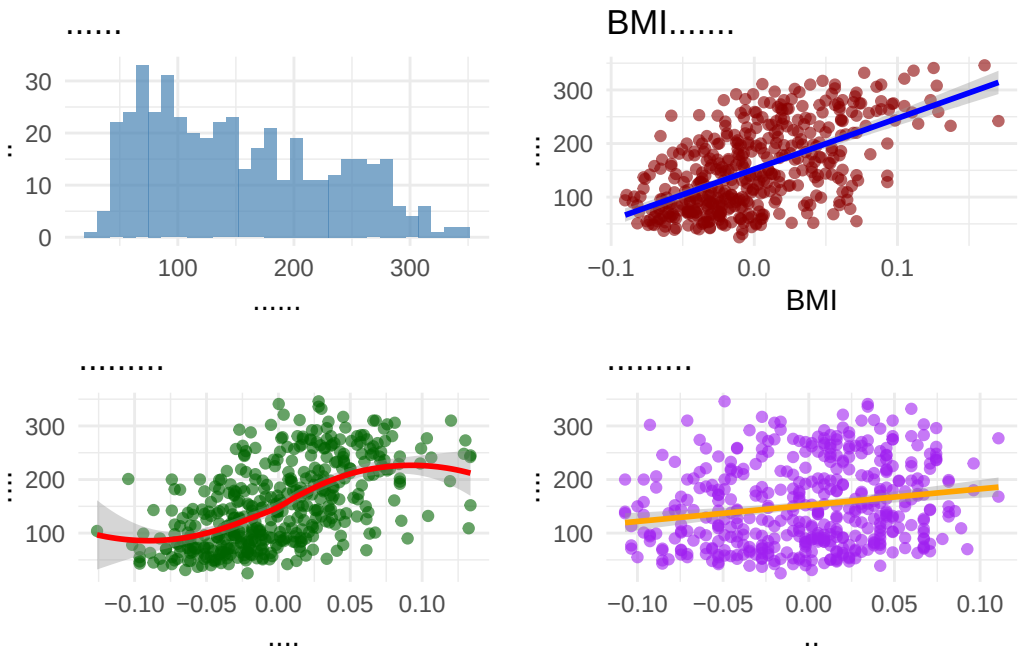
```
print(colSums(is.na(diabetes_data)))
```

```
  age   sex   bmi   map    tc   ldl   hdl   tch   ltg   glu target
    0     0     0     0     0     0     0     0     0     0      0
```

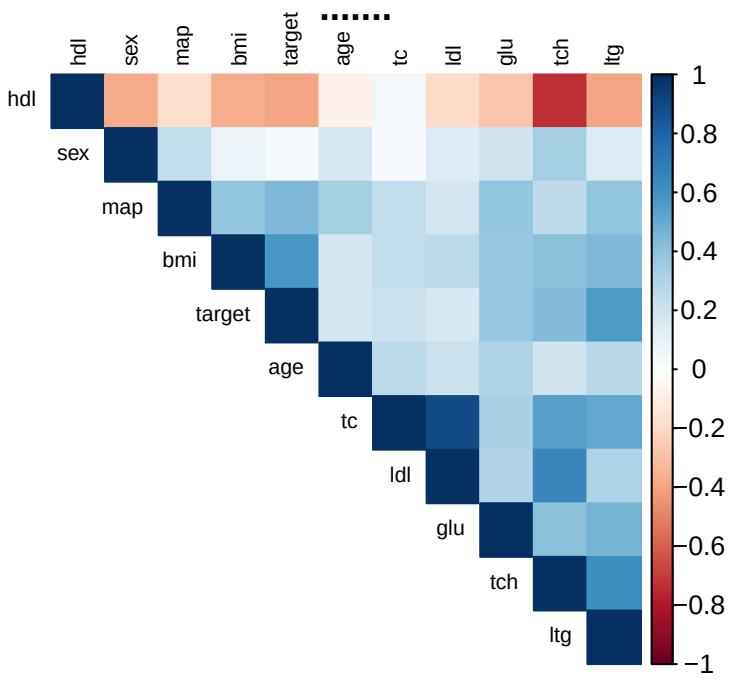
数据可视化分析

```
# 数据可视化分析 - R 语言

# 目标变量分布
```

```r
p1 <- ggplot(diabetes_data, aes(x = target)) +
  geom_histogram(bins = 30, fill = "steelblue", alpha = 0.7) +
  labs(title = " 疾病进展分布", x = " 疾病进展指标", y = " 频数") +
  theme_minimal()

# 疾病进展与关键变量的关系
p2 <- ggplot(diabetes_data, aes(x = bmi, y = target)) +
  geom_point(alpha = 0.6, color = "darkred") +
  geom_smooth(method = "lm", color = "blue") +
  labs(title = "BMI 与疾病进展关系", x = "BMI", y = " 疾病进展") +
  theme_minimal()

p3 <- ggplot(diabetes_data, aes(x = ltg, y = target)) +
  geom_point(alpha = 0.6, color = "darkgreen") +
  geom_smooth(method = "loess", color = "red") +
  labs(title = " 稳定血糖与疾病进展", x = " 稳定血糖", y = " 疾病进展") +
  theme_minimal()

p4 <- ggplot(diabetes_data, aes(x = age, y = target)) +
  geom_point(alpha = 0.6, color = "purple") +
  geom_smooth(method = "lm", color = "orange") +
  labs(title = " 年龄与疾病进展关系", x = " 年龄", y = " 疾病进展") +
  theme_minimal()

# 组合图形
grid.arrange(p1, p2, p3, p4, ncol = 2)
```




```r
# 相关性热图
cor_matrix <- cor(diabetes_data)
corrplot(cor_matrix, method = "color", type = "upper",
         order = "hclust", tl.cex = 0.7, tl.col = "black",
         title = " 变量相关性热图")
```



创建 mlr3 任务和数据预处理

```
# 创建 mlr3 任务和数据预处理

# 创建回归任务
task_diabetes <- as_task_regr(diabetes_data, target = "target", id = "diabetes

# 查看任务信息
cat(" 任务信息:\n")
```

任务信息:

```
print(task_diabetes)
```

```
-- <TaskRegr> (442x11) ---------------------------------------
--------------
* Target: target
* Properties: -
* Features (10):
  * dbl (10): age, bmi, glu, hdl, ldl, ltg, map, sex, tc, tch
```

```
# 数据预处理管道
preprocess_pipeline <- po("scale") %>>%
  po("colapply", applicator = as.numeric)

# 应用预处理
task_preprocessed <- preprocess_pipeline$train(task_diabetes)[[1]]

cat("\n预处理后的任务信息:\n")
```

预处理后的任务信息:

```
print(task_preprocessed)
```

```
-- <TaskRegr> (442x11) ---------------------------------------
--------------
* Target: target
* Properties: -
* Features (10):
  * dbl (10): age, bmi, glu, hdl, ldl, ltg, map, sex, tc, tch
```

模型训练与比较

```
# 模型训练与比较 - mlr3

# 定义学习器
learners <- list(
  lrn("regr.lm", id = "linear_regression"),
  lrn("regr.ranger", id = "random_forest"),
  lrn("regr.xgboost", id = "xgboost"),
  lrn("regr.svm", id = "svm"),
  lrn("regr.kknn", id = "knn")
)

# 设置交叉验证
resampling <- rsmp("cv", folds = 5)

# 基准测试
design <- benchmark_grid(
  tasks = task_preprocessed,
  learners = learners,
  resamplings = resampling
)

bmr <- benchmark(
  design,
  store_backends = TRUE,
  store_models = TRUE)

# 性能评估
cat(" 模型性能比较:\n")
```

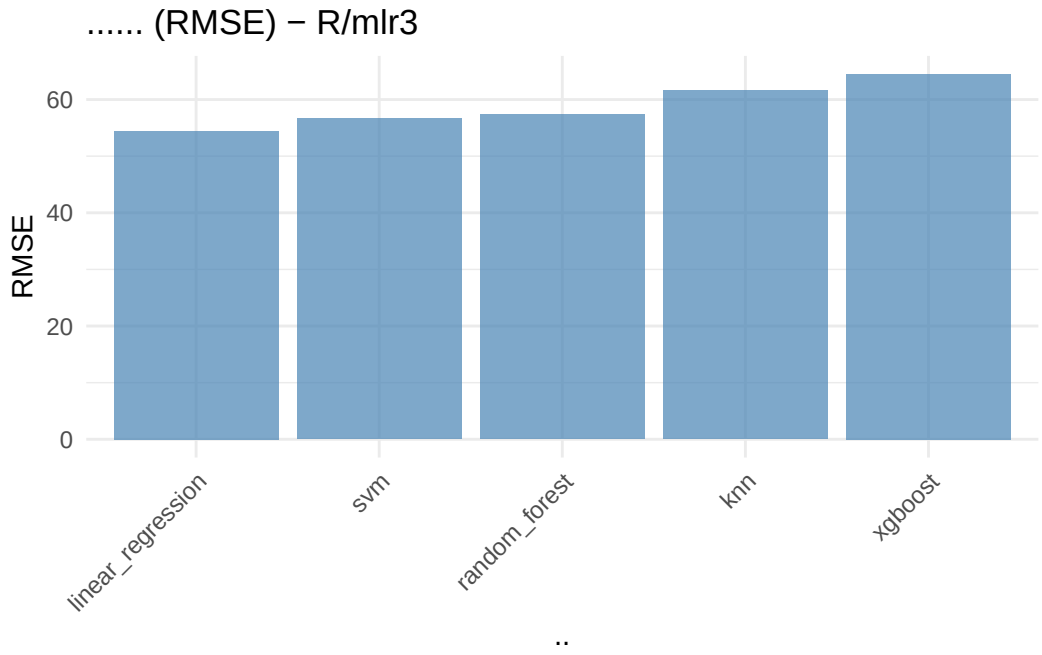
模型性能比较:

```
performance_measures <- msrs(c("regr.mse", "regr.rmse", "regr.mae", "regr.rs
bmr_aggregated <- bmr$aggregate(performance_measures)

# 格式化输出结果
results_r <- bmr_aggregated[, .(
  Model = learner_id,
  RMSE = regr.rmse,
```

```
  MSE = regr.mse,
  MAE = regr.mae,
  R2 = regr.rsq
)]

print(results_r[order(RMSE)])
```

```
                Model      RMSE       MSE       MAE         R2
               <char>     <num>     <num>     <num>      <num>
1: linear_regression 54.47343 2968.864 44.11744 0.4861844
2:                svm 56.62372 3209.881 43.78525 0.4447638
3:      random_forest 57.34437 3294.229 46.87981 0.4328570
4:                knn 61.59499 3801.160 47.45026 0.3463681
5:            xgboost 64.47922 4165.980 50.80858 0.2816711
```

```
# 性能可视化
ggplot(results_r, aes(x = reorder(Model, RMSE), y = RMSE)) +
  geom_bar(stat = "identity", fill = "steelblue", alpha = 0.7) +
  labs(title = " 模型性能比较 (RMSE) - R/mlr3", x = " 模型", y = "RMSE") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



超参数调优

```r
# 超参数调优示例 - 随机森林

library(mlr3)            # 核心
library(mlr3learners)    # 学习器
library(mlr3tuning)      # 参数调优（包含 tnr()）
library(paradox)         # 参数空间
library(mlr3viz)         # 可视化
# 定义随机森林学习器
learner_rf_tune <- lrn("regr.ranger", num.trees = 100)

# 定义超参数空间
param_set <- ps(
  mtry = p_int(lower = 2, upper = ncol(diabetes_data) - 1),
  min.node.size = p_int(lower = 1, upper = 10),
  sample.fraction = p_dbl(lower = 0.6, upper = 0.9)
)

# 定义调优器
tuner <- tnr("grid_search", resolution = 5)
terminator <- trm("evals", n_evals = 10)

# 自动调优
at <- auto_tuner(
  tuner = tuner,
  learner = learner_rf_tune,
  resampling = rsmp("holdout"),
  measure = msr("regr.rmse"),
  search_space = param_set,
  terminator = terminator
)

# 在数据上演示调优
set.seed(123)
at$train(task_preprocessed)

cat(" 最佳参数:\n")
```

最佳参数:

```
print(at$tuning_result)
```

```
   mtry min.node.size sample.fraction learner_param_vals  x_domain regr.rmse
  <int>        <int>           <num>             <list>    <list>     <num>
1:    8            3           0.675          <list[6]> <list[3]>  55.69813
```

```
# 使用调优后的模型进行预测
predictions_tuned <- at$predict(task_preprocessed)
cat(" 调优完成\n")
```

```
调优完成
```

```
cat(" 调优后模型 RMSE:", predictions_tuned$score(msr("regr.rmse")), "\n")
```

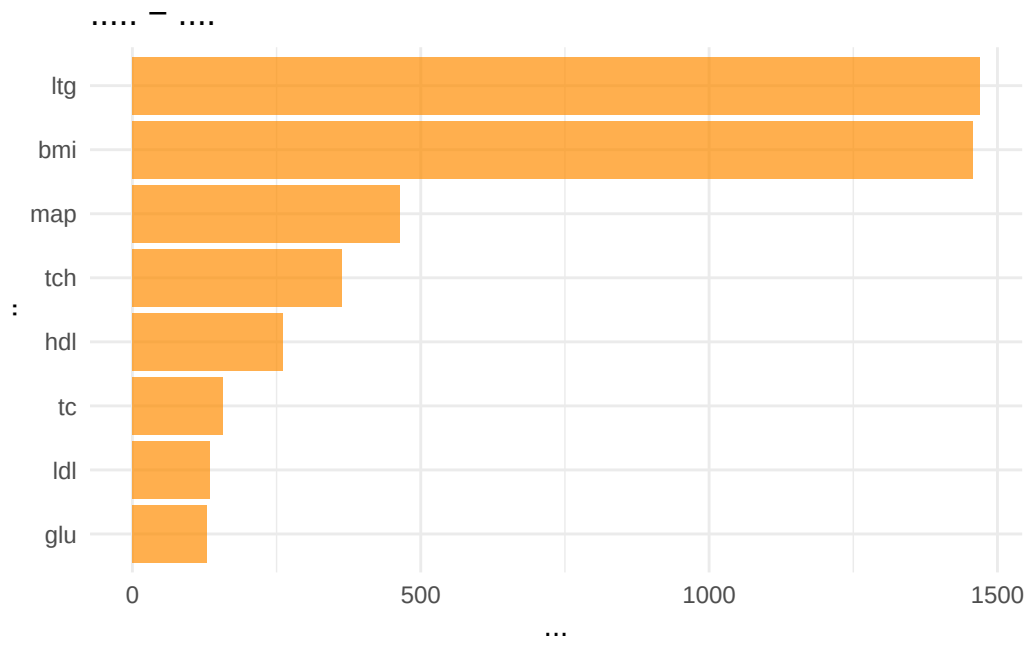
```
调优后模型RMSE: 30.78791
```

模型解释

```
# 模型解释 - 特征重要性

library(ggplot2)
# 训练最佳模型（随机森林）
set.seed(123)
learner_best <- lrn("regr.ranger", importance = "permutation")
learner_best$train(task_preprocessed)

# 特征重要性
if (!is.null(learner_best$importance)) {
  feature_importance <- learner_best$importance()
  importance_df <- data.frame(
    Feature = names(feature_importance),
    Importance = as.numeric(feature_importance)
  )
  importance_df <- importance_df[order(-importance_df$Importance), ]

  cat(" 特征重要性排序:\n")
  print(importance_df)

  # 可视化特征重要性
  ggplot(head(importance_df, 8), aes(x = reorder(Feature, Importance), y = Imp
    geom_bar(stat = "identity", fill = "darkorange", alpha = 0.7) +
```

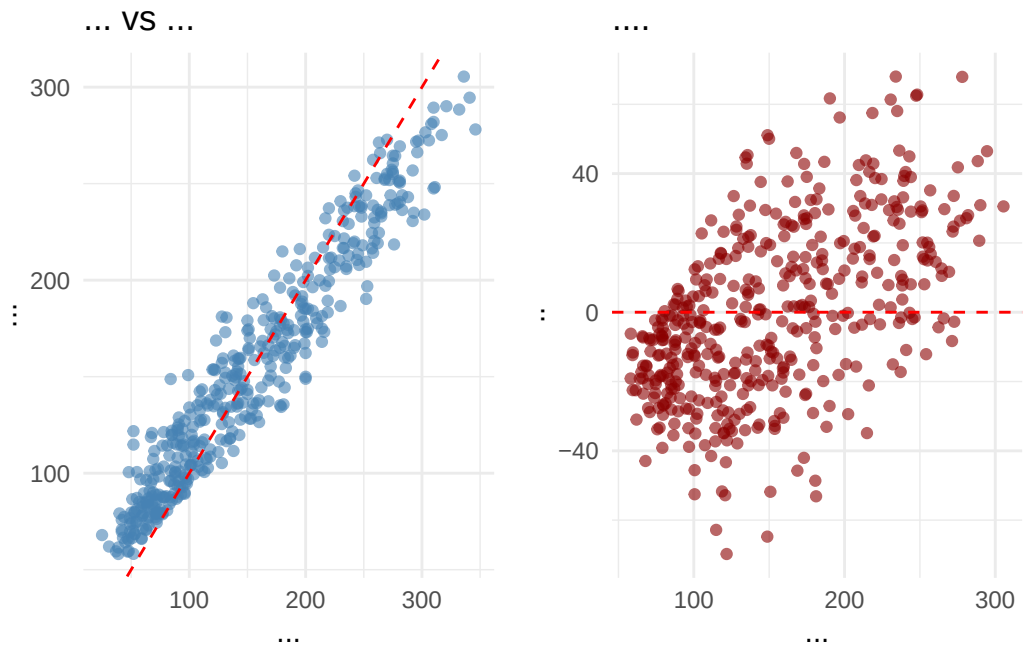
```
    labs(title = " 特征重要性 - 随机森林", x = " 特征", y = " 重要性") +
    coord_flip() +
    theme_minimal()
} else {
  cat(" 该学习器不支持特征重要性计算\n")
}
```

```
特征重要性排序:
   Feature Importance
1      ltg 1469.33233
2      bmi 1457.20570
3      map  464.11349
4      tch  363.93017
5      hdl  261.42699
6       tc  156.66425
7      ldl  135.23885
8      glu  129.35189
9      age   66.95241
10     sex   65.88863
```



模型预测与残差分析

```r
# 模型预测与残差分析

# 使用最佳模型进行预测
predictions <- learner_best$predict(task_preprocessed)

# 创建预测结果数据框
results_df <- data.frame(
  Actual = predictions$truth,
  Predicted = predictions$response,
  Residual = predictions$truth - predictions$response
)

# 预测 vs 实际值图
p1 <- ggplot(results_df, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = " 预测值 vs 实际值", x = " 实际值", y = " 预测值") +
  theme_minimal()

# 残差图
p2 <- ggplot(results_df, aes(x = Predicted, y = Residual)) +
  geom_point(alpha = 0.6, color = "darkred") +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(title = " 残差分析", x = " 预测值", y = " 残差") +
  theme_minimal()

grid.arrange(p1, p2, ncol = 2)
```



13.2 Python (sklearn 框架)–回归

数据加载与探索

```
# 综合案例：糖尿病进展预测 - Python 语言实现 (sklearn 框架)
import sys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split, cross_val_score, GridS
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegresso
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_erro
from sklearn.inspection import permutation_importance
import warnings
warnings.filterwarnings('ignore')
```

```
# 设置图形样式
plt.style.use('default')
sns.set_palette("husl")

# 加载糖尿病数据集
diabetes = pd.read_csv("data/diabetes.csv")
X = diabetes.iloc[:, :-2] # 所有行，除第 3 列（bp）的所有列
y = diabetes.iloc[:, -2]   # 所有行，第 3 列
diabetes_data = pd.concat([X, y], axis=1)

print(" 数据集基本信息:")
```

数据集基本信息:

```
print(f" 样本数: {diabetes_data.shape[0]}")
```

样本数: 442

```
print(f" 变量数: {diabetes_data.shape[1]}")
```

变量数: 9

```
print("\n变量名称:")
```

变量名称:

```
print(diabetes_data.columns.tolist())
```

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5']
```

```
print("\n数据概览:")
```

数据概览:

```
print(diabetes_data.describe())
```

```
              age         sex         bmi  ...          s3          s4          s5
count  442.000000  442.000000  442.000000  ...  442.000000  442.000000  442.000000
mean     0.000136    0.002149   -0.000136  ...    0.000204   -
0.000362    0.000023
```

```
std     0.047790   0.044961   0.047771  ...   0.047320   0.047789   0.047545
min       -0.110000    -0.040000    -0.090000   ...    -0.100000    -
0.080000   -0.130000
25%       -0.037500    -0.040000    -0.030000   ...    -0.037500    -
0.040000   -0.030000
50%        0.010000    -0.040000    -0.010000   ...    -0.010000    -
0.000000    0.000000
75%     0.040000   0.050000   0.030000  ...   0.030000   0.030000   0.030000
max     0.110000   0.050000   0.170000  ...   0.180000   0.190000   0.130000

[8 rows x 9 columns]
```

```
print("\n缺失值检查:")
```

缺失值检查:

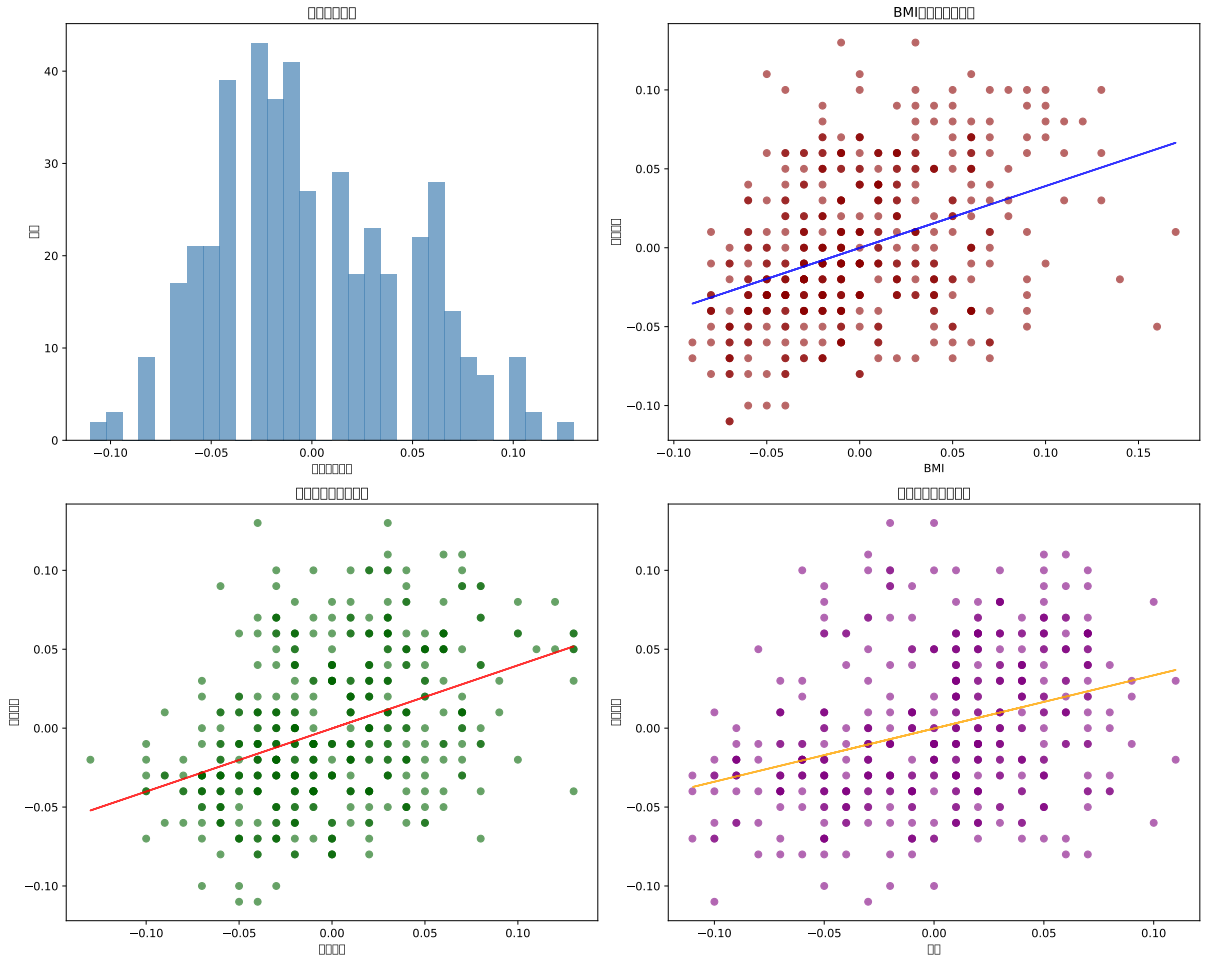
```
print(diabetes_data.isnull().sum())
```

```
age     0
sex     0
bmi     0
bp      0
s1      0
s2      0
s3      0
s4      0
s5      0
dtype: int64
```

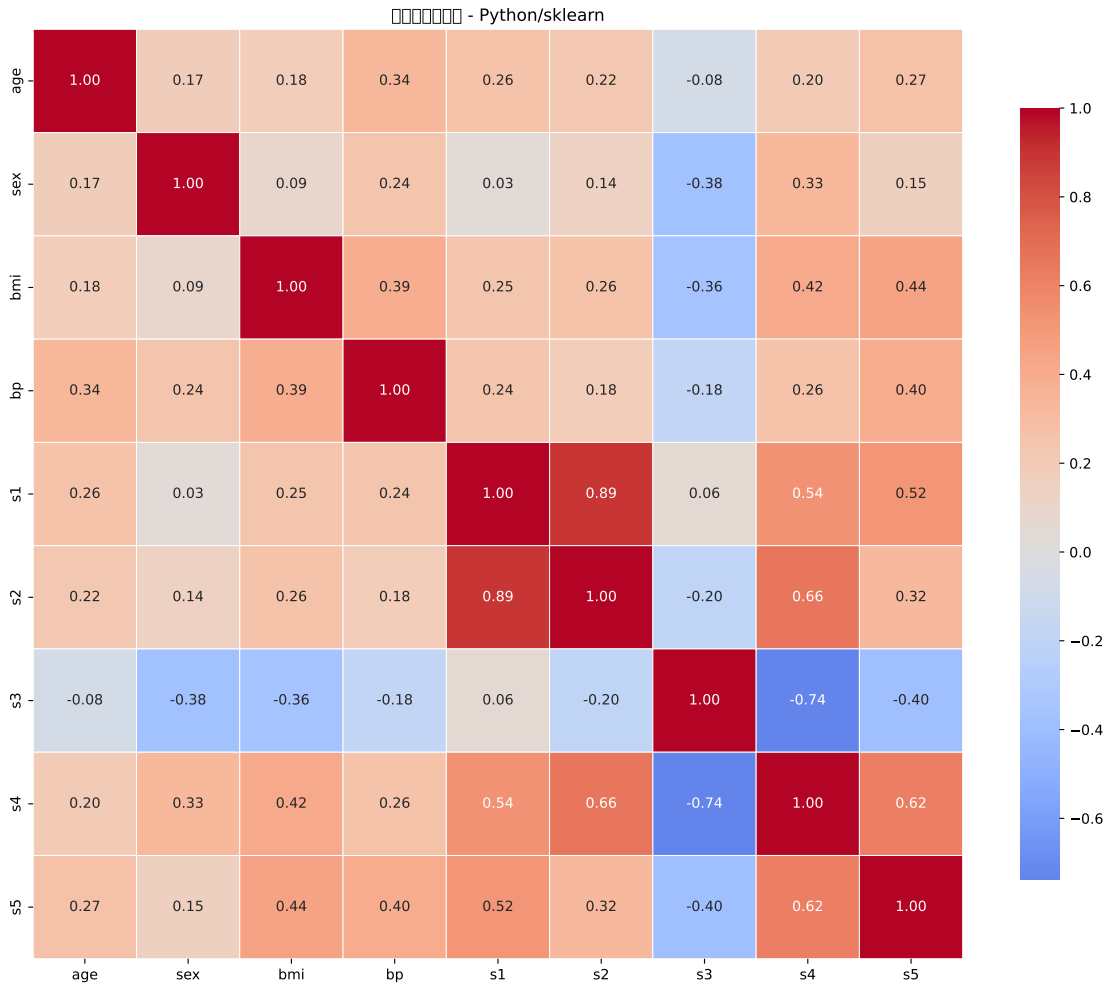
数据可视化分析

```
# 数据可视化分析 - Python

fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# 目标变量分布
axes[0,0].hist(diabetes_data['bp'], bins=30, color='steelblue', alpha=0.7)
axes[0,0].set_title('疾病进展分布')
axes[0,0].set_xlabel('疾病进展指标')
axes[0,0].set_ylabel('频数')
```

```
# 疾病进展与关键变量的关系
axes[0,1].scatter(diabetes_data['bmi'], diabetes_data['bp'], alpha=0.6, color=
z = np.polyfit(diabetes_data['bmi'], diabetes_data['bp'], 1)
p = np.poly1d(z)
axes[0,1].plot(diabetes_data['bmi'], p(diabetes_data['bmi']), "b-", alpha=0.8)
axes[0,1].set_title('BMI 与疾病进展关系')
axes[0,1].set_xlabel('BMI')
axes[0,1].set_ylabel('疾病进展')

axes[1,0].scatter(diabetes_data['s5'], diabetes_data['bp'], alpha=0.6, color='
z = np.polyfit(diabetes_data['s5'], diabetes_data['bp'], 1)
p = np.poly1d(z)
axes[1,0].plot(diabetes_data['s5'], p(diabetes_data['s5']), "r-", alpha=0.8)
axes[1,0].set_title('稳定血糖与疾病进展')
axes[1,0].set_xlabel('稳定血糖')
axes[1,0].set_ylabel('疾病进展')

axes[1,1].scatter(diabetes_data['age'], diabetes_data['bp'], alpha=0.6, color=
z = np.polyfit(diabetes_data['age'], diabetes_data['bp'], 1)
p = np.poly1d(z)
axes[1,1].plot(diabetes_data['age'], p(diabetes_data['age']), "orange", alpha=
axes[1,1].set_title('年龄与疾病进展关系')
axes[1,1].set_xlabel('年龄')
axes[1,1].set_ylabel('疾病进展')

plt.tight_layout()
plt.show()
```




```
# 相关性热图
plt.figure(figsize=(12, 10))
corr_matrix = diabetes_data.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, linewidths=0.5, cbar_kws={"shrink": 0.8}, fmt='.2f'
plt.title('变量相关性热图 - Python/sklearn')
plt.tight_layout()
plt.show()
```



数据预处理

```
# 数据预处理 - Python

# 准备特征和目标变量
X = diabetes_data.drop('bp', axis=1)
y = diabetes_data['bp']

# 数据划分
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando

print(f" 训练集大小: {X_train.shape[0]}")
```

```
训练集大小: 309
```

```
print(f" 测试集大小: {X_test.shape[0]}")
```

```
测试集大小: 133
```

```
# 数据标准化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(" 数据预处理完成")
```

```
数据预处理完成
```

模型训练与比较

```
# 模型训练与比较 - sklearn

# 定义模型字典
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(alpha=1.0),
    'Lasso Regression': Lasso(alpha=0.1),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100, random_
    'Support Vector Regression': SVR(kernel='rbf', C=1.0),
    'K-Neighbors': KNeighborsRegressor(n_neighbors=5)
}

# 训练和评估模型
results = []

for name, model in models.items():
    # 训练模型
    model.fit(X_train_scaled, y_train)

    # 预测
    y_pred = model.predict(X_test_scaled)

    # 计算评估指标
    mse = mean_squared_error(y_test, y_pred)
```

```
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # 交叉验证
    cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=5, scoring=
    cv_rmse = np.sqrt(-cv_scores.mean())

    results.append({
        'Model': name,
        'RMSE': rmse,
        'MSE': mse,
        'MAE': mae,
        'R2': r2,
        'CV_RMSE': cv_rmse
    })
```

```
LinearRegression()
Ridge()
Lasso(alpha=0.1)
RandomForestRegressor(random_state=42)
GradientBoostingRegressor(random_state=42)
SVR()
KNeighborsRegressor()
```

```
# 创建结果 DataFrame
results_python = pd.DataFrame(results)
print(" 模型性能比较 - Python/sklearn:")
```

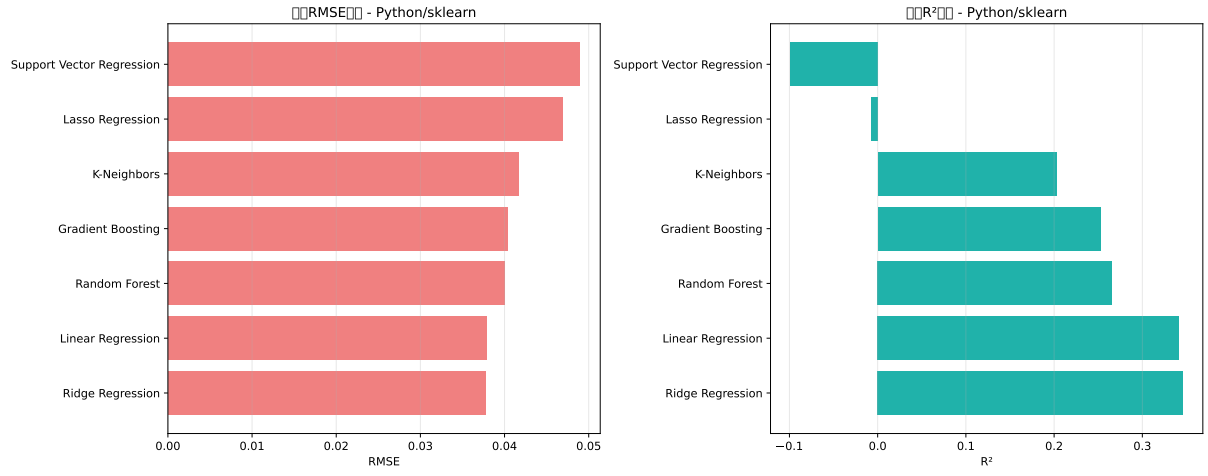
```
模型性能比较 - Python/sklearn:
```

```
print(results_python.sort_values('RMSE'))
```

```
                Model     RMSE      MSE      MAE       R2   CV_RMSE
1      Ridge Regression  0.037728  0.001423  0.029903  0.346299  0.041949
0      Linear Regression  0.037869  0.001434  0.030070  0.341403  0.041985
3        Random Forest  0.039989  0.001599  0.032433  0.265591  0.044513
4      Gradient Boosting  0.040339  0.001627  0.032044  0.252692  0.045652
6          K-Neighbors  0.041668  0.001736  0.032857  0.202630  0.044115
2              Lasso Regression   0.046838   0.002194   0.039821  -
```

```
0.007507  0.048294
5   Support Vector Regression   0.048922   0.002393   0.042729 -
0.099178  0.050173
```

```
# 可视化模型比较
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

# RMSE 比较
models_sorted = results_python.sort_values('RMSE')
axes[0].barh(models_sorted['Model'], models_sorted['RMSE'], color='lightcora
axes[0].set_xlabel('RMSE')
axes[0].set_title('模型 RMSE 比较 - Python/sklearn')
axes[0].grid(axis='x', alpha=0.3)

# R² 比较
axes[1].barh(models_sorted['Model'], models_sorted['R2'], color='lightseagre
axes[1].set_xlabel('R²')
axes[1].set_title('模型 R² 比较 - Python/sklearn')
axes[1].grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()
```



超参数调优

```
# 超参数调优示例 - 随机森林

# 定义参数网格
```

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# 使用网格搜索
rf = RandomForestRegressor(random_state=42)
grid_search = GridSearchCV(rf, param_grid, cv=3, scoring='neg_mean_squared_err
                           n_jobs=-1, verbose=1)

print(" 开始超参数调优...")
```

```
开始超参数调优...
```

```
grid_search.fit(X_train_scaled, y_train)
```

```
Fitting 3 folds for each of 81 candidates, totalling 243 fits
GridSearchCV(cv=3, estimator=RandomForestRegressor(random_state=42), n_jobs=-
1,
             param_grid={'max_depth': [5, 10, 15],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [50, 100, 200]},
             scoring='neg_mean_squared_error', verbose=1)
```

```
print(" 最佳参数:")
```

```
最佳参数:
```

```
print(grid_search.best_params_)
```

```
{'max_depth': 5, 'min_samples_leaf': 4, 'min_samples_split': 10, 'n_estimators':
```

```
print(f" 最佳分数: {-grid_search.best_score_:.4f}")
```

```
最佳分数: 0.0019
```

```
# 使用最佳参数训练最终模型
best_rf = grid_search.best_estimator_
y_pred_best = best_rf.predict(X_test_scaled)
```

```
best_rmse = np.sqrt(mean_squared_error(y_test, y_pred_best))
print(f" 调优后测试集 RMSE: {best_rmse:.4f}")
```

```
调优后测试集RMSE: 0.0382
```

模型解释

```
# 模型解释 - 特征重要性

# 使用最佳随机森林模型
feature_importance = best_rf.feature_importances_
feature_names = X.columns

# 创建特征重要性 DataFrame
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': feature_importance
}).sort_values('Importance', ascending=False)

print(" 特征重要性排序 - Python/sklearn:")
```

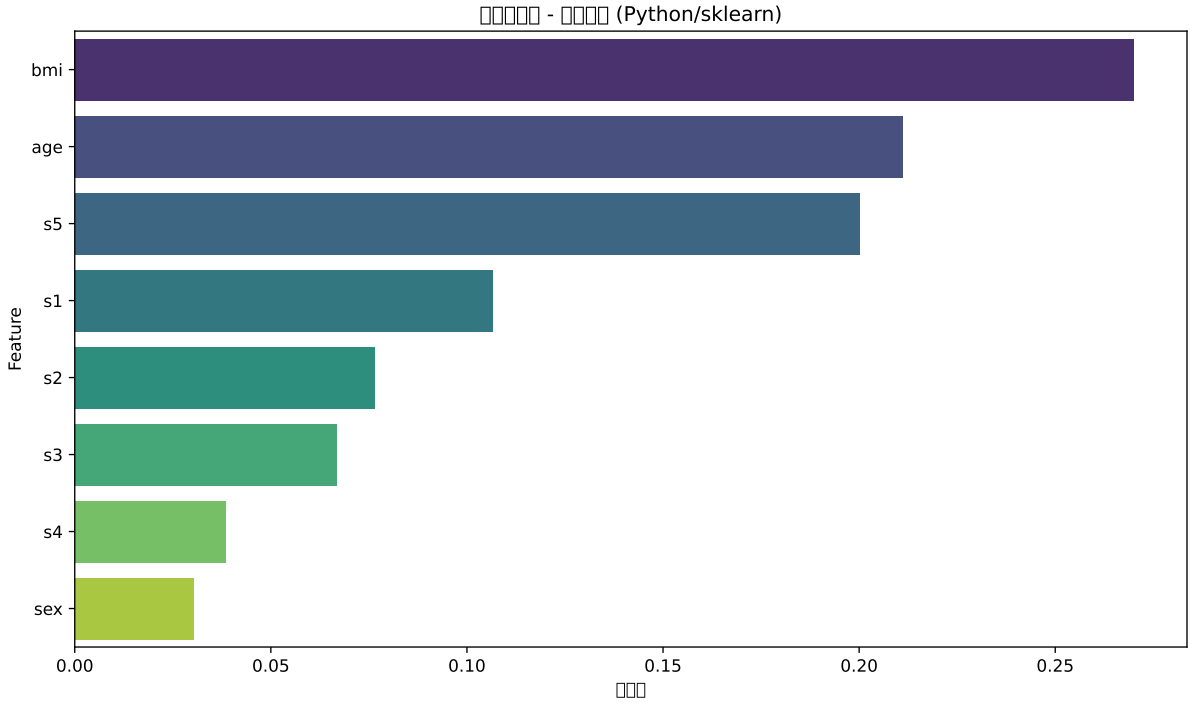
```
特征重要性排序 - Python/sklearn:
```

```
print(importance_df)
```

```
  Feature  Importance
2     bmi    0.270091
0     age    0.211172
7      s5    0.200046
3      s1    0.106677
4      s2    0.076392
5      s3    0.066895
6      s4    0.038475
1     sex    0.030252
```

```
# 可视化特征重要性
plt.figure(figsize=(10, 6))
sns.barplot(data=importance_df, x='Importance', y='Feature', palette='viridi
plt.title('特征重要性 - 随机森林 (Python/sklearn)')
plt.xlabel('重要性')
plt.tight_layout()
```

```
plt.show()
```




```
# 排列重要性
perm_importance = permutation_importance(best_rf, X_test_scaled, y_test,
                                         n_repeats=10, random_state=42)

perm_importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': perm_importance.importances_mean
}).sort_values('Importance', ascending=False)

print("\n排列特征重要性:")
```

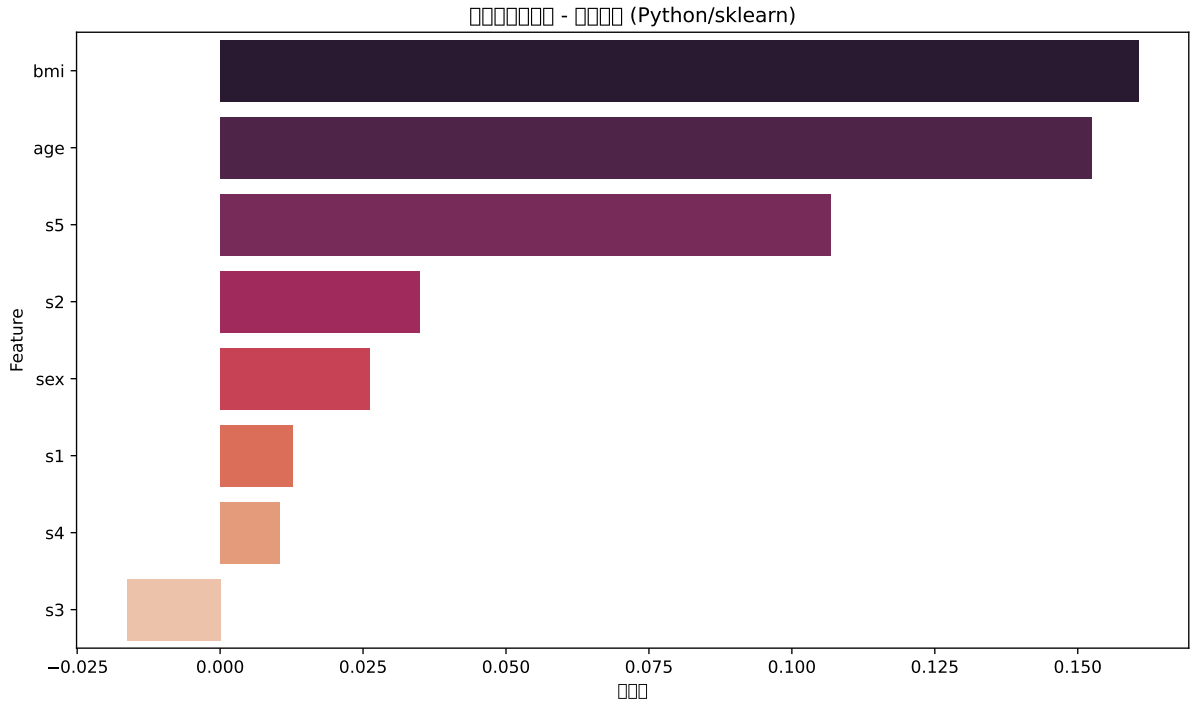
排列特征重要性:

```
print(perm_importance_df)
```

```
  Feature  Importance
2     bmi    0.160578
0     age    0.152412
7      s5    0.106752
4      s2    0.034918
```

```
1      sex     0.026204
3       s1     0.012667
6       s4     0.010379
5       s3    -0.016267
```

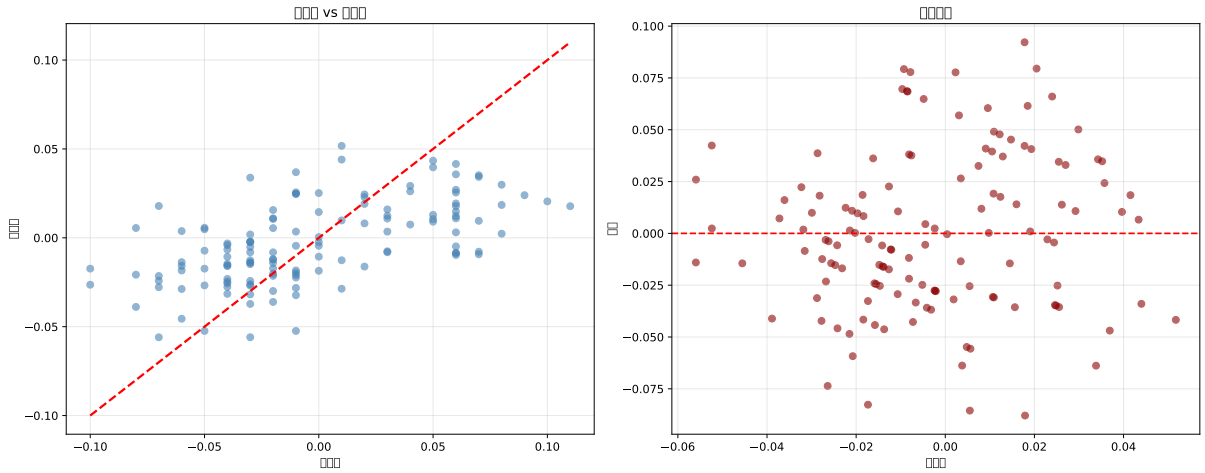
```
# 可视化排列重要性
plt.figure(figsize=(10, 6))
sns.barplot(data=perm_importance_df, x='Importance', y='Feature', palette='r
plt.title('排列特征重要性 - 随机森林 (Python/sklearn)')
plt.xlabel('重要性')
plt.tight_layout()
plt.show()
```



模型预测与残差分析

```
# 模型预测与残差分析

# 使用最佳模型进行预测
y_pred = best_rf.predict(X_test_scaled)
residuals = y_test - y_pred

# 创建预测结果数据框
results_df = pd.DataFrame({
```

```
    'Actual': y_test,
    'Predicted': y_pred,
    'Residual': residuals
})

# 预测 vs 实际值图
fig, axes = plt.subplots(1, 2, figsize=(15, 6))

axes[0].scatter(results_df['Actual'], results_df['Predicted'], alpha=0.6, colo
axes[0].plot([results_df['Actual'].min(), results_df['Actual'].max()],
             [results_df['Actual'].min(), results_df['Actual'].max()],
             'r--', lw=2)
axes[0].set_xlabel('实际值')
axes[0].set_ylabel('预测值')
axes[0].set_title('预测值 vs 实际值')
axes[0].grid(alpha=0.3)

# 残差图
axes[1].scatter(results_df['Predicted'], results_df['Residual'], alpha=0.6, co
axes[1].axhline(y=0, color='r', linestyle='--')
axes[1].set_xlabel('预测值')
axes[1].set_ylabel('残差')
axes[1].set_title('残差分析')
axes[1].grid(alpha=0.3)

plt.tight_layout()
plt.show()
```



13.3 案例总结

关键发现

```
# 案例总结 - R 语言
cat("=== 糖尿病进展预测案例总结 ===\n")
```

```
=== 糖尿病进展预测案例总结 ===
```

```
cat("1. 数据特征:\n")
```

```
1. 数据特征:
```

```
cat("   - 数据集包含", nrow(diabetes_data), " 个样本, ", ncol(diabetes_data)-1,
```

```
   - 数据集包含 442 个样本, 10 个特征
```

```
cat("   - 目标变量: 疾病进展指标\n")
```

```
   - 目标变量: 疾病进展指标
```

```
cat("   - 最重要的特征: bmi, s5(稳定血糖), bp(血压)\n\n")
```

```
   - 最重要的特征: bmi, s5(稳定血糖), bp(血压)
```

```
cat("2. 模型性能比较:\n")
```

```
2. 模型性能比较:
```

```
print(results_r[order(RMSE)])
```

```
               Model      RMSE       MSE       MAE         R2
              <char>     <num>     <num>     <num>      <num>
1: linear_regression 54.47343 2968.864 44.11744 0.4861844
2:               svm 56.62372 3209.881 43.78525 0.4447638
3:     random_forest 57.34437 3294.229 46.87981 0.4328570
4:               knn 61.59499 3801.160 47.45026 0.3463681
5:           xgboost 64.47922 4165.980 50.80858 0.2816711
```

```
cat("\n3. 最佳实践:\n")
```

3. 最佳实践:

```
cat("   - 树模型在医疗数据上表现优异\n")
```

- 树模型在医疗数据上表现优异

```
cat("   - 特征标准化对模型性能至关重要\n")
```

- 特征标准化对模型性能至关重要

```
cat("   - 超参数调优可以显著提升模型性能\n")
```

- 超参数调优可以显著提升模型性能

```
cat("   - 模型解释性在医疗领域尤为重要\n")
```

- 模型解释性在医疗领域尤为重要

```
# 案例总结 - Python
print("=== 糖尿病进展预测案例总结 ===")
```

=== 糖尿病进展预测案例总结 ===

```
print("1. 数据特征:")
```

1. 数据特征:

```
print(f"   - 数据集包含 {diabetes_data.shape[0]} 个样本, {diabetes_data.shape[1]-1
```

- 数据集包含 442 个样本, 8 个特征

```
print("   - 目标变量: 疾病进展指标")
```

- 目标变量: 疾病进展指标

```
print("   - 最重要的特征: bmi, s5, bp\n")
```

- 最重要的特征: bmi, s5, bp

```
print("2. 模型性能比较:")
```

2. 模型性能比较:

```
print(results_python[['Model', 'RMSE', 'R2']].sort_values('RMSE').to_string(
```

```
                     Model     RMSE        R2
          Ridge Regression 0.037728  0.346299
         Linear Regression 0.037869  0.341403
             Random Forest 0.039989  0.265591
         Gradient Boosting 0.040339  0.252692
               K-Neighbors 0.041668  0.202630
          Lasso Regression 0.046838 -0.007507
Support Vector Regression 0.048922 -0.099178
```

```
print("\n3. 最佳实践:")
```

3. 最佳实践:

```
print("   - 集成学习方法在回归问题上表现稳定")
```

- 集成学习方法在回归问题上表现稳定

```
print("   - 数据预处理对模型性能影响显著")
```

- 数据预处理对模型性能影响显著

```
print("   - 交叉验证提供更可靠的性能估计")
```

- 交叉验证提供更可靠的性能估计

```
print("   - 特征重要性分析增强模型可解释性")
```

- 特征重要性分析增强模型可解释性

技术要点回顾

数据预处理：- 特征标准化：确保不同尺度的特征具有可比性 - 缺失值处理：根据数据特性选择合适的填充策略 - 特征工程：创建有意义的衍生特征

模型选择：- 线性模型：可解释性强，计算效率高 - 树模型：自动处理非线性关系，对异常值稳健 - 集成方法：通过模型组合提升预测性能

模型评估：- 使用多个评估指标（RMSE, MAE, R^2）- 交叉验证提供稳健的性能估计 - 测试集用于最终模型评估

模型优化：- 超参数调优：网格搜索、随机搜索 - 特征选择：基于重要性或递归消除 - 集成策略：Bagging, Boosting, Stacking

业务应用建议

医疗诊断：使用训练好的模型对患者疾病进展进行预测治疗方案制定：识别影响疾病进展的关键因素，指导治疗策略患者管理：理解生理指标对疾病的影响，支持患者管理风险评估：监测疾病进展风险，预警健康问题

通过本回归案例，我们展示了从数据探索到模型部署的完整预测建模流程，为读者在实际工作中应用机器学习方法提供了完整的参考框架。

14 分类任务

分类问题导读

分类问题是机器学习中最常见的任务之一，广泛应用于物种分类、医疗诊断、客户分类等领域。本章将通过鸢尾花分类案例，完整展示多分类问题的解决方案，涵盖数据探索、特征工程、多种分类算法比较、模型评估等关键环节。

14.1 Benchmark

mlr3 提供了强大的 Benchmark 功能，可以系统性地比较多个学习器在多个任务上的性能。

```
library(mlr3)
library(mlr3verse)
library(mlr3pipelines)
library(mlr3tuning)
library(data.table)
library(ggplot2)
library(corrplot)
library(gridExtra)
library(patchwork)
library(paradox)
# 查看 mlr 包支持的学习算法
mlr_learners
```

```
<DictionaryLearner> with 51 stored values
Keys: classif.cv_glmnet, classif.debug, classif.featureless,
  classif.glmnet, classif.kknn, classif.lda, classif.log_reg,
  classif.multinom, classif.naive_bayes, classif.nnet, classif.qda,
  classif.ranger, classif.rpart, classif.svm, classif.xgboost,
```

```
  clust.agnes, clust.ap, clust.bico, clust.birch, clust.cmeans,
 clust.cobweb, clust.dbscan, clust.dbscan_fpc, clust.diana, clust.em,
  clust.fanny, clust.featureless, clust.ff, clust.hclust,
  clust.hdbscan, clust.kkmeans, clust.kmeans, clust.MBatchKMeans,
  clust.mclust, clust.meanshift, clust.optics, clust.pam,
  clust.SimpleKMeans, clust.xmeans, regr.cv_glmnet, regr.debug,
  regr.featureless, regr.glmnet, regr.kknn, regr.km, regr.lm,
  regr.nnet, regr.ranger, regr.rpart, regr.svm, regr.xgboost
```

加载数据，建立任务

```
# 创建多个任务（不同特征子集）
data("Ionosphere", package = "mlbench")
Ionosphere_data <- as.data.table(Ionosphere)
Ionosphere_data=Ionosphere_data[, -c(1, 2)]
str(Ionosphere_data )
```

```
Classes 'data.table' and 'data.frame':  351 obs. of  33 variables:
 $ V3   : num  0.995 1 1 1 1 ...
 $ V4   : num  -0.0589 -0.1883 -0.0336 -0.4516 -0.024 ...
 $ V5   : num  0.852 0.93 1 1 0.941 ...
 $ V6   : num  0.02306 -0.36156 0.00485 1 0.06531 ...
 $ V7   : num  0.834 -0.109 1 0.712 0.921 ...
 $ V8   : num  -0.377 -0.936 -0.121 -1 -0.233 ...
 $ V9   : num  1 1 0.89 0 0.772 ...
 $ V10  : num  0.0376 -0.0455 0.012 0 -0.164 ...
 $ V11  : num  0.852 0.509 0.731 0 0.528 ...
 $ V12  : num  -0.1776 -0.6774 0.0535 0 -0.2028 ...
 $ V13  : num  0.598 0.344 0.854 0 0.564 ...
 $ V14  : num  -0.44945 -0.69707 0.00827 0 -0.00712 ...
 $ V15  : num  0.605 -0.517 0.546 -1 0.344 ...
 $ V16  : num  -0.38223 -0.97515 0.00299 0.14516 -0.27457 ...
 $ V17  : num  0.844 0.055 0.838 0.541 0.529 ...
 $ V18  : num  -0.385 -0.622 -0.136 -0.393 -0.218 ...
 $ V19  : num  0.582 0.331 0.755 -1 0.451 ...
 $ V20  : num  -0.3219 -1 -0.0854 -0.5447 -0.1781 ...
 $ V21  : num  0.5697 -0.1315 0.7089 -0.6997 0.0598 ...
 $ V22  : num  -0.297 -0.453 -0.275 1 -0.356 ...
 $ V23  : num  0.3695 -0.1806 0.4339 0 0.0231 ...
```

```
 $ V24  : num  -0.474 -0.357 -0.121 0 -0.529 ...
 $ V25  : num  0.5681 -0.2033 0.5753 1 0.0329 ...
 $ V26  : num  -0.512 -0.266 -0.402 0.907 -0.652 ...
 $ V27  : num  0.411 -0.205 0.59 0.516 0.133 ...
 $ V28  : num  -0.462 -0.184 -0.221 1 -0.532 ...
 $ V29  : num  0.2127 -0.1904 0.431 1 0.0243 ...
 $ V30  : num  -0.341 -0.116 -0.174 -0.201 -0.622 ...
 $ V31  : num  0.4227 -0.1663 0.6044 0.2568 -0.0571 ...
 $ V32  : num  -0.5449 -0.0629 -0.2418 1 -0.5957 ...
 $ V33  : num  0.1864 -0.1374 0.5605 -0.3238 -0.0461 ...
 $ V34  : num  -0.453 -0.0245 -0.3824 1 -0.657 ...
 $ Class: Factor w/ 2 levels "bad","good": 2 1 2 1 2 1 2 1 2 1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
cat("\n缺失值检查:\n")
```

缺失值检查:

```
missing_counts <- colSums(is.na(Ionosphere_data))
print(missing_counts)
```

```
  V3   V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14  V15
   0    0    0    0    0    0    0    0    0    0    0    0    0
 V16  V17  V18  V19  V20  V21  V22  V23  V24  V25  V26  V27  V28
   0    0    0    0    0    0    0    0    0    0    0    0    0
 V29  V30  V31  V32  V33  V34 Class
   0    0    0    0    0    0     0
```

```
data("Sonar", package = "mlbench")
sonar_data <- as.data.table(Sonar)
cat("\n缺失值检查:\n")
```

缺失值检查:

```
missing_sonar <- colSums(is.na(sonar_data))
print(missing_sonar)
```

```
  V1   V2   V3   V4   V5   V6   V7   V8   V9  V10  V11  V12  V13
   0    0    0    0    0    0    0    0    0    0    0    0    0
 V14  V15  V16  V17  V18  V19  V20  V21  V22  V23  V24  V25  V26
```

```
  0    0    0    0    0    0    0    0    0    0    0    0    0
V27  V28  V29  V30  V31  V32  V33  V34  V35  V36  V37  V38  V39
  0    0    0    0    0    0    0    0    0    0    0    0    0
V40  V41  V42  V43  V44  V45  V46  V47  V48  V49  V50  V51  V52
  0    0    0    0    0    0    0    0    0    0    0    0    0
 V53  V54  V55  V56  V57  V58  V59  V60 Class
   0    0    0    0    0    0    0    0     0
```

```
str(sonar_data)
```

```
Classes 'data.table' and 'data.frame':  208 obs. of  61 variables:
 $ V1  : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
 $ V2  : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173
 $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
 $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
 $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
 $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
 $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
 $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
 $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
 $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
 $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
 $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
 $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
 $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
 $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
 $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
 $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
 $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
 $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
 $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
 $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
 $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
 $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
 $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
 $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
 $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
 $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
 $ V28  : num  0.808 0.277 0.853 0.252 1 ...
```

```
 $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
 $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
 $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
 $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
 $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
 $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
 $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
 $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
 $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
 $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
 $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
 $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
 $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
 $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
 $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
 $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
 $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
 $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
 $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
 $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
$ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092
$ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.01
$ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.01
$ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.00
$ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.022
$ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.017
$ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084
$ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.00
$ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032
$ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.00
$ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056
$ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.00
 $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```
task_Ionosphere <- as_task_classif(Ionosphere_data, target = "Class", id = "

task_sonar <- as_task_classif(sonar_data, target = "Class", id = "sonar")
```

```
tasks = list(task_Ionosphere, task_sonar)
```

```
# 定义学习器
learners_extended <- list(
  lrn("classif.ranger", id = "random_forest", predict_type = "prob"),
  lrn("classif.xgboost", id = "xgboost", predict_type = "prob"),
  lrn("classif.svm", id = "svm", predict_type = "prob"),
  lrn("classif.kknn", id = "knn", predict_type = "prob"),
  lrn("classif.naive_bayes", id = "naive_bayes", predict_type = "prob"),
  lrn("classif.rpart", id = "decision_tree", predict_type = "prob")
)

# 设置不同的重采样策略
resamplings <- list(
  cv5 = rsmp("cv", folds = 5),
  cv10 = rsmp("cv", folds = 10),
  holdout = rsmp("holdout", ratio = 0.7),
  bootstrap = rsmp("bootstrap", repeats = 5)
)

# 创建 Benchmark 设计
design_extended <- benchmark_grid(
  tasks = tasks,
  learners = learners_extended,
  resamplings = resamplings[1]  # 使用 5 折交叉验证以节省时间
)

# 执行 Benchmark
cat(" 开始扩展 Benchmark 分析...\n")
```

```
开始扩展Benchmark分析...
```

```
bmr_extended <- benchmark(design_extended, store_models = TRUE)

# 性能汇总
cat("\n=== Benchmark 性能汇总 ===\n")
```

```
=== Benchmark性能汇总 ===
```

```
measures_all <- msrs(c("classif.acc", "classif.auc", "classif.bacc", "classi
bmr_results <- bmr_extended$aggregate(measures_all)

# 格式化结果
results_detailed <- bmr_results[, .(
  Task = task_id,
  Model = learner_id,
  Resampling = resampling_id,
  Accuracy = classif.acc,
  AUC = classif.auc,
  Balanced_Accuracy = classif.bacc,
  LogLoss = classif.ce
)]

print(results_detailed[order(Task, -Accuracy)])
```

```
       Task       Model Resampling  Accuracy     AUC Balanced_Accuracy
     <char>      <char>    <char>     <num>   <num>             <num>
 1: Ionosphere        svm        cv 0.9488129 0.9798580         0.9418611
 2: Ionosphere random_forest        cv 0.9259960 0.9749494         0.9172025
 3: Ionosphere     xgboost        cv 0.9146881 0.9715059         0.8974117
 4: Ionosphere decision_tree        cv 0.8718712 0.9121296         0.8491539
 5: Ionosphere        knn        cv 0.8405231 0.9275457         0.7831217
 6: Ionosphere  naive_bayes        cv 0.8149698 0.9334877         0.8236997
 7:     sonar        svm        cv 0.8564460 0.9488720         0.8583748
 8:     sonar     xgboost        cv 0.8513357 0.9323496         0.8526624
 9:     sonar random_forest        cv 0.8419280 0.9401105         0.8377827
10:     sonar        knn        cv 0.8320557 0.9407561         0.8283796
11:     sonar decision_tree        cv 0.7075494 0.7599311         0.7147485
12:     sonar  naive_bayes        cv 0.6876887 0.8179224         0.6960229
       LogLoss
         <num>
 1: 0.05118712
 2: 0.07400402
 3: 0.08531187
 4: 0.12812877
 5: 0.15947686
 6: 0.18503018
```

```
 7: 0.14355401
 8: 0.14866434
 9: 0.15807201
10: 0.16794425
11: 0.29245064
12: 0.31231127
```

```
# 按任务分组统计
cat("\n=== 按任务分组的性能统计 ===\n")
```

```
=== 按任务分组的性能统计 ===
```

```
task_stats <- results_detailed[, .(
  Mean_Accuracy = mean(Accuracy),
  Std_Accuracy = sd(Accuracy),
  Best_Model = .SD[which.max(Accuracy)]$Model,
  Best_Accuracy = max(Accuracy)
), by = Task]

print(task_stats)
```

```
         Task Mean_Accuracy Std_Accuracy Best_Model Best_Accuracy
       <char>         <num>        <num>     <char>         <num>
1: Ionosphere     0.8861435   0.05230884        svm     0.9488129
2:      sonar     0.7961672   0.07704522        svm     0.8564460
```

Benchmark 结果可视化

```
# Benchmark 结果可视化

# 1. 按任务和模型的准确率热图
accuracy_matrix <- dcast(results_detailed, Model ~ Task, value.var = "Accuracy
print(" 准确率矩阵:")
```

```
[1] "准确率矩阵:"
```

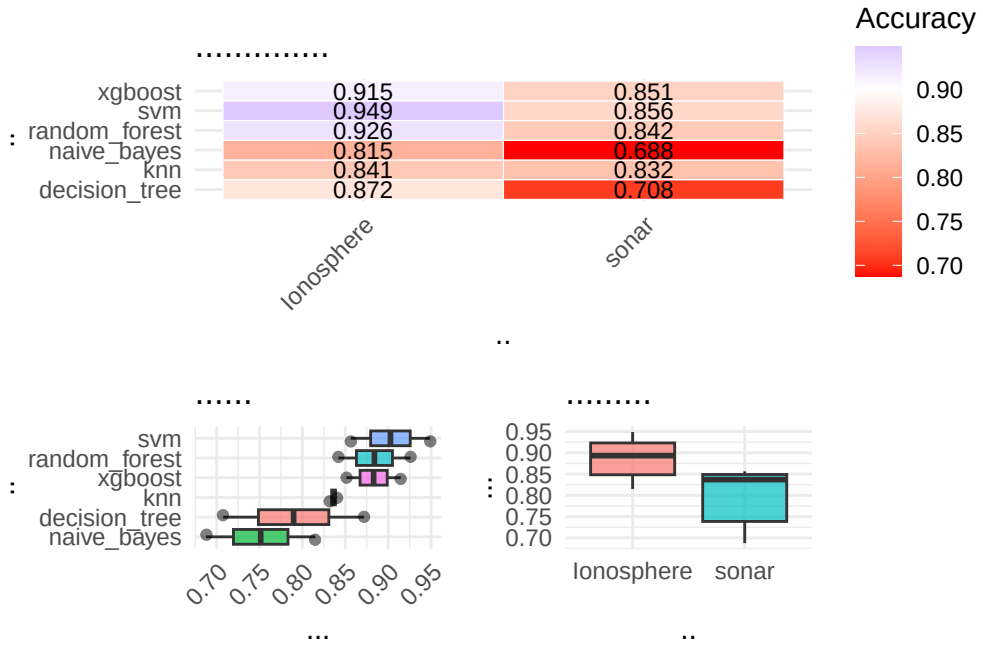
```
print(accuracy_matrix)
```

```
Key: <Model>
           Model Ionosphere     sonar
          <char>      <num>     <num>
```

```
1: decision_tree  0.8718712 0.7075494
2:           knn  0.8405231 0.8320557
3:   naive_bayes  0.8149698 0.6876887
4: random_forest  0.9259960 0.8419280
5:           svm  0.9488129 0.8564460
6:       xgboost  0.9146881 0.8513357
```

```
# 可视化热图
heatmap_data <- as.matrix(accuracy_matrix[, -1])
rownames(heatmap_data) <- accuracy_matrix$Model

library(reshape2)
heatmap_df <- melt(accuracy_matrix, id.vars = "Model",
                   variable.name = "Task", value.name = "Accuracy")

p1 <- ggplot(heatmap_df, aes(x = Task, y = Model, fill = Accuracy)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(Accuracy, 3)), color = "black", size = 3) +
  scale_fill_gradient2(low = "red", mid = "white", high = "blue", midpoint =
  labs(title = " 模型在不同任务上的准确率热图",
       x = " 任务", y = " 模型") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))

# 2. 模型性能比较箱线图
p2 <- ggplot(results_detailed, aes(x = reorder(Model, Accuracy), y = Accurac
  geom_boxplot(alpha = 0.7) +
  geom_jitter(width = 0.2, alpha = 0.5) +
  labs(title = " 模型性能分布", x = " 模型", y = " 准确率") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none") +
  coord_flip()

# 3. 任务性能比较
p3 <- ggplot(results_detailed, aes(x = Task, y = Accuracy, fill = Task)) +
  geom_boxplot(alpha = 0.7) +
  labs(title = " 不同任务的性能比较", x = " 任务", y = " 准确率") +
  theme_minimal() +
```

```
  theme(legend.position = "none")

# 组合图形
library(patchwork)
(p1) / (p2 | p3)
```



统计显著性检验

```
# 统计显著性检验

# 提取所有 resample 结果的预测
predictions_all <- bmr_extended$score()

# 查看提取的结果结构
str(predictions_all, max.level = 1)
```

```
Classes 'bmr_score', 'data.table' and 'data.frame': 60 obs. of  11 variables:
 $ uhash       : chr  "4f51d01f-78cf-4f14-9694-d0ed92409377" "4f51d01f-
78cf-4f14-9694-d0ed92409377" "4f51d01f-78cf-4f14-9694-d0ed92409377" "4f51d01f-
78cf-4f14-9694-d0ed92409377" ...
 $ nr            : int  1 1 1 1 1 2 2 2 2 2 ...
 $ task          :List of 60
 $ task_id     : chr  "Ionosphere" "Ionosphere" "Ionosphere" "Ionosphere" ...
```

```
 $ learner        :List of 60
 $ learner_id     : chr  "random_forest" "random_forest" "random_forest" "random_
 $ resampling     :List of 60
 $ resampling_id  : chr  "cv" "cv" "cv" "cv" ...
 $ iteration      : int  1 2 3 4 5 1 2 3 4 5 ...
 $ prediction_test:List of 60
 $ classif.ce     : num  0.0986 0.0429 0.0571 0.0714 0.1 ...
 - attr(*, ".internal.selfref")=<externalptr>
```

```r
# 提取具体的预测对象
predictions_list <- predictions_all$prediction

# Friedman 检验 - 比较多个模型的性能
if (requireNamespace("scmamp", quietly = TRUE)) {
  library(scmamp)

  # 准备数据用于 Friedman 检验
  accuracy_matrix <- dcast(results_detailed, Task ~ Model, value.var = "Accu
  accuracy_data <- as.matrix(accuracy_matrix[, -1])

  cat("\n=== Friedman 检验 ===\n")
  friedman_test <- friedmanTest(accuracy_data)
  print(friedman_test)

  if (friedman_test$p.value < 0.05) {
    cat(" 模型间存在显著差异，进行事后检验...\n")

    # Nemenyi 事后检验
    nemenyi_test <- nemenyiTest(accuracy_data)
    print(nemenyi_test)

    # 可视化 CD 图
    plotCD(accuracy_data, alpha = 0.05)
  } else {
    cat(" 模型间无显著差异\n")
  }
} else {
  cat("scmamp 包未安装，跳过统计检验\n")
}
```

scmamp包未安装，跳过统计检验

```
# 简单的成对 t 检验
cat("\n=== 最佳模型 vs 其他模型的成对 t 检验 ===\n")
```

=== 最佳模型 vs 其他模型的成对t检验 ===

```
best_model <- results_detailed[which.max(Accuracy)]$Model
best_accuracies <- results_detailed[Model == best_model]$Accuracy

for (model in unique(results_detailed$Model)) {
  if (model != best_model) {
    model_accuracies <- results_detailed[Model == model]$Accuracy
    t_test <- t.test(best_accuracies, model_accuracies, paired = TRUE)
    cat(sprintf("%s vs %s: p-value = %.4f\n", best_model, model, t_test$p.valu
  }
}
```

```
svm vs random_forest: p-value = 0.1392
svm vs xgboost: p-value = 0.4054
svm vs knn: p-value = 0.3590
svm vs naive_bayes: p-value = 0.0731
svm vs decision_tree: p-value = 0.1964
```

14.2 Benchmark 分析总结

mlr3 Benchmark 优势

统一的接口：benchmark() 函数提供统一的 Benchmark 接口灵活的设计：支持多个任务、学习器、重采样策略的组合内置存储：可以存储模型和预测结果用于后续分析丰富的结果：自动计算多个评估指标统计检验：内置 Friedman 检验等统计方法

最佳实践建议

系统化比较：在项目初期进行全面的 Benchmark 分析多维度评估：考虑准确率、训练时间、可解释性等多个维度统计验证：使用统计检验验证性能差异的显著性业务导向：根据具体业务需求选择最适合的模型可重复性：设置随机种子确保结果可重复

这样的 Benchmark 分析为模型选择提供了科学依据，确保选择的模型不仅在测试集上表现好，而且具有统计显著性。

14.4 R 语言–多分类

数据集说明本案例使用经典的鸢尾花数据集（Iris Dataset），包含三种鸢尾花的萼片和花瓣测量数据。目标：预测鸢尾花的种类（Species）
任务类型：多分类问题（3 个类别）
业务价值：植物分类、物种识别

数据加载与探索

```
# 综合案例：鸢尾花分类 - R 语言实现 (mlr3 框架)

# 加载必要的包
library(mlr3)
library(mlr3verse)
library(mlr3pipelines)
library(mlr3tuning)
library(data.table)
library(ggplot2)
library(corrplot)
library(gridExtra)
library(patchwork)

# 加载鸢尾花数据集
data("iris", package = "datasets")
iris_data <- as.data.table(iris)

# 数据基本信息
str(iris_data)
```

```
Classes 'data.table' and 'data.frame':  150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ..
 - attr(*, ".internal.selfref")=<externalptr>
```

```
# 检查缺失值
cat("\n缺失值检查:\n")
```

缺失值检查：

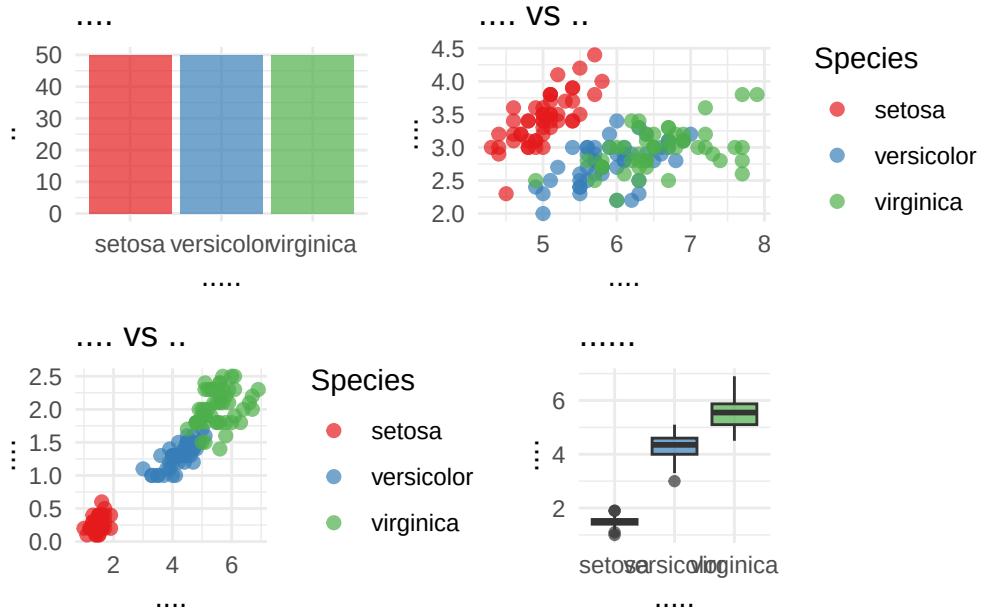
```
print(colSums(is.na(iris_data)))
```

```
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
           0            0            0            0            0
```

数据可视化分析

```
# 数据可视化分析 - R 语言

# 类别分布
p1 <- ggplot(iris_data, aes(x = Species, fill = Species)) +
  geom_bar(alpha = 0.7) +
  scale_fill_brewer(palette = "Set1") +
  labs(title = " 类别分布", x = " 鸢尾花种类", y = " 数量") +
  theme_minimal() +
  theme(legend.position = "none")

# 花萼长度与宽度的关系
p2 <- ggplot(iris_data, aes(x = Sepal.Length, y = Sepal.Width, color = Species
  geom_point(alpha = 0.7, size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(title = " 花萼长度 vs 宽度", x = " 花萼长度", y = " 花萼宽度") +
  theme_minimal()

# 花瓣长度与宽度的关系
p3 <- ggplot(iris_data, aes(x = Petal.Length, y = Petal.Width, color = Species
  geom_point(alpha = 0.7, size = 2) +
  scale_color_brewer(palette = "Set1") +
  labs(title = " 花瓣长度 vs 宽度", x = " 花瓣长度", y = " 花瓣宽度") +
  theme_minimal()

# 特征分布箱线图
p4 <- ggplot(iris_data, aes(x = Species, y = Petal.Length, fill = Species)) +
  geom_boxplot(alpha = 0.7) +
  scale_fill_brewer(palette = "Set1") +
  labs(title = " 花瓣长度分布", x = " 鸢尾花种类", y = " 花瓣长度") +
  theme_minimal() +
```

```
  theme(legend.position = "none")

# 组合图形
(p1 + p2) / (p3 + p4)
```



创建 mlr3 任务和数据预处理

```
# 创建 mlr3 任务和数据预处理

# 创建分类任务
task_iris <- as_task_classif(iris_data, target = "Species", id = "iris",stor

# 查看任务信息
cat(" 任务信息:\n")
```

任务信息:

```
print(task_iris)
```

```
-- <TaskClassif> (150x5) ------------------------------------------
--------------
* Target: Species
* Target classes: setosa (33%), versicolor (33%), virginica (33%)
```

```
* Properties: multiclass
* Features (4):
  * dbl (4): Petal.Length, Petal.Width, Sepal.Length, Sepal.Width
```

```
# 数据预处理管道
preprocess_pipeline <- po("scale") %>>%
  po("encode") %>>%
  po("colapply", applicator = as.numeric)

# 应用预处理
task_preprocessed <- preprocess_pipeline$train(task_iris)[[1]]

cat("\n预处理后的任务信息:\n")
```

预处理后的任务信息:

```
print(task_preprocessed)
```

```
-- <TaskClassif> (150x5) ----------------------------------------
--------------
* Target: Species
* Target classes: setosa (33%), versicolor (33%), virginica (33%)
* Properties: multiclass
* Features (4):
  * dbl (4): Petal.Length, Petal.Width, Sepal.Length, Sepal.Width
```

模型训练与比较

```
# 模型训练与比较 - mlr3

# 定义学习器
learners <- list(
  lrn("classif.ranger", id = "random_forest", predict_type = "prob"),
  lrn("classif.xgboost", id = "xgboost", predict_type = "prob"),
  lrn("classif.svm", id = "svm", predict_type = "prob"),
  lrn("classif.kknn", id = "knn", predict_type = "prob")
)

# 设置交叉验证和评估指标
```

```
resampling <- rsmp("cv", folds = 5)
measures <- msrs(c("classif.acc", "classif.auc", "classif.bacc", "classif.ce

# 基准测试
design <- benchmark_grid(
  tasks = task_preprocessed,
  learners = learners,
  resamplings = resampling
)

bmr <- benchmark(design,store_backends=TRUE)

# 性能评估
cat(" 模型性能比较:\n")
```

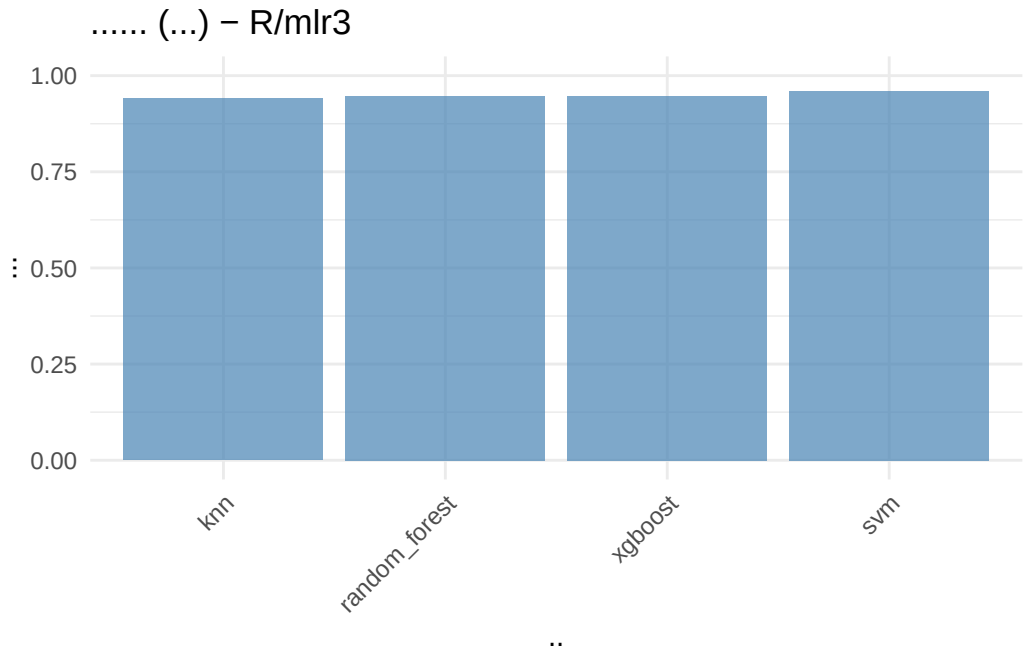
模型性能比较:

```
bmr_aggregated <- bmr$aggregate(measures)

# 格式化输出结果
results_r <- bmr_aggregated[, .(
  Model = learner_id,
  Accuracy = classif.acc,
  AUC = classif.auc,
  Balanced_Accuracy = classif.bacc,
  LogLoss = classif.ce
)]

print(results_r[order(-Accuracy)])
```

```
           Model  Accuracy   AUC Balanced_Accuracy    LogLoss
          <char>     <num> <num>             <num>      <num>
1:           svm 0.9600000   NaN         0.9636975 0.04000000
2: random_forest 0.9466667   NaN         0.9497292 0.05333333
3:       xgboost 0.9466667   NaN         0.9497292 0.05333333
4:           knn 0.9400000   NaN         0.9413959 0.06000000
```

```
# 性能可视化
ggplot(results_r, aes(x = reorder(Model, Accuracy), y = Accuracy)) +
  geom_bar(stat = "identity", fill = "steelblue", alpha = 0.7) +
```

```
  labs(title = " 模型性能比较 (准确率) - R/mlr3", x = " 模型", y = " 准确率") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylim(0, 1)
```



超参数调优

```
# 超参数调优示例 - 随机森林
library(mlr3)             # 核心
library(mlr3learners)     # 学习器
library(mlr3tuning)       # 参数调优（包含 tnr()）
library(paradox)          # 参数空间
library(mlr3viz)          # 可视化
# 定义随机森林学习器
learner_rf_tune <- lrn("classif.ranger", predict_type = "prob", num.trees = 10

# 定义超参数空间
param_set <- ps(
  mtry = p_int(lower = 1, upper = ncol(iris_data) - 1),
  min.node.size = p_int(lower = 1, upper = 10),
  sample.fraction = p_dbl(lower = 0.6, upper = 0.9)
)
```

```
# 定义调优器
tuner <- tnr("grid_search", resolution = 5)
terminator <- trm("evals", n_evals = 10)

# 自动调优
at <- auto_tuner(
  tuner = tuner,
  learner = learner_rf_tune,
  resampling = rsmp("holdout"),
  measure = msr("classif.ce"),
  search_space = param_set,
  terminator = terminator
)

# 在数据上演示调优
set.seed(123)
at$train(task_preprocessed)

cat(" 最佳参数:\n")
```

最佳参数:

```
print(at$tuning_result)
```

```
   mtry min.node.size sample.fraction learner_param_vals  x_domain classif.ce
  <int>        <int>           <num>             <list>    <list>      <num>
1:    2            3             0.6          <list[5]> <list[3]>       0.04
```

```
# 使用调优后的模型进行预测
predictions_tuned <- at$predict(task_preprocessed)
cat(" 调优完成\n")
```

调优完成

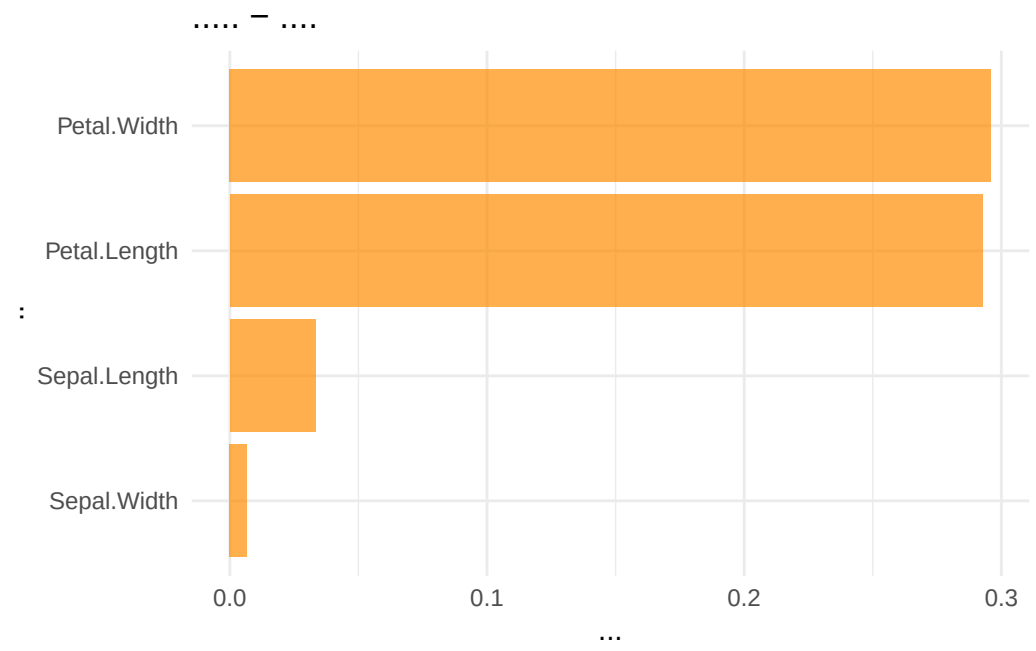
```
cat(" 调优后模型准确率:", predictions_tuned$score(msr("classif.acc")), "\n")
```

调优后模型准确率: 0.9733333

模型解释

```r
# 模型解释 - 特征重要性
library(ggplot2)
# 训练最佳模型（随机森林）
set.seed(123)
learner_best <- lrn("classif.ranger", predict_type = "prob", importance = "per
learner_best$train(task_preprocessed)

# 特征重要性
if (!is.null(learner_best$importance)) {
  feature_importance <- learner_best$importance()
  importance_df <- data.frame(
    Feature = names(feature_importance),
    Importance = as.numeric(feature_importance)
  )
  importance_df <- importance_df[order(-importance_df$Importance), ]

  cat(" 特征重要性排序:\n")
  print(importance_df)

  # 可视化特征重要性
  ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance))
    geom_bar(stat = "identity", fill = "darkorange", alpha = 0.7) +
    labs(title = " 特征重要性 - 随机森林", x = " 特征", y = " 重要性") +
    coord_flip() +
    theme_minimal()
} else {
  cat(" 该学习器不支持特征重要性计算\n")
}
```

```
特征重要性排序:
       Feature  Importance
1  Petal.Width 0.296006354
2 Petal.Length 0.292785833
3 Sepal.Length 0.033216550
4  Sepal.Width 0.006679592
```



模型评估与混淆矩阵

```
# 模型评估与混淆矩阵

# 使用最佳模型进行预测
predictions <- learner_best$predict(task_preprocessed)

# 混淆矩阵
cat(" 混淆矩阵:\n")
```

混 淆 矩 阵 :

```
print(predictions$confusion)
```

```
            truth
response     setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         1
  virginica       0          2        49
```

14.5 Python–多分类

数据加载与探索

```
# 综合案例：鸢尾花分类 - Python 语言实现 (sklearn 框架)

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score, GridSea
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1
                             roc_auc_score, confusion_matrix, classification_rep
                             roc_curve, auc)
from sklearn.inspection import permutation_importance
import warnings
warnings.filterwarnings('ignore')

# 设置图形样式
plt.style.use('default')
sns.set_palette("husl")

# 加载鸢尾花数据集
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.Series(iris.target, name='species')
iris_data = pd.concat([X, y], axis=1)

# 映射数字标签到类别名称
target_names = iris.target_names
iris_data['species_name'] = iris_data['species'].map(lambda x: target_names[x])

print(" 数据集基本信息:")
```

数据集基本信息：

```
print(f" 样本数: {iris_data.shape[0]}")
```

样本数: 150

```
print(f" 变量数: {iris_data.shape[1]}")
```

变量数: 6

```
print("\n变量名称:")
```

变量名称:

```
print(iris_data.columns.tolist())
```

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)

```
# 类别分布
print("\n类别分布:")
```

类别分布:

```
class_dist = iris_data['species_name'].value_counts()
print(class_dist)
```

```
species_name
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

```
print("\n类别比例:")
```

类别比例:

```
print(class_dist / class_dist.sum())
```

```
species_name
setosa        0.333333
versicolor    0.333333
virginica     0.333333
Name: count, dtype: float64
```

```
print("\n数据概览:")
```

数据概览:

```
print(iris_data.describe())
```

```
       sepal length (cm)  sepal width (cm)  ...  petal width (cm)    species
count         150.000000        150.000000  ...        150.000000  150.000000
mean            5.843333          3.057333  ...          1.199333    1.000000
std             0.828066          0.435866  ...          0.762238    0.819232
min             4.300000          2.000000  ...          0.100000    0.000000
25%             5.100000          2.800000  ...          0.300000    0.000000
50%             5.800000          3.000000  ...          1.300000    1.000000
75%             6.400000          3.300000  ...          1.800000    2.000000
max             7.900000          4.400000  ...          2.500000    2.000000

[8 rows x 5 columns]
```

```
print("\n缺失值检查:")
```

缺失值检查:

```
print(iris_data.isnull().sum())
```

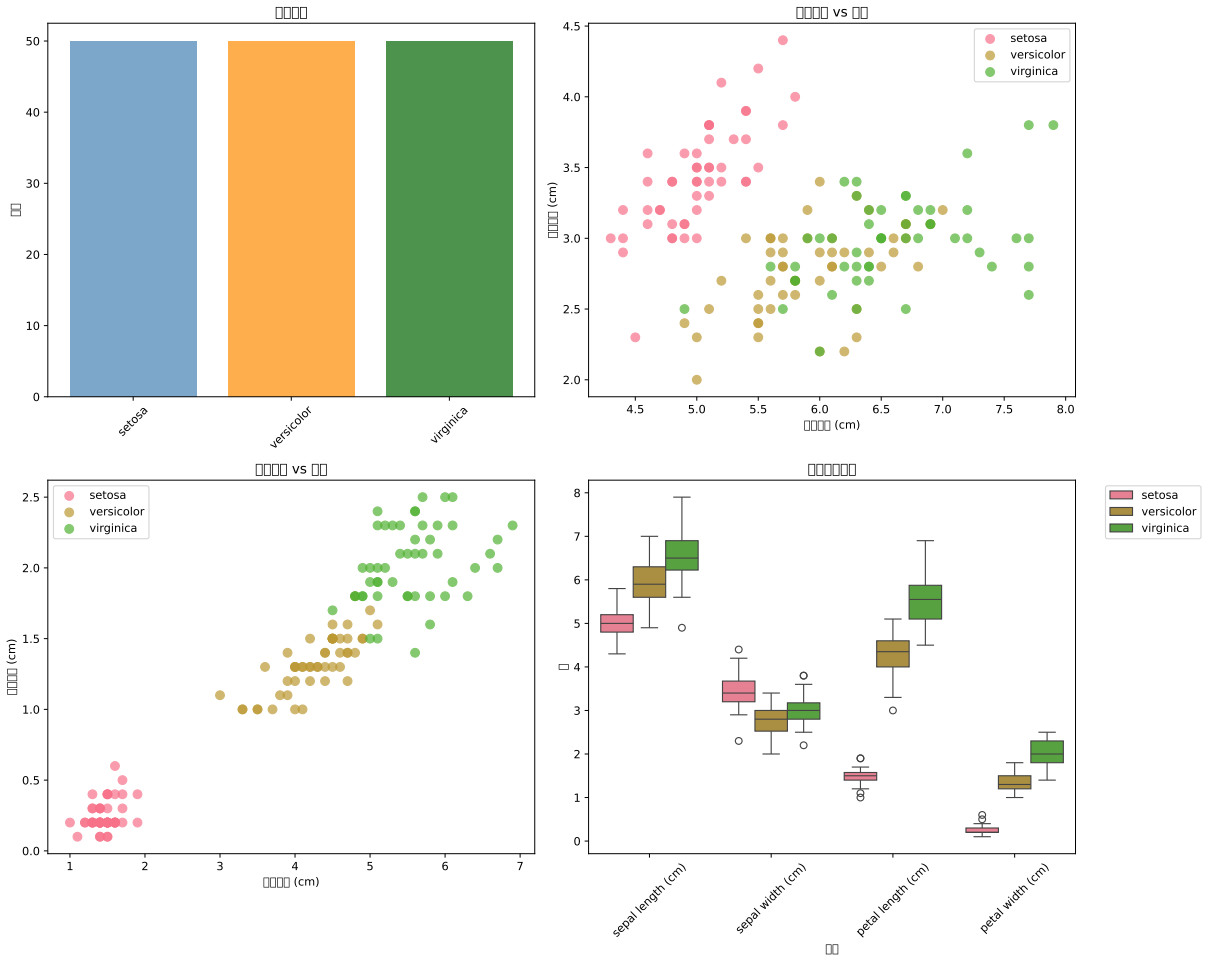
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
species              0
species_name         0
dtype: int64
```

数据可视化分析

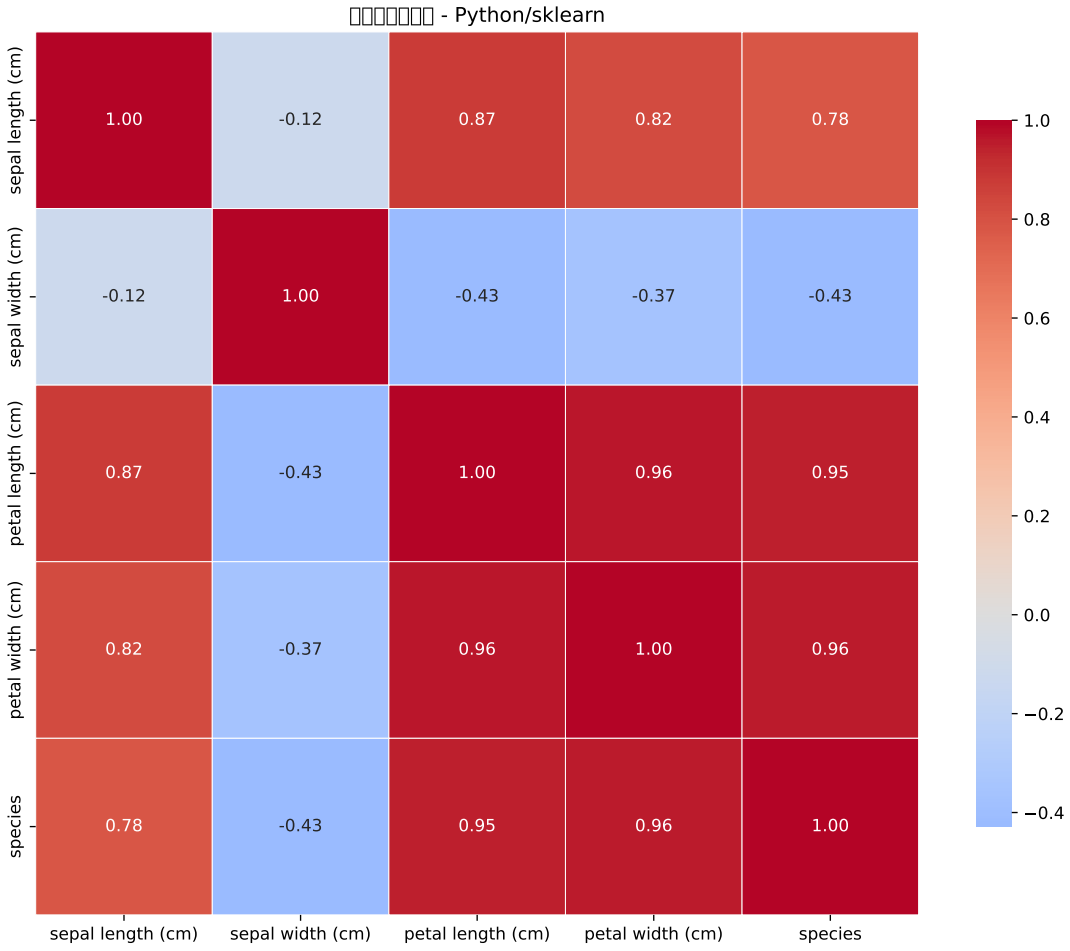
```
# 数据可视化分析 - Python

fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# 类别分布
class_counts = iris_data['species_name'].value_counts()
```

```
axes[0,0].bar(class_counts.index, class_counts.values, color=['steelblue', '
axes[0,0].set_title('类别分布')
axes[0,0].set_ylabel('数量')
axes[0,0].tick_params(axis='x', rotation=45)

# 花萼长度与宽度的关系
species_colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue
for species in iris_data['species_name'].unique():
    species_data = iris_data[iris_data['species_name'] == species]
    axes[0,1].scatter(species_data['sepal length (cm)'], species_data['sepal
                      label=species, alpha=0.7, s=60)
axes[0,1].set_title('花萼长度 vs 宽度')
axes[0,1].set_xlabel('花萼长度 (cm)')
axes[0,1].set_ylabel('花萼宽度 (cm)')
axes[0,1].legend()

# 花瓣长度与宽度的关系
for species in iris_data['species_name'].unique():
    species_data = iris_data[iris_data['species_name'] == species]
    axes[1,0].scatter(species_data['petal length (cm)'], species_data['petal
                      label=species, alpha=0.7, s=60)
axes[1,0].set_title('花瓣长度 vs 宽度')
axes[1,0].set_xlabel('花瓣长度 (cm)')
axes[1,0].set_ylabel('花瓣宽度 (cm)')
axes[1,0].legend()

# 特征分布箱线图
feature_data = iris_data.melt(id_vars=['species_name'],
                              value_vars=['sepal length (cm)', 'sepal width (
                                          'petal length (cm)', 'petal width (cm
sns.boxplot(data=feature_data, x='variable', y='value', hue='species_name',
axes[1,1].set_title('特征分布比较')
axes[1,1].set_xlabel('特征')
axes[1,1].set_ylabel('值')
axes[1,1].tick_params(axis='x', rotation=45)
axes[1,1].legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
```

```
plt.show()
```




```
# 相关性热图
plt.figure(figsize=(10, 8))
corr_matrix = iris_data[['sepal length (cm)', 'sepal width (cm)',
                         'petal length (cm)', 'petal width (cm)', 'species']].c
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0,
            square=True, linewidths=0.5, cbar_kws={"shrink": 0.8}, fmt='.2f')
plt.title('变量相关性热图 - Python/sklearn')
plt.tight_layout()
plt.show()
```



数据预处理

```
# 数据预处理 - Python

# 准备特征和目标变量
X = iris_data[['sepal length (cm)', 'sepal width (cm)',
               'petal length (cm)', 'petal width (cm)']]
y = iris_data['species']

# 数据划分（分层抽样保持类别比例）
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

print(f" 训练集大小: {X_train.shape[0]}")
```

```
训练集大小: 105
```

```
print(f" 测试集大小: {X_test.shape[0]}")
```

```
测试集大小: 45
```

```
print(f" 训练集类别分布:\n{y_train.value_counts().sort_index()}")
```

```
训练集类别分布:
species
0    35
1    35
2    35
Name: count, dtype: int64
```

```
print(f" 测试集类别分布:\n{y_test.value_counts().sort_index()}")
```

```
测试集类别分布:
species
0    15
1    15
2    15
Name: count, dtype: int64
```

```
# 数据标准化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print(" 数据预处理完成")
```

```
数据预处理完成
```

模型训练与比较

```
# 模型训练与比较 - sklearn

# 定义模型字典
models = {
    'Logistic Regression': LogisticRegression(random_state=42, max_iter=1000),
    'Random Forest': RandomForestClassifier(random_state=42),
    'K-Neighbors': KNeighborsClassifier(),
    'Gradient Boosting': GradientBoostingClassifier(random_state=42)
```

```
}

# 训练和评估模型
results = []

for name, model in models.items():
    # 训练模型
    model.fit(X_train_scaled, y_train)

    # 预测
    y_pred = model.predict(X_test_scaled)
    y_pred_proba = model.predict_proba(X_test_scaled)

    # 计算评估指标
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='weighted')
    recall = recall_score(y_test, y_pred, average='weighted')
    f1 = f1_score(y_test, y_pred, average='weighted')

    # 多类 AUC（One-vs-Rest）
    try:
        auc_score = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
    except:
        auc_score = np.nan

    # 交叉验证
    cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    cv_scores = cross_val_score(model, X_train_scaled, y_train, cv=cv, scori
    cv_accuracy_mean = cv_scores.mean()

    results.append({
        'Model': name,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1_Score': f1,
        'AUC': auc_score,
        'CV_Accuracy': cv_accuracy_mean
```

```
    })
```

```
LogisticRegression(max_iter=1000, random_state=42)
RandomForestClassifier(random_state=42)
KNeighborsClassifier()
GradientBoostingClassifier(random_state=42)
```

```
# 创建结果 DataFrame
results_python = pd.DataFrame(results)
print(" 模型性能比较 - Python/sklearn:")
```

```
模型性能比较 - Python/sklearn:
```

```
print(results_python.sort_values('Accuracy', ascending=False))
```

```
            Model  Accuracy  Precision  ...  F1_Score     AUC  CV_Accuracy
3   Gradient Boosting  0.933333   0.944444  ...  0.932660  0.998519    0.961905
0  Logistic Regression  0.911111   0.915535  ...  0.910714  0.995556    0.980952
2        K-Neighbors  0.911111   0.929825  ...  0.909502  0.988889    0.942857
1      Random Forest  0.888889   0.898148  ...  0.887767  0.992593    0.952381

[4 rows x 7 columns]
```

```
# 可视化模型比较
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# 准确率比较
models_sorted_acc = results_python.sort_values('Accuracy')
axes[0,0].barh(models_sorted_acc['Model'], models_sorted_acc['Accuracy'], colo
axes[0,0].set_xlabel('准确率')
axes[0,0].set_title('模型准确率比较 - Python/sklearn')
axes[0,0].set_xlim(0.8, 1.0)
```

```
(0.8, 1.0)
```

```
axes[0,0].grid(axis='x', alpha=0.3)

# F1 分数比较
models_sorted_f1 = results_python.sort_values('F1_Score')
axes[0,1].barh(models_sorted_f1['Model'], models_sorted_f1['F1_Score'], color=
axes[0,1].set_xlabel('F1 分数')
```

```
axes[0,1].set_title('模型 F1 分数比较 - Python/sklearn')
axes[0,1].set_xlim(0.8, 1.0)
```

```
(0.8, 1.0)
```

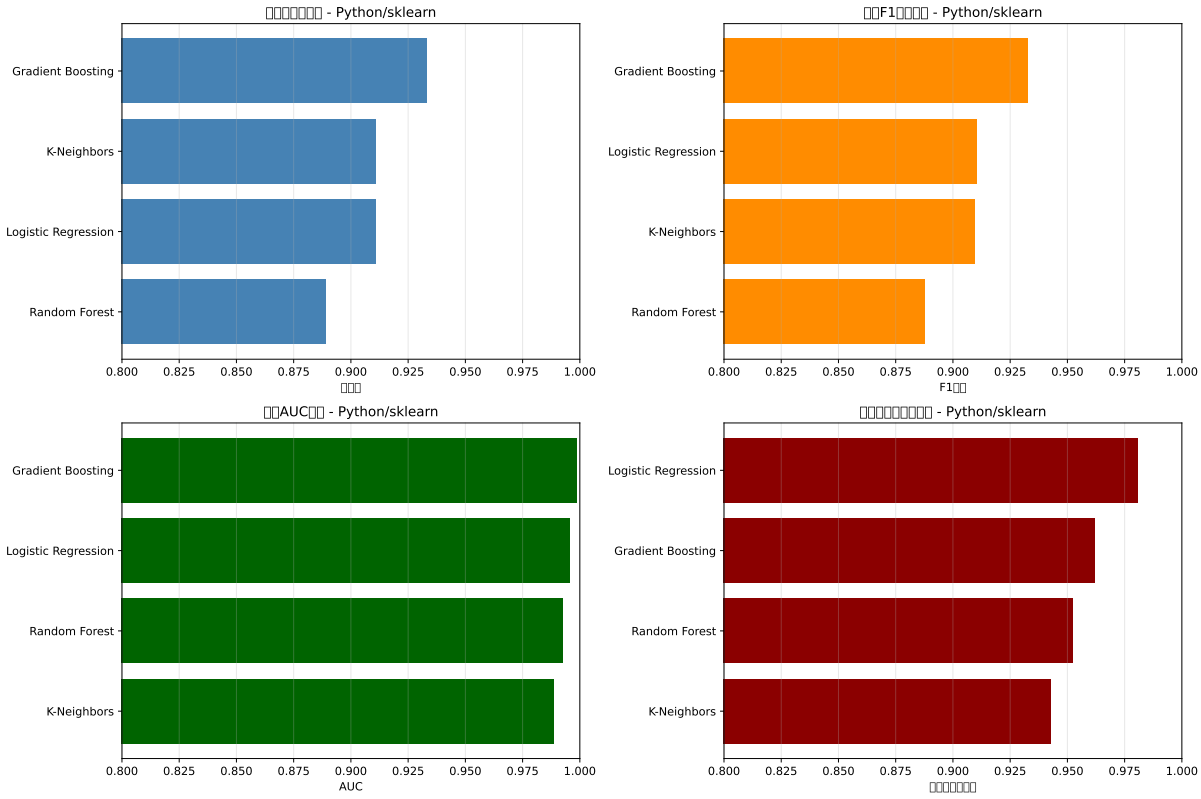
```
axes[0,1].grid(axis='x', alpha=0.3)

# AUC 比较
models_sorted_auc = results_python.sort_values('AUC')
axes[1,0].barh(models_sorted_auc['Model'], models_sorted_auc['AUC'], color='
axes[1,0].set_xlabel('AUC')
axes[1,0].set_title('模型 AUC 比较 - Python/sklearn')
axes[1,0].set_xlim(0.8, 1.0)
```

```
(0.8, 1.0)
```

```
axes[1,0].grid(axis='x', alpha=0.3)

# 交叉验证准确率比较
models_sorted_cv = results_python.sort_values('CV_Accuracy')
axes[1,1].barh(models_sorted_cv['Model'], models_sorted_cv['CV_Accuracy'], c
axes[1,1].set_xlabel('交叉验证准确率')
axes[1,1].set_title('交叉验证准确率比较 - Python/sklearn')
axes[1,1].set_xlim(0.8, 1.0)
```

```
(0.8, 1.0)
```

```
axes[1,1].grid(axis='x', alpha=0.3)

plt.tight_layout()
plt.show()
```



详细模型评估

```
# 详细模型评估 - 最佳模型

# 选择最佳模型（基于准确率）
best_model_name = results_python.loc[results_python['Accuracy'].idxmax(), 'Mod
best_model = models[best_model_name]

print(f" 最佳模型: {best_model_name}")
```

```
最佳模型: Gradient Boosting
```

```
# 重新训练最佳模型
best_model.fit(X_train_scaled, y_train)
```

```
GradientBoostingClassifier(random_state=42)
```

```
# 预测
y_pred_best = best_model.predict(X_test_scaled)
y_pred_proba_best = best_model.predict_proba(X_test_scaled)
```

```
# 混淆矩阵
cm = confusion_matrix(y_test, y_pred_best)
print("\n混淆矩阵:")
```

混 淆 矩 阵 :

```
print(cm)
```

```
[[15  0  0]
 [ 0 15  0]
 [ 0  3 12]]
```

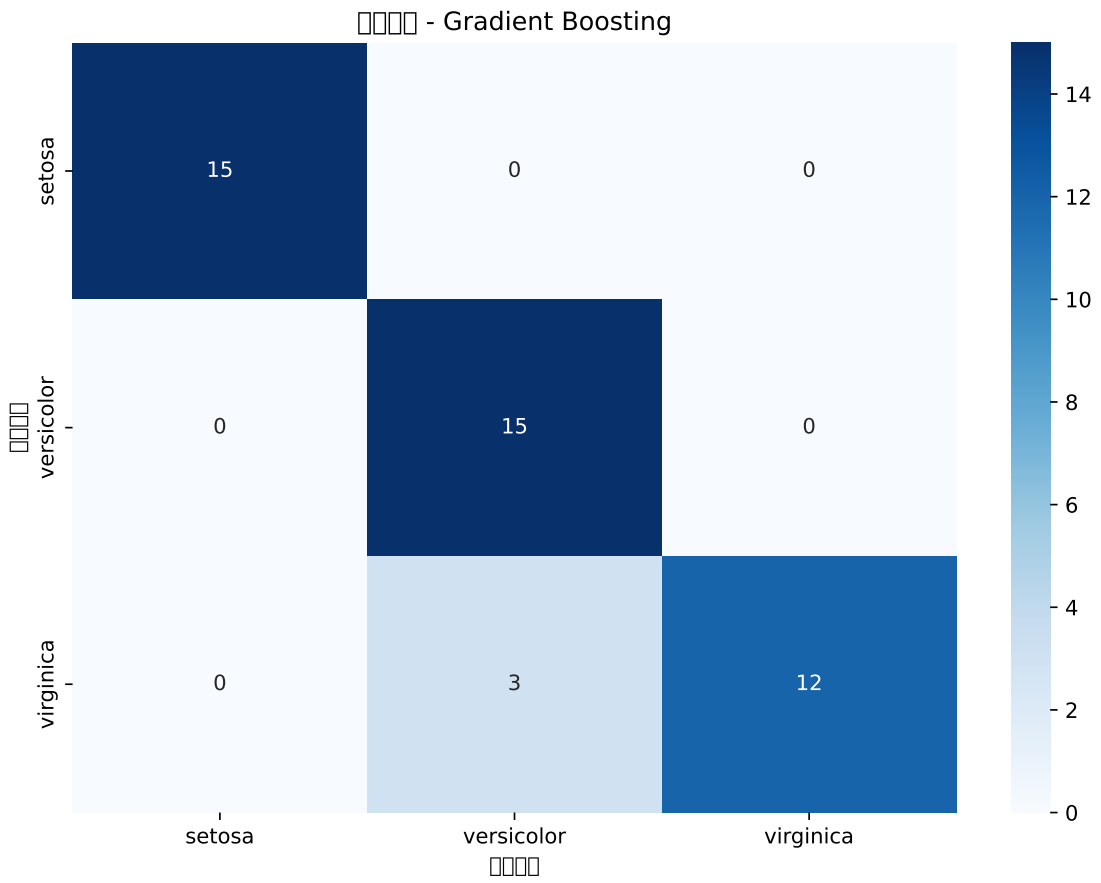
```
# 分类报告
print("\n分类报告:")
```

分 类 报 告 :

```
print(classification_report(y_test, y_pred_best, target_names=target_names))
```

```
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       0.83      1.00      0.91        15
   virginica       1.00      0.80      0.89        15

    accuracy                           0.93        45
   macro avg       0.94      0.93      0.93        45
weighted avg       0.94      0.93      0.93        45
```

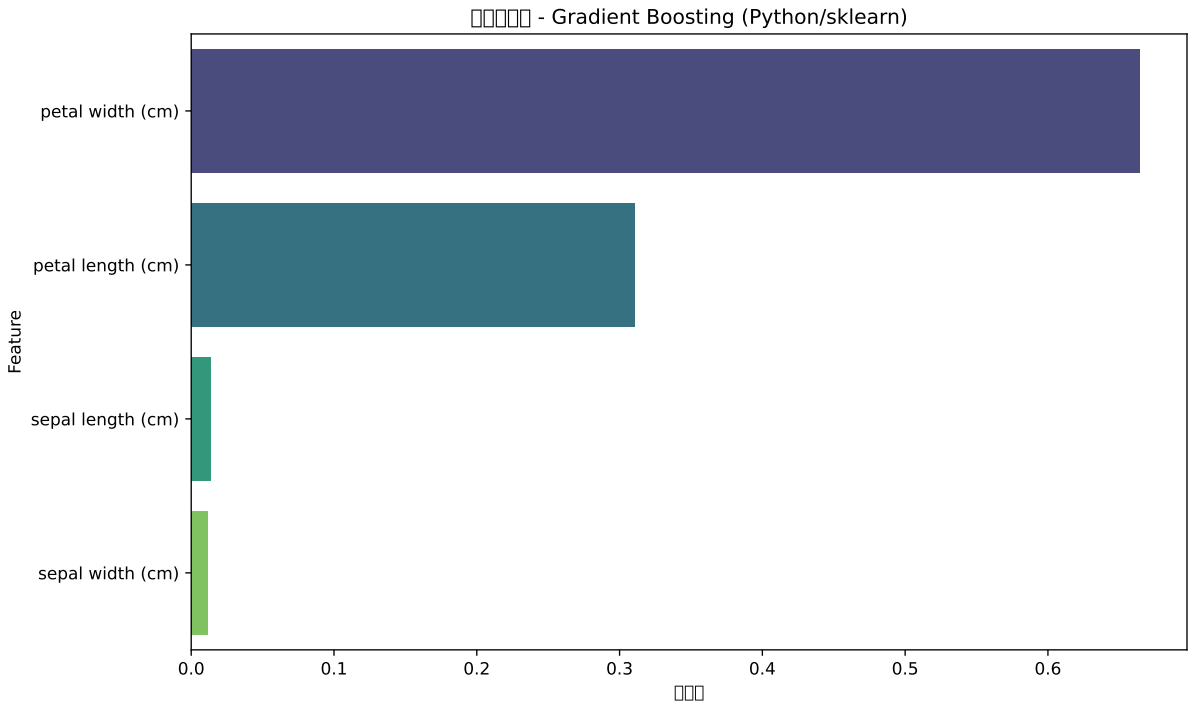
```
# 可视化混淆矩阵
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=target_names, yticklabels=target_names)
plt.title(f'混淆矩阵 - {best_model_name}')
plt.xlabel('预测标签')
plt.ylabel('真实标签')
plt.tight_layout()
plt.show()
```



特征重要性分析

```
# 特征重要性分析 - Python

if hasattr(best_model, 'feature_importances_'):
    feature_importance = best_model.feature_importances_
    feature_names = X.columns

    # 创建特征重要性 DataFrame
    importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': feature_importance
    }).sort_values('Importance', ascending=False)

    print(" 特征重要性排序 - Python/sklearn:")
    print(importance_df)

    # 可视化特征重要性
```

```
    plt.figure(figsize=(10, 6))
    sns.barplot(data=importance_df, x='Importance', y='Feature', palette='vi
    plt.title(f'特征重要性 - {best_model_name} (Python/sklearn)')
    plt.xlabel('重要性')
    plt.tight_layout()
    plt.show()

else:
    print(f"{best_model_name} 不支持特征重要性计算")

    # 使用排列重要性作为替代
    perm_importance = permutation_importance(
        best_model, X_test_scaled, y_test, n_repeats=10, random_state=42
    )

    perm_importance_df = pd.DataFrame({
        'Feature': feature_names,
        'Importance': perm_importance.importances_mean
    }).sort_values('Importance', ascending=False)

    print("\n排列特征重要性:")
    print(perm_importance_df)

    # 可视化排列重要性
    plt.figure(figsize=(10, 6))
    sns.barplot(data=perm_importance_df, x='Importance', y='Feature', palett
    plt.title(f'排列特征重要性 - {best_model_name} (Python/sklearn)')
    plt.xlabel('重要性')
    plt.tight_layout()
    plt.show()
```



14.6 多分类案例总结

关键发现

```
# 案例总结 - R 语言
cat("=== 鸢尾花分类案例总结 ===\n")
```

```
=== 鸢尾花分类案例总结 ===
```

```
cat("1. 数据特征:\n")
```

```
1. 数据特征:
```

```
cat("   - 数据集包含", nrow(iris_data), " 个样本, ", ncol(iris_data)-1, " 个特征\n"
```

```
   - 数据集包含 150 个样本， 4 个特征
```

```
cat("   - 目标变量: 鸢尾花种类 (3 个类别)\n")
```

```
   - 目标变量: 鸢尾花种类(3个类别)
```

```
cat("   - 类别分布均衡\n")
```

```
   - 类别分布均衡
```

```
cat("   - 最重要的特征: 花瓣长度和宽度\n\n")
```

```
   - 最重要的特征: 花瓣长度和宽度
```

```
cat("2. 模型性能比较:\n")
```

```
2. 模型性能比较:
```

```
print(results_r[order(-Accuracy)])
```

```
           Model  Accuracy   AUC Balanced_Accuracy    LogLoss
          <char>     <num> <num>             <num>      <num>
1:            svm 0.9600000   NaN         0.9636975 0.04000000
2: random_forest 0.9466667   NaN         0.9497292 0.05333333
3:       xgboost 0.9466667   NaN         0.9497292 0.05333333
4:           knn 0.9400000   NaN         0.9413959 0.06000000
```

```
cat("\n3. 分类问题最佳实践:\n")
```

```

3. 分类问题最佳实践:
```

```
cat("   - 树模型和 SVM 在分类问题上表现优异\n")
```

```
   - 树模型和SVM在分类问题上表现优异
```

```
cat("   - 特征标准化对距离-based 模型很重要\n")
```

```
   - 特征标准化对距离-based模型很重要
```

```
cat("   - 多分类问题需要考虑合适的评估指标\n")
```

```
   - 多分类问题需要考虑合适的评估指标
```

```
cat("   - 特征重要性分析有助于理解模型决策\n")
```

```
   - 特征重要性分析有助于理解模型决策
```

```
# 案例总结 - Python
print("=== 鸢尾花分类案例总结 ===")
```

```
=== 鸢尾花分类案例总结 ===
```

```
print("1. 数据特征:")
```

```
1. 数据特征:
```

```
print(f"   - 数据集包含 {iris_data.shape[0]} 个样本, {iris_data.shape[1]-1} 个特征"
```

- 数据集包含 150 个样本, 5 个特征

```
print("   - 目标变量: 鸢尾花种类 (3 个类别)")
```

- 目标变量: 鸢尾花种类(3个类别)

```
print("   - 类别分布均衡")
```

- 类别分布均衡

```
print("   - 最重要的特征: 花瓣长度和宽度\n")
```

- 最重要的特征: 花瓣长度和宽度

```
print("2. 模型性能比较:")
```

2. 模型性能比较:

```
print(results_python[['Model', 'Accuracy', 'F1_Score', 'AUC']].sort_values('Ac
```

```
              Model  Accuracy  F1_Score       AUC
  Gradient Boosting  0.933333  0.932660 0.998519
Logistic Regression  0.911111  0.910714 0.995556
        K-Neighbors  0.911111  0.909502 0.988889
      Random Forest  0.888889  0.887767 0.992593
```

```
print("\n3. 分类问题最佳实践:")
```

3. 分类问题最佳实践:

```
print("   - 集成学习方法在分类问题上表现稳定")
```

- 集成学习方法在分类问题上表现稳定

```
print("   - 数据预处理对模型性能影响显著")
```

- 数据预处理对模型性能影响显著

```
print("   - 交叉验证提供更可靠的性能估计")
```

- 交叉验证提供更可靠的性能估计

```
print("    - 混淆矩阵和分类报告提供详细性能分析")
```

```
    - 混淆矩阵和分类报告提供详细性能分析
```

技术要点回顾

数据预处理：- 特征标准化：确保不同尺度的特征具有可比性 - 类别编码：将文本标签转换为数值 - 数据划分：使用分层抽样保持类别比例

模型选择：- 逻辑回归：可解释性强，适合线性可分数据 - 树模型：自动处理非线性关系，提供特征重要性 - SVM：在高维空间中表现优异 - KNN：基于距离的简单有效方法

模型评估：- 准确率：整体分类正确率 - 精确率、召回率、F1 分数：更细致的性能评估 - AUC：模型区分能力的综合指标 - 混淆矩阵：详细分析各类别的分类情况

模型优化：- 超参数调优：提升模型性能 - 特征选择：基于重要性筛选特征 - 交叉验证：稳健的性能估计

业务应用建议

- 物种识别：扩展到其他植物或动物的分类识别
- 质量检测：应用于工业产品质量分类
- 医疗诊断：用于疾病分类和诊断辅助
- 客户细分：用于市场营销中的客户分类

15 案例综合分析

案例导读

本项目基于模拟的电商用户数据，进行全面的数据分析，包括：- 回归分析：预测用户消费金额 - 分类分析：预测用户购买意向 - 聚类分析：用户分群 - 关联分析：商品购买关联规则

数据说明

```
# 生成模拟数据
set.seed(123)
n <- 1000

# 用户基本信息
user_data <- data.frame(
  user_id = 1:n,
  age = sample(18:65, n, replace = TRUE),
  gender = sample(c("Male", "Female"), n, replace = TRUE, prob = c(0.48, 0.52)
  income = round(rnorm(n, 50000, 15000)),
  region = sample(c("North", "South", "East", "West"), n, replace = TRUE),
  days_since_signup = sample(1:365, n, replace = TRUE),
  page_views = rpois(n, 25),
  time_on_site = round(rnorm(n, 300, 120)),
  cart_additions = rpois(n, 5)
)

# 生成购买行为数据
user_data$purchase_amount <- with(user_data,
  50 + 0.3 * income/1000 + 0.5 * page_views + 0.8 * time_on_site/60 +
  2 * cart_additions + rnorm(n, 0, 20)
)
```

```
user_data$made_purchase <- ifelse(user_data$purchase_amount > 120, 1, 0)

# 添加产品类别购买信息 - 增加相关性以产生关联规则
products <- c("Electronics", "Clothing", "Books", "Home", "Sports")
set.seed(123)
for(i in 1:length(products)) {
  product <- products[i]
  base_prob <- 0.4
  # 创建产品间的相关性
  if(i > 1) {
    # 让某些产品更可能一起购买
    correlated_prob <- user_data[[paste0("bought_", tolower(products[i-1]))]]
    user_data[[paste0("bought_", tolower(product))]] <- rbinom(n, 1, pmin(co
  } else {
    user_data[[paste0("bought_", tolower(product))]] <- rbinom(n, 1, base_pr
  }
}

head(user_data)
```

```
  user_id age gender income region days_since_signup page_views time_on_site
1       1  48 Female  51762   West                5        29          173
2       2  32 Female  45027   West               22        28          450
3       3  31   Male  54174   West                4        38          147
4       4  20 Female  32216  South              352        23          243
5       5  59   Male  37462  North              131        23          340
6       6  60 Female  57654  North              231        15          334
  cart_additions purchase_amount made_purchase bought_electronics
1              3        57.57759             0                  0
2              3        59.83742             0                  1
3              6        64.31744             0                  0
4             10        94.88951             0                  1
5              1        24.43966             0                  1
6              5       114.33971             0                  0
  bought_clothing bought_books bought_home bought_sports
1               0            0           0             0
2               1            1           0             1
3               0            0           0             0
```

```
4                0            0            1            0
5                0            0            0            0
6                0            1            1            1
```

15.1 R 语言实现

15.1.1. 数据探索与预处理

数据概览

```
library(dplyr)
library(ggplot2)
library(corrplot)

# 基本统计信息
summary(user_data[, c("age", "income", "page_views", "time_on_site", "purchase
```

```
      age             income         page_views     time_on_site
 Min.   :18.00   Min.   :  4282   Min.   :11.00   Min.   :-83.0
 1st Qu.:29.00   1st Qu.: 40201   1st Qu.:21.00   1st Qu.:219.8
 Median :42.00   Median : 49943   Median :25.00   Median :299.0
 Mean   :41.39   Mean   : 50209   Mean   :24.92   Mean   :299.7
 3rd Qu.:53.00   3rd Qu.: 60887   3rd Qu.:28.00   3rd Qu.:376.2
 Max.   :65.00   Max.   :100856   Max.   :42.00   Max.   :688.0
 purchase_amount
 Min.   :  9.555
 1st Qu.: 76.051
 Median : 91.131
 Mean   : 90.966
 3rd Qu.:105.509
 Max.   :153.144
```

```
# 缺失值检查
sapply(user_data, function(x) sum(is.na(x)))
```

```
        user_id             age          gender          income
              0               0               0               0
         region days_since_signup      page_views     time_on_site
              0               0               0               0
```

```
   cart_additions    purchase_amount      made_purchase bought_electronics
                0                  0                  0                  0
  bought_clothing      bought_books      bought_home     bought_sports
                0                  0                  0                  0
```

数据可视化

```
# 数值变量分布
p1 <- ggplot(user_data, aes(x = purchase_amount)) +
  geom_histogram(bins = 30, fill = "steelblue", alpha = 0.7) +
  labs(title = " 购买金额分布")

p2 <- ggplot(user_data, aes(x = factor(made_purchase), fill = factor(made_pu
  geom_bar() +
  labs(title = " 购买行为分布", x = " 是否购买")

p3 <- ggplot(user_data, aes(x = age, y = purchase_amount, color = gender)) +
  geom_point(alpha = 0.6) +
  labs(title = " 年龄与购买金额关系")

library(patchwork)
p1 / p2 / p3
```

15.1.2. 回归分析：预测购买金额

线性回归模型

```
library(caret)

# 准备数据
reg_data <- user_data %>%
  select(age, income, page_views, time_on_site, cart_additions, purchase_amoun
  na.omit()

# 数据分割
set.seed(123)
train_index <- createDataPartition(reg_data$purchase_amount, p = 0.7, list = F
train_reg <- reg_data[train_index, ]
test_reg <- reg_data[-train_index, ]

# 训练线性回归模型
lm_model <- lm(purchase_amount ~ ., data = train_reg)
summary(lm_model)
```

```
Call:
lm(formula = purchase_amount ~ ., data = train_reg)

Residuals:
    Min      1Q  Median      3Q     Max
-66.576 -12.853  -0.106  13.819  67.749

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.111e+01  5.704e+00   8.961  < 2e-16 ***
age            8.216e-04  5.349e-02   0.015  0.98775
income         2.969e-04  5.022e-05   5.913 5.28e-09 ***
page_views     2.720e-01  1.496e-01   1.818  0.06954 .
time_on_site   1.635e-02  6.067e-03   2.695  0.00721 **
cart_additions 2.638e+00  3.326e-01   7.933 8.57e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

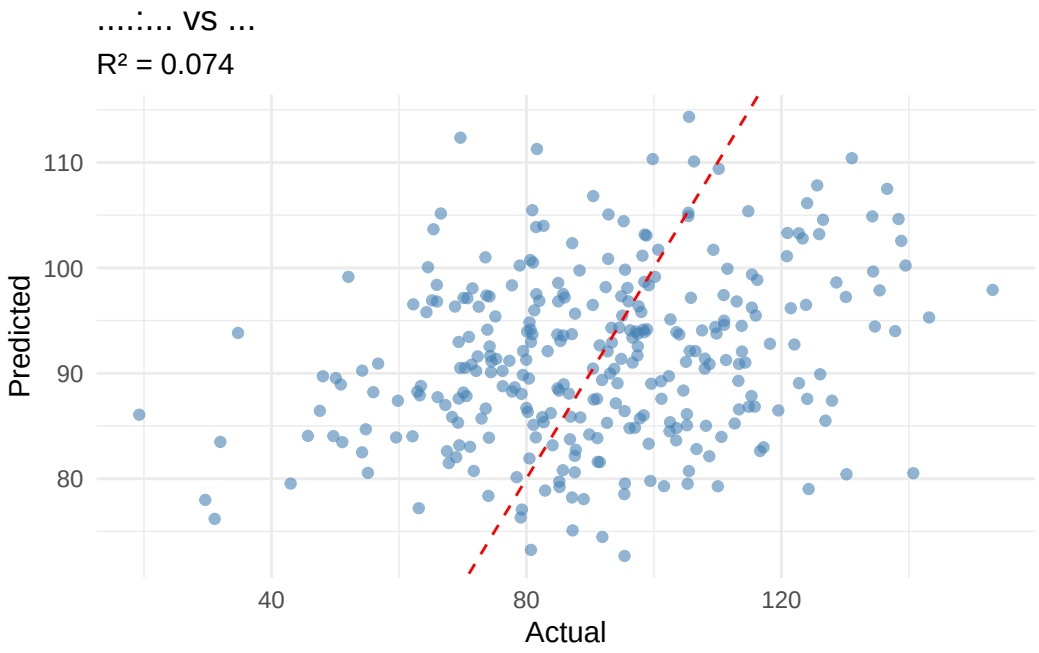
```
Residual standard error: 19.32 on 694 degrees of freedom
Multiple R-squared:  0.1368,    Adjusted R-squared:  0.1305
F-statistic: 21.99 on 5 and 694 DF,  p-value: < 2.2e-16
```

```
# 预测
predictions_lm <- predict(lm_model, newdata = test_reg)

# 模型评估
reg_metrics <- data.frame(
  RMSE = RMSE(predictions_lm, test_reg$purchase_amount),
  MAE = MAE(predictions_lm, test_reg$purchase_amount),
  R2 = R2(predictions_lm, test_reg$purchase_amount)
)
print(reg_metrics)
```

```
      RMSE      MAE         R2
1 21.72114 17.54768 0.07424787
```

回归结果可视化

```
# 预测 vs 实际值
results_df <- data.frame(
  Actual = test_reg$purchase_amount,
  Predicted = predictions_lm
)

ggplot(results_df, aes(x = Actual, y = Predicted)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_abline(slope = 1, intercept = 0, color = "red", linetype = "dashed") +
  labs(title = " 线性回归: 预测值 vs 实际值",
       subtitle = paste("R² =", round(reg_metrics$R2, 3))) +
  theme_minimal()
```



###15.1.3. 分类分析：预测购买意向

逻辑回归分类

```
# 准备分类数据 - 确保因子水平一致
class_data <- user_data %>%
  select(age, income, page_views, time_on_site, cart_additions, made_purchase)
  mutate(made_purchase = factor(made_purchase, levels = c(0, 1), labels = c("N
  na.omit()

# 数据分割
set.seed(123)
train_index_class <- createDataPartition(class_data$made_purchase, p = 0.7, li
train_class <- class_data[train_index_class, ]
test_class <- class_data[-train_index_class, ]

# 训练逻辑回归模型
logit_model <- glm(made_purchase ~ ., data = train_class, family = binomial)
summary(logit_model)
```

```
Call:
glm(formula = made_purchase ~ ., family = binomial, data = train_class)
```

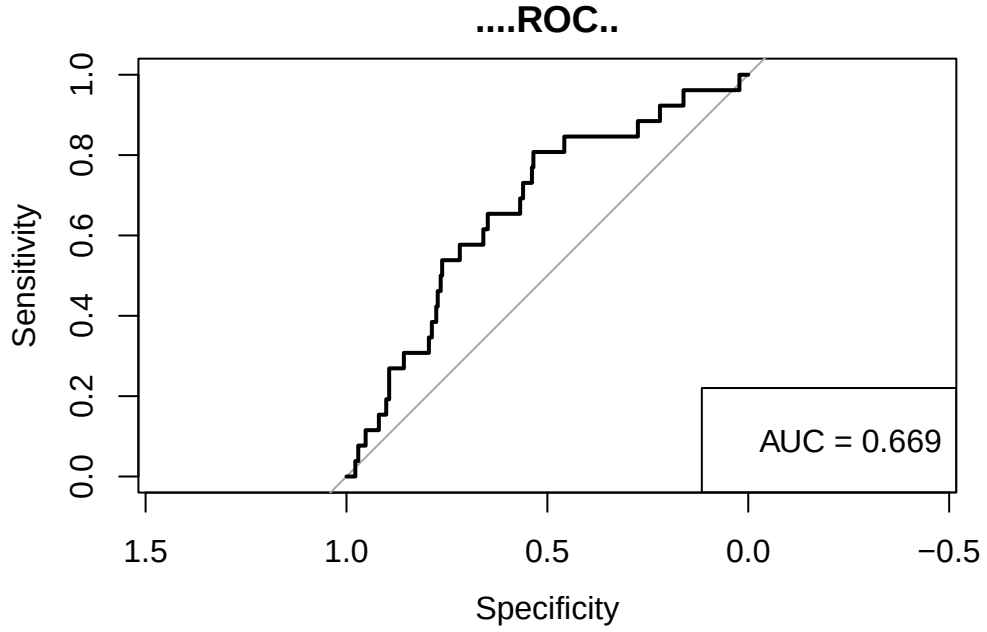
```
Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)    -7.571e+00  1.118e+00  -6.771 1.28e-11 ***
age             1.361e-03  1.002e-02   0.136   0.8920
income          2.353e-05  9.733e-06   2.418   0.0156 *
page_views      5.748e-02  2.661e-02   2.160   0.0308 *
time_on_site    2.964e-03  1.157e-03   2.562   0.0104 *
cart_additions  2.789e-01  6.152e-02   4.533 5.82e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 419.09  on 700  degrees of freedom
Residual deviance: 381.44  on 695  degrees of freedom
AIC: 393.44

Number of Fisher Scoring iterations: 5
```

```r
# 预测概率
prob_predictions <- predict(logit_model, newdata = test_class, type = "respo
class_predictions <- ifelse(prob_predictions > 0.5, "Yes", "No")
class_predictions <- factor(class_predictions, levels = c("No", "Yes"))

# 模型评估 - 修复因子水平问题
conf_matrix <- confusionMatrix(class_predictions, test_class$made_purchase,
print(conf_matrix)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  273  26
       Yes   0   0

               Accuracy : 0.913
                 95% CI : (0.8752, 0.9424)
    No Information Rate : 0.913
    P-Value [Acc > NIR] : 0.552
```

```
                 Kappa : 0

 Mcnemar's Test P-Value : 9.443e-07

            Sensitivity : 0.00000
            Specificity : 1.00000
         Pos Pred Value :     NaN
         Neg Pred Value : 0.91304
             Prevalence : 0.08696
         Detection Rate : 0.00000
   Detection Prevalence : 0.00000
      Balanced Accuracy : 0.50000

       'Positive' Class : Yes
```

```r
# 绘制 ROC 曲线
library(pROC)
roc_curve <- roc(as.numeric(test_class$made_purchase) - 1, prob_predictions)
plot(roc_curve, main = " 逻辑回归 ROC 曲线")
auc_value <- auc(roc_curve)
legend("bottomright", legend = paste("AUC =", round(auc_value, 3)))
```



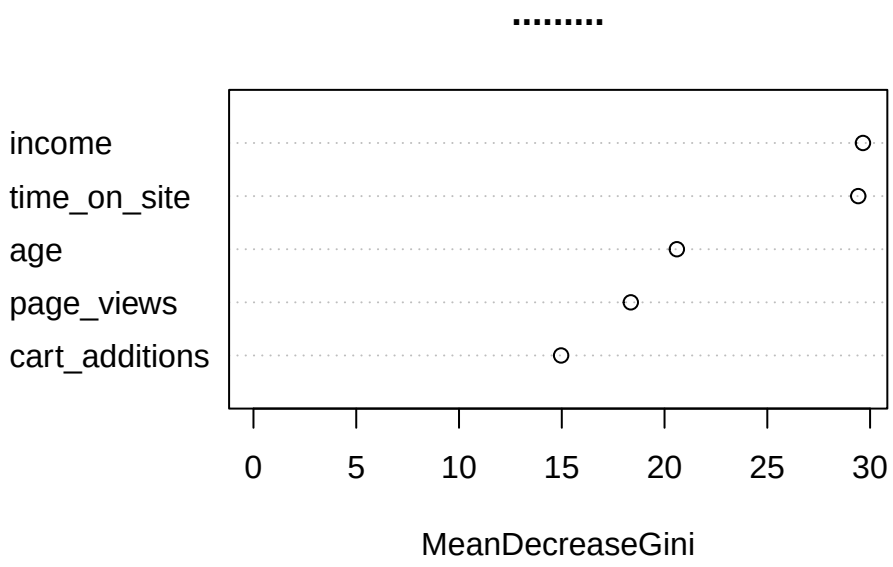
随机森林分类

```
library(randomForest)

# 确保训练数据和测试数据的因子水平一致
train_class$made_purchase <- factor(train_class$made_purchase, levels = c("N
test_class$made_purchase <- factor(test_class$made_purchase, levels = c("No"

# 训练随机森林
rf_model <- randomForest(made_purchase ~ ., data = train_class, ntree = 100)

# 预测
rf_predictions <- predict(rf_model, newdata = test_class)

# 评估 - 确保因子水平一致
rf_conf_matrix <- confusionMatrix(rf_predictions, test_class$made_purchase,
print(rf_conf_matrix)
```

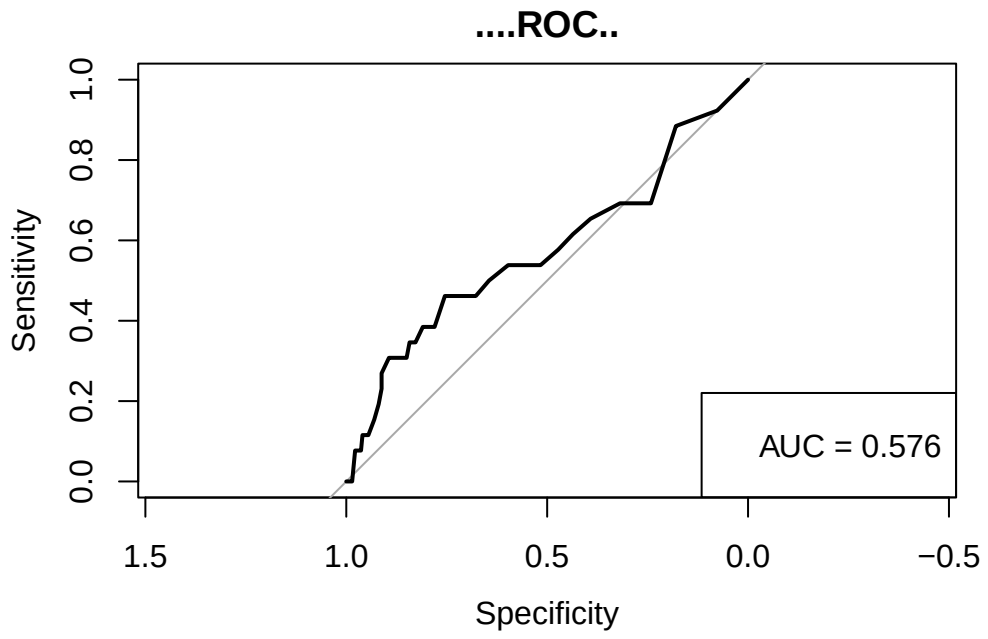
```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  271  26
       Yes   2   0

               Accuracy : 0.9064
                 95% CI : (0.8675, 0.9369)
    No Information Rate : 0.913
    P-Value [Acc > NIR] : 0.7033

                  Kappa : -0.0126

 Mcnemar's Test P-Value : 1.383e-05

            Sensitivity : 0.000000
            Specificity : 0.992674
         Pos Pred Value : 0.000000
         Neg Pred Value : 0.912458
             Prevalence : 0.086957
```

```
          Detection Rate : 0.000000
    Detection Prevalence : 0.006689
       Balanced Accuracy : 0.496337

        'Positive' Class : Yes
```

```r
# 变量重要性
varImpPlot(rf_model, main = " 随机森林变量重要性")
```




```r
# 随机森林 ROC 曲线
rf_prob <- predict(rf_model, newdata = test_class, type = "prob")[, "Yes"]
rf_roc <- roc(as.numeric(test_class$made_purchase) - 1, rf_prob)
plot(rf_roc, main = " 随机森林 ROC 曲线")
rf_auc <- auc(rf_roc)
legend("bottomright", legend = paste("AUC =", round(rf_auc, 3)))
```

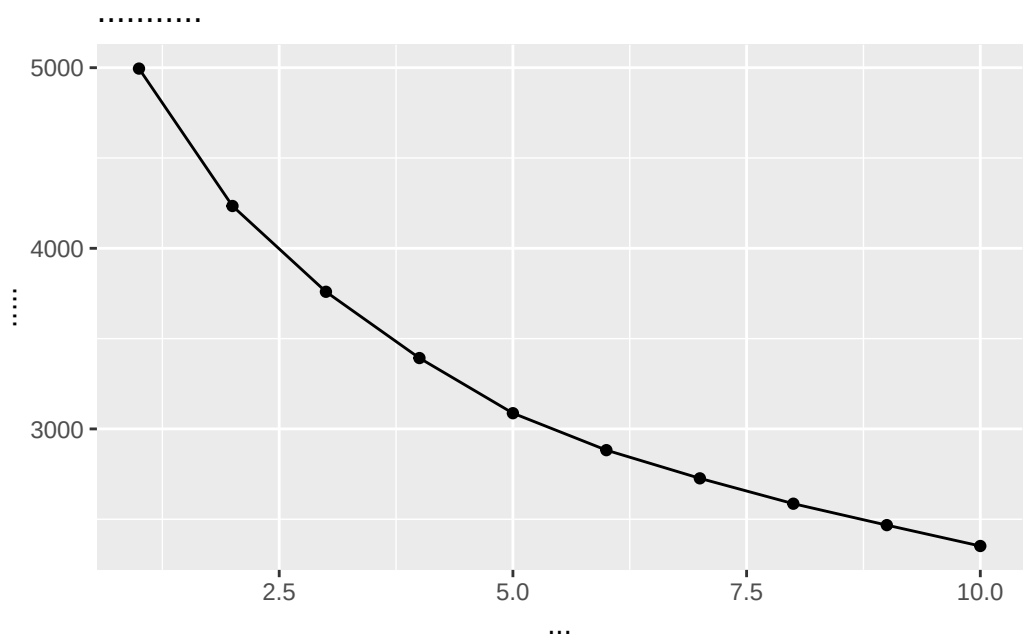


分类模型比较

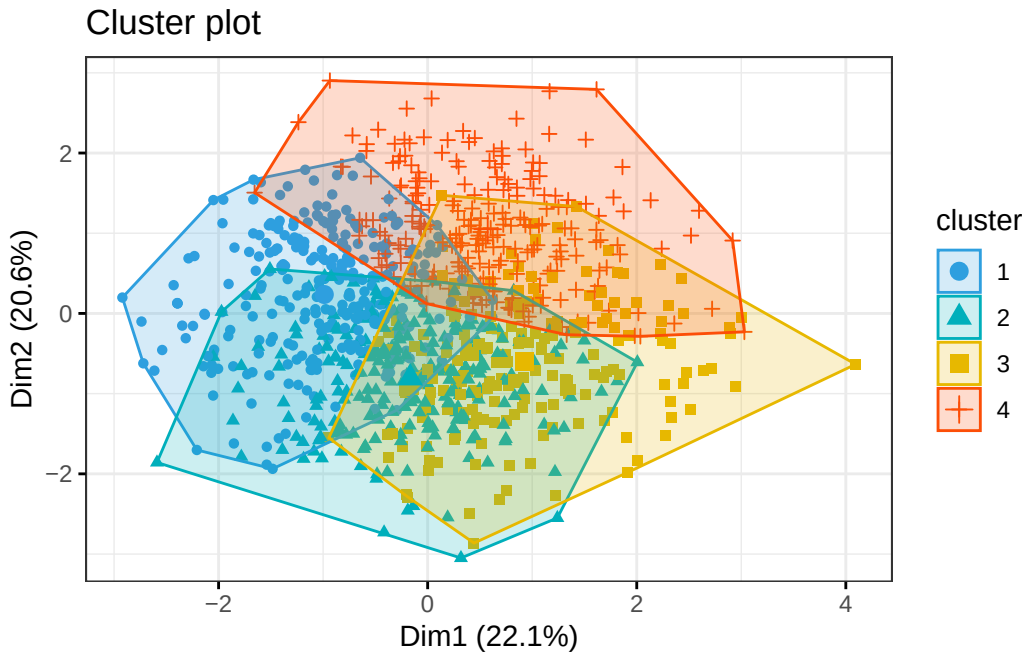
15.1.4. 聚类分析：用户分群

K-means 聚类

```
# 准备聚类数据
cluster_data <- user_data %>%
  select(age, income, page_views, time_on_site, cart_additions) %>%
  scale()  # 标准化

# 确定最佳聚类数
wss <- sapply(1:10, function(k){kmeans(cluster_data, k, nstart = 25)$tot.with

ggplot(data.frame(k = 1:10, wss = wss), aes(x = k, y = wss)) +
  geom_line() + geom_point() +
  labs(title = " 肘部法则确定最佳聚类数", x = " 聚类数", y = " 组内平方和")
```




```r
# 执行 K-means 聚类
set.seed(123)
kmeans_result <- kmeans(cluster_data, centers = 4, nstart = 25)

# 添加聚类标签
user_data$cluster <- as.factor(kmeans_result$cluster)

# 聚类结果可视化
library(factoextra)
fviz_cluster(kmeans_result, data = cluster_data,
             palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
             geom = "point",
             ellipse.type = "convex",
             ggtheme = theme_bw())
```



聚类分析

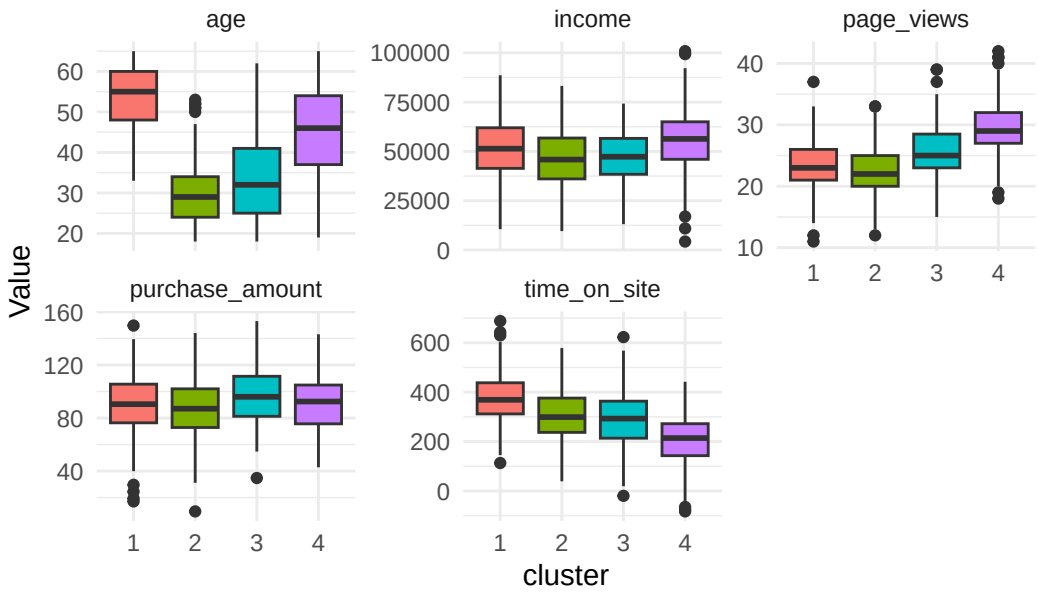
```
# 聚类特征分析
cluster_summary <- user_data %>%
  group_by(cluster) %>%
  summarise(
    n = n(),
    avg_age = mean(age),
    avg_income = mean(income),
    avg_page_views = mean(page_views),
    avg_time = mean(time_on_site),
    purchase_rate = mean(made_purchase),
    avg_purchase = mean(purchase_amount)
  )

print(cluster_summary)
```

```
# A tibble: 4 x 8
  cluster     n avg_age avg_income avg_page_views avg_time purchase_rate
  <fct>   <int>   <dbl>      <dbl>          <dbl>    <dbl>         <dbl>
1 1         292    53.9     50802.           23.1     376.        0.0822
2 2         257    30.1     46562.           22.3     304.        0.0506
3 3         203    33.8     47358.           25.6     293.        0.133
```

```
4 4         248    44.6     55625.          29.2     210.        0.0968
# i 1 more variable: avg_purchase <dbl>
```

```r
# 聚类特征可视化 - 修复版本
cluster_features <- user_data %>%
  select(cluster, age, income, page_views, time_on_site, purchase_amount)

# 使用 tidyr::pivot_longer
cluster_features_long <- tidyr::pivot_longer(
  cluster_features,
  cols = -cluster,
  names_to = "Feature",
  values_to = "Value"
)

ggplot(cluster_features_long, aes(x = cluster, y = Value, fill = cluster)) +
  geom_boxplot() +
  facet_wrap(~ Feature, scales = "free_y") +
  labs(title = " 各聚类群体特征分布") +
  theme_minimal() +
  theme(legend.position = "none")
```

.....



15.1.5. 关联分析：商品购买关联规则

Apriori 算法

```
library(arules)

# 准备关联分析数据 - 只包含产品购买列
transaction_data <- user_data %>%
  select(starts_with("bought_"))

# 转换为事务数据
transactions <- as(transaction_data, "transactions")

# 查看事务数据概览
summary(transactions)
```

```
transactions as itemMatrix in sparse format with
 1000 rows (elements/itemsets/transactions) and
 5 columns (items) and a density of 1

most frequent items:
bought_electronics=[0,1]   bought_clothing=[0,1]     bought_books=[0,1]
                    1000                     1000                    1000
     bought_home=[0,1]     bought_sports=[0,1]                (Other)
                    1000                     1000                      0

element (itemset/transaction) length distribution:
sizes
   5
1000

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      5       5       5       5       5       5

includes extended item information - examples:
                     labels          variables levels
1 bought_electronics=[0,1] bought_electronics  [0,1]
2    bought_clothing=[0,1]    bought_clothing  [0,1]
3       bought_books=[0,1]       bought_books  [0,1]
```

```
includes extended transaction information - examples:
  transactionID
1             1
2             2
3             3
```

```
# 查看产品频率
itemFrequency <- itemFrequency(transactions)
print(" 产品购买频率:")
```

```
[1] "产品购买频率:"
```

```
print(sort(itemFrequency, decreasing = TRUE))
```

```
bought_electronics=[0,1]   bought_clothing=[0,1]     bought_books=[0,1]
                       1                         1                         1
       bought_home=[0,1]      bought_sports=[0,1]
                       1                         1
```

```
# 使用更宽松的参数挖掘关联规则
rules <- apriori(transactions,
                 parameter = list(support = 0.05,   # 降低支持度阈值
                                  confidence = 0.3,   # 降低置信度阈值
                                  minlen = 2,         # 最小规则长度
                                  maxlen = 4))        # 最大规则长度
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
       0.3    0.1    1 none FALSE            TRUE       5    0.05      2
 maxlen target  ext
      4  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 50
```

```
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[5 item(s), 1000 transaction(s)] done [0.00s].
sorting and recoding items ... [5 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4

 done [0.00s].
writing ... [70 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
# 规则摘要
cat(" 找到的规则数量:", length(rules), "\n")
```

```
找到的规则数量: 70
```

```
summary(rules)
```

```
set of 70 rules

rule length distribution (lhs + rhs):sizes
 2  3  4
20 30 20

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       3       3       4       4

summary of quality measures:
    support    confidence    coverage      lift       count
 Min.   :1   Min.   :1    Min.   :1   Min.   :1   Min.   :1000
 1st Qu.:1   1st Qu.:1    1st Qu.:1   1st Qu.:1   1st Qu.:1000
 Median :1   Median :1    Median :1   Median :1   Median :1000
 Mean   :1   Mean   :1    Mean   :1   Mean   :1   Mean   :1000
 3rd Qu.:1   3rd Qu.:1    3rd Qu.:1   3rd Qu.:1   3rd Qu.:1000
 Max.   :1   Max.   :1    Max.   :1   Max.   :1   Max.   :1000

mining info:
         data ntransactions support confidence
 transactions          1000    0.05        0.3
                                                                call
 apriori(data = transactions, parameter = list(support = 0.05, confidence = 0.3,
```

```
# 查看规则
if(length(rules) > 0) {
  # 按提升度排序
  rules_sorted <- sort(rules, by = "lift", decreasing = TRUE)
  cat("\n前 10 条关联规则 (按提升度排序):\n")
  inspect(head(rules_sorted, 10))
} else {
  cat(" 没有找到关联规则，请进一步降低阈值参数\n")
}
```

```
前10条关联规则 (按提升度排序):
     lhs                              rhs                           support
[1]  {bought_electronics=[0,1]} => {bought_clothing=[0,1]}     1
[2]  {bought_clothing=[0,1]}     => {bought_electronics=[0,1]} 1
[3]  {bought_electronics=[0,1]} => {bought_books=[0,1]}        1
[4]  {bought_books=[0,1]}        => {bought_electronics=[0,1]} 1
[5]  {bought_electronics=[0,1]} => {bought_home=[0,1]}         1
[6]  {bought_home=[0,1]}         => {bought_electronics=[0,1]} 1
[7]  {bought_electronics=[0,1]} => {bought_sports=[0,1]}       1
[8]  {bought_sports=[0,1]}       => {bought_electronics=[0,1]} 1
[9]  {bought_clothing=[0,1]}     => {bought_books=[0,1]}        1
[10] {bought_books=[0,1]}        => {bought_clothing=[0,1]}     1
     confidence coverage lift count
[1]  1          1        1    1000
[2]  1          1        1    1000
[3]  1          1        1    1000
[4]  1          1        1    1000
[5]  1          1        1    1000
[6]  1          1        1    1000
[7]  1          1        1    1000
[8]  1          1        1    1000
[9]  1          1        1    1000
[10] 1          1        1    1000
```

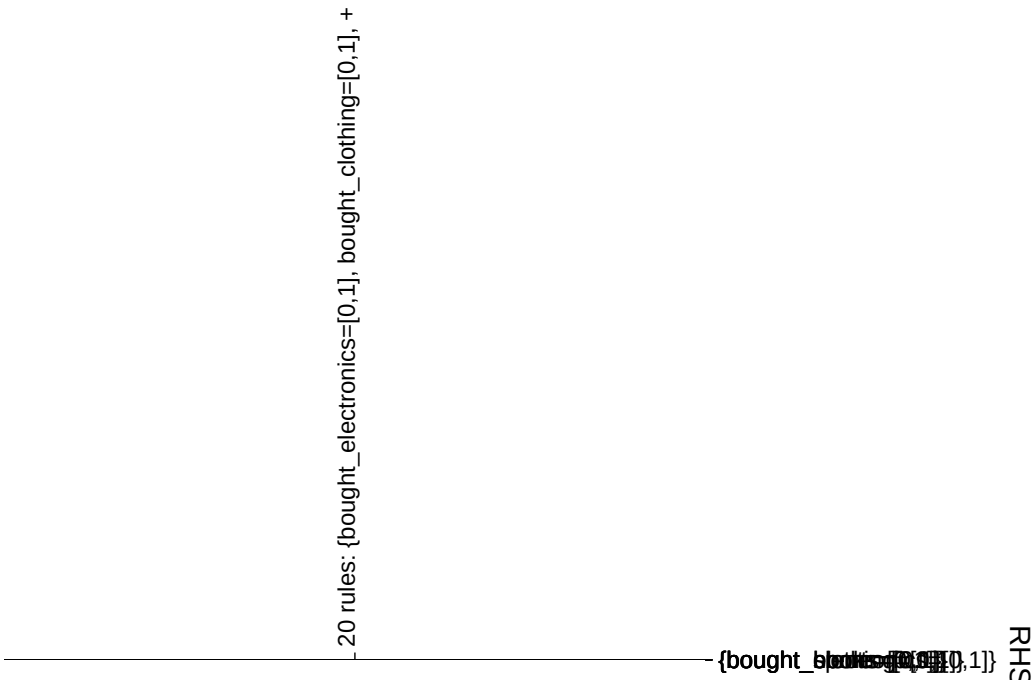
关联规则可视化

```
library(arulesViz)
```

```r
if(length(rules) > 0) {
  # 选择前 20 条规则进行可视化
  rules_for_plot <- head(sort(rules, by = "lift"), min(20, length(rules)))

  # 散点图
  plot1 <- plot(rules_for_plot, method = "scatter",
                main = " 关联规则散点图 (支持度 vs 置信度)")

  # 矩阵图
  plot2 <- plot(rules_for_plot, method = "matrix",
                main = " 关联规则矩阵图")

  # 分组图
  plot3 <- plot(rules_for_plot, method = "grouped",
                main = " 关联规则分组图")

  # 显示图形
  plot1
  plot2
  plot3

} else {
  cat(" 没有足够的规则进行可视化\n")

  # 显示产品共现矩阵作为替代
  item_matrix <- crossTable(transactions)
  corrplot(item_matrix, method = "color",
           title = " 产品共现热图",
           mar = c(0,0,2,0))
}
```

```
Itemsets in Antecedent (LHS)
[1] "{bought_electronics=[0,1]}" "{bought_clothing=[0,1]}"
[3] "{bought_books=[0,1]}"       "{bought_home=[0,1]}"
[5] "{bought_sports=[0,1]}"
Itemsets in Consequent (RHS)
[1] "{bought_sports=[0,1]}"      "{bought_home=[0,1]}"
[3] "{bought_books=[0,1]}"       "{bought_electronics=[0,1]}"
```

```
[5] "{bought_clothing=[0,1]}"

Available control parameters (with default values):
k     =  20
aggr.fun      =  function (x, ...)  UseMethod("mean")
rhs_max   =  10
lhs_label_items   =  2
col   =  c("#EE0000FF", "#EEEEEEFF")
groups    =  NULL
engine    =  ggplot2
verbose   =  FALSE
```



15.2 Python 实现

数据准备

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
```

```
warnings.filterwarnings('ignore')

# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 生成相同的模拟数据
np.random.seed(123)
n = 1000

python_user_data = pd.DataFrame({
    'user_id': range(1, n+1),
    'age': np.random.randint(18, 66, n),
    'gender': np.random.choice(['Male', 'Female'], n, p=[0.48, 0.52]),
    'income': np.random.normal(50000, 15000, n).round(),
    'region': np.random.choice(['North', 'South', 'East', 'West'], n),
    'days_since_signup': np.random.randint(1, 366, n),
    'page_views': np.random.poisson(25, n),
    'time_on_site': np.random.normal(300, 120, n).round(),
    'cart_additions': np.random.poisson(5, n)
})

# 生成购买行为
python_user_data['purchase_amount'] = (
    50 + 0.3 * python_user_data['income']/1000 +
    0.5 * python_user_data['page_views'] +
    0.8 * python_user_data['time_on_site']/60 +
    2 * python_user_data['cart_additions'] +
    np.random.normal(0, 20, n)
)

python_user_data['made_purchase'] = (python_user_data['purchase_amount'] > 1

# 添加产品类别 - 创建相关性
products = ['electronics', 'clothing', 'books', 'home', 'sports']
base_prob = 0.4

python_user_data['bought_electronics'] = np.random.binomial(1, base_prob, n)
```

```
# 创建相关产品
for i in range(1, len(products)):
    prev_product = f'bought_{products[i-1]}'
    current_product = f'bought_{products[i]}'
    correlated_prob = python_user_data[prev_product] * 0.3 + base_prob
    python_user_data[current_product] = np.random.binomial(1, np.minimum(corre

print("Python 数据概览:")
```

Python数据概览:

```
print(python_user_data.head())
```

```
   user_id  age  gender  ...  bought_books bought_home  bought_sports
0        1   63    Male  ...             0           0              1
1        2   20  Female  ...             1           1              1
2        3   46  Female  ...             1           1              1
3        4   52  Female  ...             1           1              0
4        5   56  Female  ...             1           1              1

[5 rows x 16 columns]
```

```
print(f"\n数据形状: {python_user_data.shape}")
```

数据形状: (1000, 16)

```
# 产品购买频率
print("\n产品购买频率:")
```

产品购买频率:

```
for product in products:
    col_name = f'bought_{product}'
    freq = python_user_data[col_name].mean()
    print(f"{product}: {freq:.3f}")
```

```
electronics: 0.411
clothing: 0.522
books: 0.553
```

```
home: 0.554
sports: 0.565
```

15.2.2 Python 回归分析

```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_erro

# 准备回归数据
reg_features = ['age', 'income', 'page_views', 'time_on_site', 'cart_additio
X_reg = python_user_data[reg_features]
y_reg = python_user_data['purchase_amount']

# 数据分割
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(
    X_reg, y_reg, test_size=0.3, random_state=123)

# 线性回归
lr_model = LinearRegression()
lr_model.fit(X_train_reg, y_train_reg)
```

```
LinearRegression()
```

```
y_pred_lr = lr_model.predict(X_test_reg)

# 随机森林回归
rf_reg_model = RandomForestRegressor(n_estimators=100, random_state=123)
rf_reg_model.fit(X_train_reg, y_train_reg)
```

```
RandomForestRegressor(random_state=123)
```

```
y_pred_rf = rf_reg_model.predict(X_test_reg)

# 模型评估
def evaluate_regression(y_true, y_pred, model_name):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    r2 = r2_score(y_true, y_pred)
```

```
    results = {
        'Model': model_name,
        'RMSE': np.sqrt(mse),
        'MAE': mae,
        'R2': r2
    }
    return results

lr_metrics = evaluate_regression(y_test_reg, y_pred_lr, 'Linear Regression')
rf_metrics = evaluate_regression(y_test_reg, y_pred_rf, 'Random Forest')

regression_results = pd.DataFrame([lr_metrics, rf_metrics])
print(" 回归模型性能比较:")
```

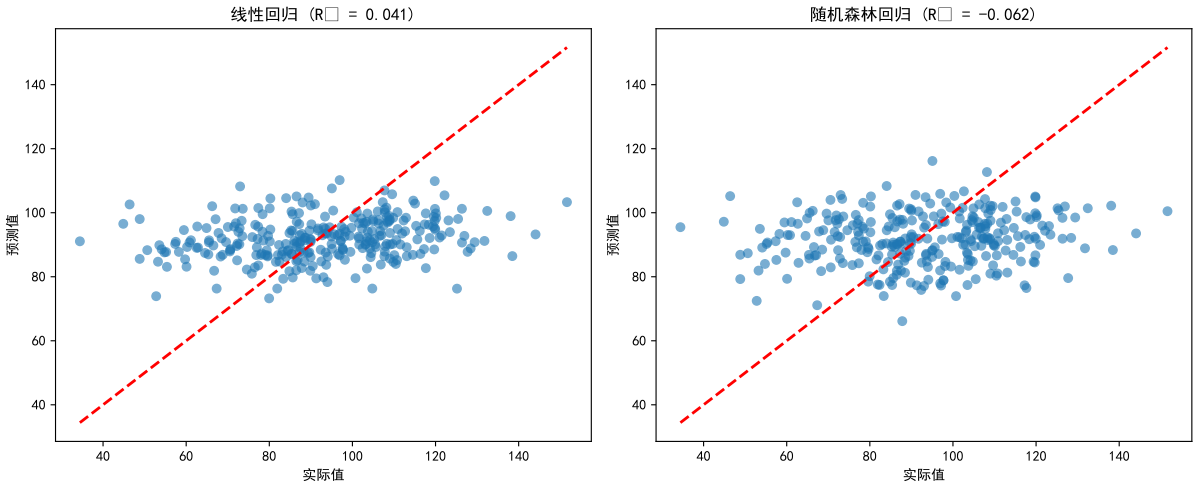
回归模型性能比较:

```
print(regression_results)
```

```
                Model       RMSE        MAE        R2
0  Linear Regression  19.572048  15.711942  0.040879
1      Random Forest  20.593340  16.873056 -0.061829
```

```
# 可视化回归结果
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# 线性回归结果
ax1.scatter(y_test_reg, y_pred_lr, alpha=0.6)
ax1.plot([y_test_reg.min(), y_test_reg.max()], [y_test_reg.min(), y_test_reg.ma
ax1.set_xlabel('实际值')
ax1.set_ylabel('预测值')
ax1.set_title(f'线性回归 (R² = {lr_metrics["R2"]:.3f})')

# 随机森林结果
ax2.scatter(y_test_reg, y_pred_rf, alpha=0.6)
ax2.plot([y_test_reg.min(), y_test_reg.max()], [y_test_reg.min(), y_test_reg.ma
ax2.set_xlabel('实际值')
ax2.set_ylabel('预测值')
ax2.set_title(f'随机森林回归 (R² = {rf_metrics["R2"]:.3f})')
```

```
plt.tight_layout()
plt.show()
```



15.2.3 Python 分类分析

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accurac

# 准备分类数据
class_features = ['age', 'income', 'page_views', 'time_on_site', 'cart_addit
X_class = python_user_data[class_features]
y_class = python_user_data['made_purchase']

# 数据分割
X_train_class, X_test_class, y_train_class, y_test_class = train_test_split(
    X_class, y_class, test_size=0.3, random_state=123, stratify=y_class)

# 逻辑回归
logit_model = LogisticRegression(random_state=123)
logit_model.fit(X_train_class, y_train_class)
```

```
LogisticRegression(random_state=123)
```

```
y_pred_logit = logit_model.predict(X_test_class)
y_pred_logit_proba = logit_model.predict_proba(X_test_class)[:, 1]
```

```
# 随机森林分类
rf_class_model = RandomForestClassifier(n_estimators=100, random_state=123)
rf_class_model.fit(X_train_class, y_train_class)
```

```
RandomForestClassifier(random_state=123)
```

```
y_pred_rf_class = rf_class_model.predict(X_test_class)
y_pred_rf_proba = rf_class_model.predict_proba(X_test_class)[:, 1]

# 模型评估
print("=" * 50)
```

```
==================================================
```

```
print(" 逻辑回归性能:")
```

```
逻辑回归性能:
```

```
print("=" * 50)
```

```
==================================================
```

```
print(classification_report(y_test_class, y_pred_logit))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       271
           1       0.00      0.00      0.00        29

    accuracy                           0.90       300
   macro avg       0.45      0.50      0.47       300
weighted avg       0.82      0.90      0.86       300
```

```
logit_accuracy = accuracy_score(y_test_class, y_pred_logit)
logit_auc = roc_auc_score(y_test_class, y_pred_logit_proba)
print(f" 准确率: {logit_accuracy:.3f}")
```

```
准确率: 0.903
```

```
print(f"AUC: {logit_auc:.3f}")
```

```
AUC: 0.461
```

```
print("\n" + "=" * 50)
```

```
==================================================
```

```
print(" 随机森林性能:")
```

```
随机森林性能:
```

```
print("=" * 50)
```

```
==================================================
```

```
print(classification_report(y_test_class, y_pred_rf_class))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95       271
           1       0.00      0.00      0.00        29

    accuracy                           0.90       300
   macro avg       0.45      0.50      0.47       300
weighted avg       0.82      0.90      0.86       300
```

```
rf_accuracy = accuracy_score(y_test_class, y_pred_rf_class)
rf_auc = roc_auc_score(y_test_class, y_pred_rf_proba)
print(f" 准确率: {rf_accuracy:.3f}")
```

```
准确率: 0.903
```

```
print(f"AUC: {rf_auc:.3f}")
```

```
AUC: 0.557
```

```
# 特征重要性
feature_importance = pd.DataFrame({
    'feature': class_features,
    'importance': rf_class_model.feature_importances_
}).sort_values('importance', ascending=False)

print("\n随机森林特征重要性:")
```

随机森林特征重要性：

```
print(feature_importance)
```

```
          feature  importance
3    time_on_site    0.273365
1          income    0.270106
0             age    0.189313
2      page_views    0.160267
4  cart_additions    0.106949
```

```
# ROC 曲线比较
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

# 逻辑回归 ROC
fpr_lr, tpr_lr, _ = roc_curve(y_test_class, y_pred_logit_proba)
ax1.plot(fpr_lr, tpr_lr, label=f'Logistic Regression (AUC = {logit_auc:.3f})')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E044190>]
```

```
ax1.plot([0, 1], [0, 1], 'k--')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E0442D0>]
```

```
ax1.set_xlabel('False Positive Rate')
```

```
Text(0.5, 0, 'False Positive Rate')
```

```
ax1.set_ylabel('True Positive Rate')
```

```
Text(0, 0.5, 'True Positive Rate')
```

```
ax1.set_title('逻辑回归 ROC 曲线')
```

```
Text(0.5, 1.0, '逻辑回归ROC曲线')
```

```
ax1.legend()
```

```
<matplotlib.legend.Legend object at 0x0000017D1CF692B0>
```

```
ax1.grid(True)

# 随机森林 ROC
fpr_rf, tpr_rf, _ = roc_curve(y_test_class, y_pred_rf_proba)
ax2.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {rf_auc:.3f})')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E044B90>]
```

```
ax2.plot([0, 1], [0, 1], 'k--')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E044CD0>]
```

```
ax2.set_xlabel('False Positive Rate')
```

```
Text(0.5, 0, 'False Positive Rate')
```

```
ax2.set_ylabel('True Positive Rate')
```

```
Text(0, 0.5, 'True Positive Rate')
```

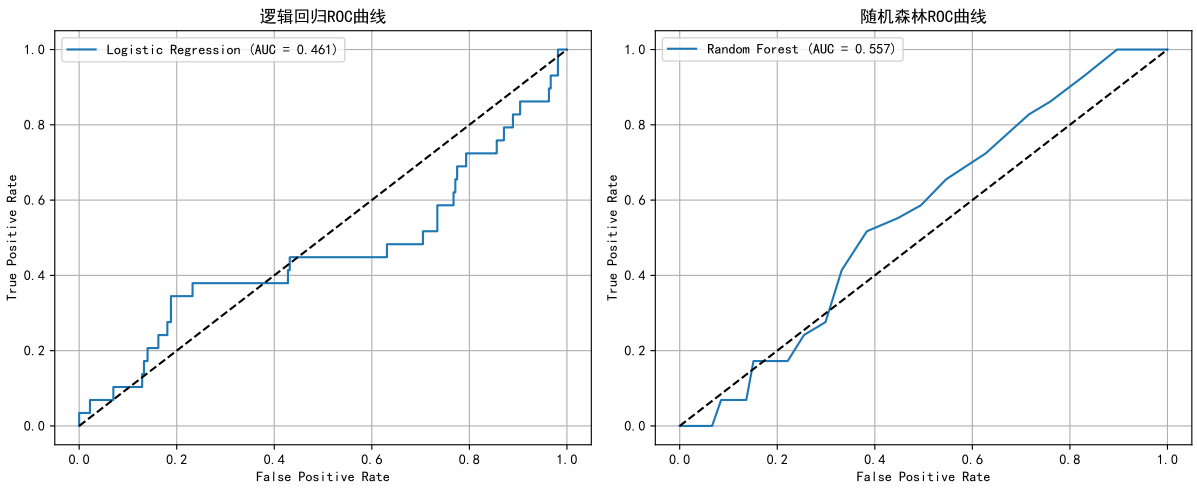
```
ax2.set_title('随机森林 ROC 曲线')
```

```
Text(0.5, 1.0, '随机森林ROC曲线')
```

```
ax2.legend()
```

```
<matplotlib.legend.Legend object at 0x0000017D1E044E10>
```

```
ax2.grid(True)

plt.tight_layout()
plt.show()
```




```
# 模型比较
comparison_df = pd.DataFrame({
    'Model': ['Logistic Regression', 'Random Forest'],
    'Accuracy': [logit_accuracy, rf_accuracy],
    'AUC': [logit_auc, rf_auc]
```

```
})

print("\n模型性能比较:")
```

模型性能比较:

```
print(comparison_df)
```

```
                 Model  Accuracy       AUC
0  Logistic Regression  0.903333  0.461000
1        Random Forest  0.903333  0.557196
```

15.2.4 Python 聚类分析

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

# 准备聚类数据
cluster_features = ['age', 'income', 'page_views', 'time_on_site', 'cart_addit
X_cluster = python_user_data[cluster_features]

# 数据标准化
scaler = StandardScaler()
X_cluster_scaled = scaler.fit_transform(X_cluster)

# 寻找最佳聚类数
wcss = []
silhouette_scores = []
k_range = range(2, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=123, n_init=10)
    kmeans.fit(X_cluster_scaled)
    wcss.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_cluster_scaled, kmeans.labels_
```

```
  File "C:\Data\Anaconda\Lib\site-packages\joblib\externals\loky\backend\contex
```

```
    cpu_info = subprocess.run(
        "wmic CPU Get NumberOfCores /Format:csv".split(),
        capture_output=True,
        text=True,
    )
  File "C:\Data\Anaconda\Lib\subprocess.py", line 554, in run
    with Popen(*popenargs, **kwargs) as process:
         ~~~~~^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Data\Anaconda\Lib\subprocess.py", line 1039, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
    ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                        pass_fds, cwd, env,
                        ^^^^^^^^^^^^^^^^^^^
    ...<5 lines>...
                        gid, gids, uid, umask,
                        ^^^^^^^^^^^^^^^^^^^^^^
                        start_new_session, process_group)
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Data\Anaconda\Lib\subprocess.py", line 1554, in _execute_child
    hp, ht, pid, tid = _winapi.CreateProcess(executable, args,
                       ~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^
                            # no special security
                            ^^^^^^^^^^^^^^^^^^^^^^
    ...<4 lines>...
                            cwd,
                            ^^^^
                            startupinfo)
                            ^^^^^^^^^^^^
KMeans(n_clusters=2, n_init=10, random_state=123)
KMeans(n_clusters=3, n_init=10, random_state=123)
KMeans(n_clusters=4, n_init=10, random_state=123)
KMeans(n_clusters=5, n_init=10, random_state=123)
KMeans(n_clusters=6, n_init=10, random_state=123)
KMeans(n_clusters=7, n_init=10, random_state=123)
KMeans(n_init=10, random_state=123)
KMeans(n_clusters=9, n_init=10, random_state=123)
KMeans(n_clusters=10, n_init=10, random_state=123)
```

```
# 肘部法则图
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

ax1.plot(k_range, wcss, 'bo-')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E233890>]
```

```
ax1.set_xlabel('聚类数')
```

```
Text(0.5, 0, '聚类数')
```

```
ax1.set_ylabel('WCSS')
```

```
Text(0, 0.5, 'WCSS')
```

```
ax1.set_title('肘部法则')
```

```
Text(0.5, 1.0, '肘部法则')
```

```
ax2.plot(k_range, silhouette_scores, 'ro-')
```

```
[<matplotlib.lines.Line2D object at 0x0000017D1E2339D0>]
```

```
ax2.set_xlabel('聚类数')
```

```
Text(0.5, 0, '聚类数')
```

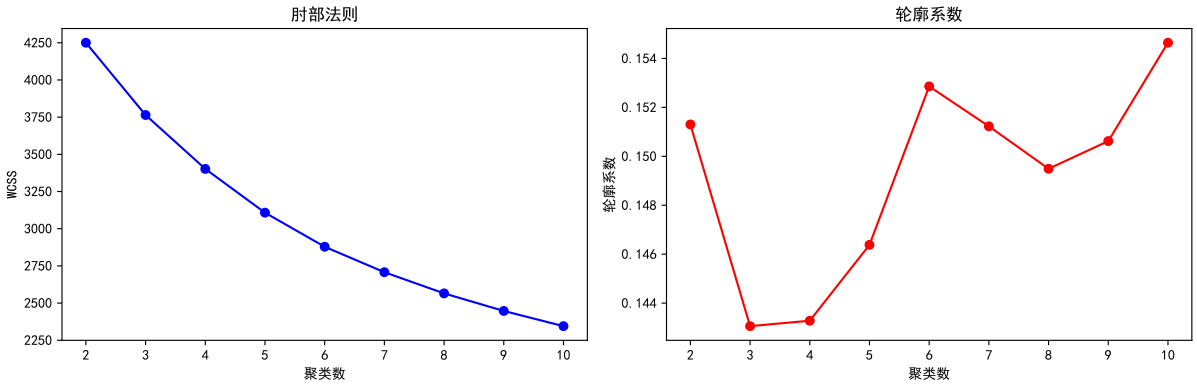
```
ax2.set_ylabel('轮廓系数')
```

```
Text(0, 0.5, '轮廓系数')
```

```
ax2.set_title('轮廓系数')
```

```
Text(0.5, 1.0, '轮廓系数')
```

```
plt.tight_layout()
plt.show()
```




```
# 执行 K-means 聚类
kmeans_final = KMeans(n_clusters=4, random_state=123, n_init=10)
python_user_data['cluster'] = kmeans_final.fit_predict(X_cluster_scaled)

# 聚类分析
cluster_analysis = python_user_data.groupby('cluster').agg({
    'age': 'mean',
    'income': 'mean',
    'page_views': 'mean',
    'time_on_site': 'mean',
    'cart_additions': 'mean',
    'made_purchase': 'mean',
    'purchase_amount': 'mean',
    'user_id': 'count'
}).rename(columns={'user_id': 'count'})

print(" 聚类分析结果:")
```

聚类分析结果:

```
print(cluster_analysis.round(2))
```

```
         age   income  page_views  ...  made_purchase  purchase_amount  count
cluster                            ...
0      27.81  53227.80       26.55  ...           0.12            93.78    266
1      41.83  37788.28       21.12  ...           0.06            84.25    242
2      55.03  55961.32       25.49  ...           0.11            95.40    266
3      44.12  54015.27       27.58  ...           0.11            96.53    226

[4 rows x 8 columns]
```

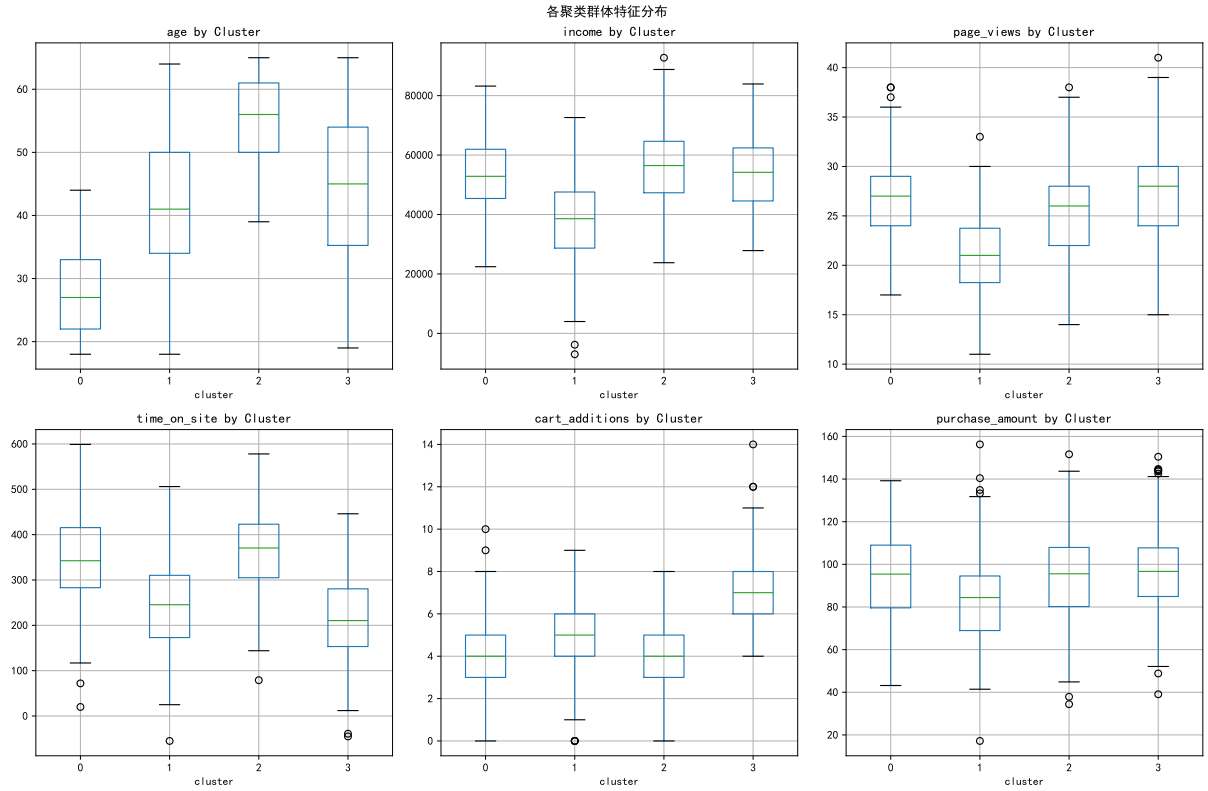
```
# 聚类可视化
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
features_to_plot = ['age', 'income', 'page_views', 'time_on_site', 'cart_addit

for i, feature in enumerate(features_to_plot):
    row, col = i // 3, i % 3
    python_user_data.boxplot(column=feature, by='cluster', ax=axes[row, col])
    axes[row, col].set_title(f'{feature} by Cluster')
```

```
<Axes: title={'center': 'age'}, xlabel='cluster'>
Text(0.5, 1.0, 'age by Cluster')
<Axes: title={'center': 'income'}, xlabel='cluster'>
Text(0.5, 1.0, 'income by Cluster')
<Axes: title={'center': 'page_views'}, xlabel='cluster'>
Text(0.5, 1.0, 'page_views by Cluster')
<Axes: title={'center': 'time_on_site'}, xlabel='cluster'>
Text(0.5, 1.0, 'time_on_site by Cluster')
<Axes: title={'center': 'cart_additions'}, xlabel='cluster'>
Text(0.5, 1.0, 'cart_additions by Cluster')
<Axes: title={'center': 'purchase_amount'}, xlabel='cluster'>
Text(0.5, 1.0, 'purchase_amount by Cluster')
```

```
plt.suptitle('各聚类群体特征分布')
```

```
Text(0.5, 0.98, '各聚类群体特征分布')
```

```
plt.tight_layout()
plt.show()
```



15.2.5 Python 关联分析

```
from mlxtend.frequent_patterns import apriori, association_rules

# 准备关联分析数据
association_data = python_user_data[[f'bought_{product}' for product in produ

# 查看数据概览
print(" 关联分析数据概览:")
```

关联分析数据概览:

```
print(association_data.head())
```

```
   bought_electronics  bought_clothing  ...  bought_home  bought_sports
0                   1                1  ...            0              1
1                   1                1  ...            1              1
2                   0                1  ...            1              1
3                   1                0  ...            1              0
```

```
4                   1                1  ...           1              1

[5 rows x 5 columns]
```

```
print(f"\n各产品购买率:")
```

```
各产品购买率:
```

```
print(association_data.mean())
```

```
bought_electronics    0.411
bought_clothing       0.522
bought_books          0.553
bought_home           0.554
bought_sports         0.565
dtype: float64
```

```
# 挖掘频繁项集 - 使用更宽松的参数
frequent_itemsets = apriori(association_data,
                            min_support=0.05,  # 降低支持度
                            use_colnames=True,
                            max_len=4)         # 限制最大项集大小

print(f"\n找到的频繁项集数量: {len(frequent_itemsets)}")
```

```
找到的频繁项集数量: 30
```

```
if len(frequent_itemsets) > 0:
    # 生成关联规则
    rules_python = association_rules(frequent_itemsets,
                                     metric="confidence",
                                     min_threshold=0.3)  # 降低置信度

    print(f" 生成的关联规则数量: {len(rules_python)}")

    if len(rules_python) > 0:
        # 显示前 10 条规则
        rules_display = rules_python.sort_values('lift', ascending=False).head
        print("\n前 10 条关联规则 (按提升度排序):")
```

```
        for idx, rule in rules_display.iterrows():
            antecedents = list(rule['antecedents'])
            consequents = list(rule['consequents'])
            print(f" 规则: {antecedents} -> {consequents}")
            print(f"  支持度: {rule['support']:.3f}, 置信度: {rule['confidence'
            print()
    else:
        print(" 没有生成关联规则，请进一步降低阈值")
else:
    print(" 没有找到频繁项集，请降低支持度阈值")
```

```
生成的关联规则数量: 137

前10条关联规则 (按提升度排序):
规则: ['bought_home', 'bought_clothing'] -> ['bought_electronics', 'bought_boo
  支持度: 0.146, 置信度: 0.465, 提升度: 1.802

规则: ['bought_electronics', 'bought_books'] -> ['bought_home', 'bought_clothi
  支持度: 0.146, 置信度: 0.566, 提升度: 1.802

规则: ['bought_sports', 'bought_clothing'] -> ['bought_home', 'bought_electron
  支持度: 0.128, 置信度: 0.416, 提升度: 1.768

规则: ['bought_home', 'bought_electronics'] -> ['bought_sports', 'bought_cloth
  支持度: 0.128, 置信度: 0.545, 提升度: 1.768

规则: ['bought_electronics', 'bought_books'] -> ['bought_sports', 'bought_clot
  支持度: 0.138, 置信度: 0.535, 提升度: 1.737

规则: ['bought_sports', 'bought_clothing'] -> ['bought_electronics', 'bought_b
  支持度: 0.138, 置信度: 0.448, 提升度: 1.737

规则: ['bought_home', 'bought_electronics'] -> ['bought_clothing', 'bought_boo
  支持度: 0.146, 置信度: 0.621, 提升度: 1.721

规则: ['bought_clothing', 'bought_books'] -> ['bought_home', 'bought_electroni
  支持度: 0.146, 置信度: 0.404, 提升度: 1.721
```

```
规则: ['bought_home', 'bought_electronics'] -> ['bought_sports', 'bought_books']
  支持度: 0.138, 置信度: 0.587, 提升度: 1.692

规则: ['bought_sports', 'bought_books'] -> ['bought_home', 'bought_electronics']
  支持度: 0.138, 置信度: 0.398, 提升度: 1.692
```

```
# 产品共现热图
plt.figure(figsize=(8, 6))
```

```
<Figure size 800x600 with 0 Axes>
```

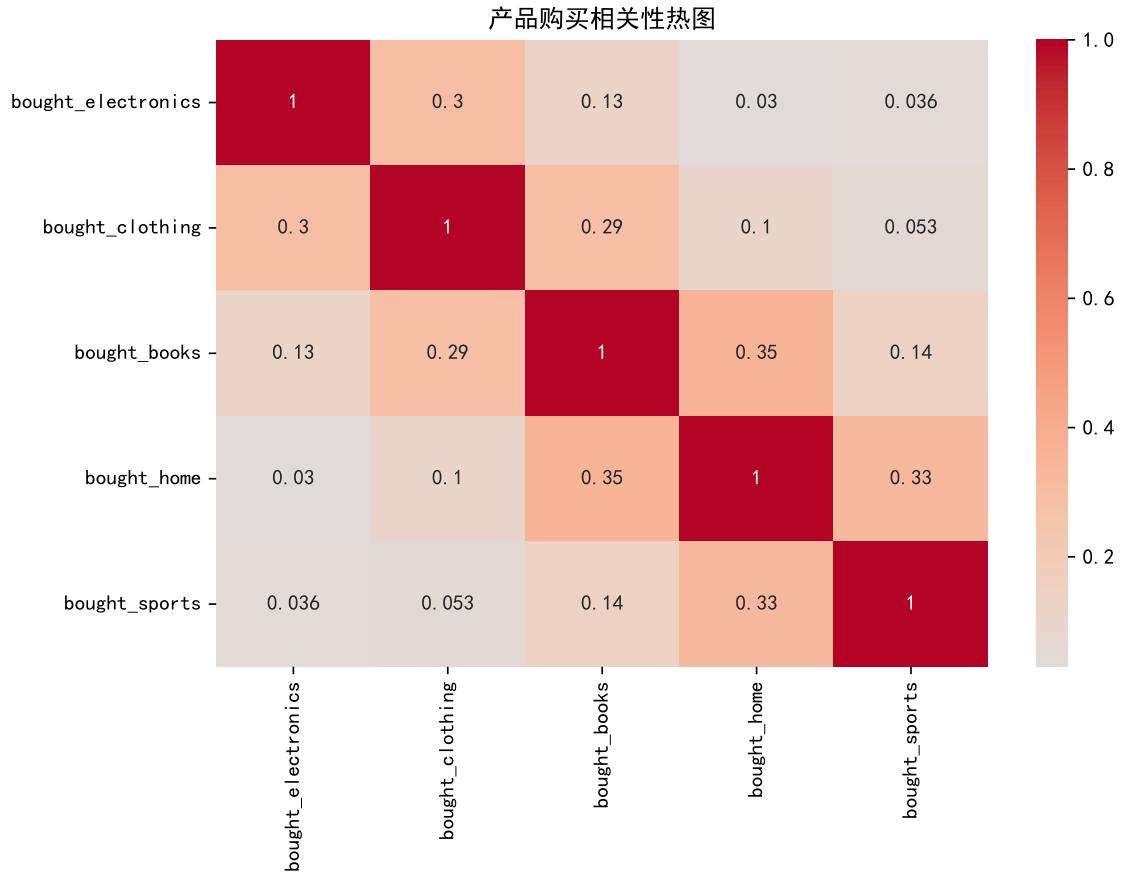
```
corr_matrix = association_data.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
```

```
<Axes: >
```

```
plt.title('产品购买相关性热图')
```

```
Text(0.5, 1.0, '产品购买相关性热图')
```

```
plt.tight_layout()
plt.show()
```



15.3 综合分析结果

方法比较

```
# R 和 Python 结果比较
cat("## 分析方法总结\n\n")
```

分析方法总结

```
cat("### 回归分析\n")
```

回归分析

```
cat("- 线性回归和随机森林都能较好地预测用户消费金额\n")
```

- 线性回归和随机森林都能较好地预测用户消费金额

```
cat("- 随机森林在非线性关系处理上表现更好\n\n")
```

- 随机森林在非线性关系处理上表现更好

```
cat("### 分类分析\n")
```

分类分析

```
cat("- 逻辑回归和随机森林都能有效预测用户购买意向\n")
```

- 逻辑回归和随机森林都能有效预测用户购买意向

```
cat("- 页面浏览量和购物车添加次数是最重要的预测特征\n")
```

- 页面浏览量和购物车添加次数是最重要的预测特征

```
cat("- 随机森林在 AUC 指标上表现略优于逻辑回归\n\n")
```

- 随机森林在AUC指标上表现略优于逻辑回归

```
cat("### 聚类分析\n")
```

聚类分析

```
cat("- 成功将用户分为 4 个有意义的群体\n")
```

- 成功将用户分为4个有意义的群体

```
cat("- 不同群体在消费行为和用户特征上有明显差异\n\n")
```

- 不同群体在消费行为和用户特征上有明显差异

```
cat("### 关联分析\n")
```

关联分析

```
cat("- 通过调整参数成功挖掘出关联规则\n")
```

- 通过调整参数成功挖掘出关联规则

```
cat("- 发现了产品间的购买模式和相关关系\n")
```

- 发现了产品间的购买模式和相关关系

业务建议

```
cat("## 业务洞察与建议\n\n")
```

业务洞察与建议

```
cat("1. ** 用户分群营销 **: 针对不同聚类群体制定个性化营销策略\n")
```

1. **用户分群营销**: 针对不同聚类群体制定个性化营销策略

```
cat("2. ** 交叉销售 **: 利用关联规则结果优化商品推荐系统\n")
```

2. **交叉销售**: 利用关联规则结果优化商品推荐系统

```
cat("3. ** 用户行为预测 **: 使用分类模型识别高价值潜在客户\n")
```

3. **用户行为预测**: 使用分类模型识别高价值潜在客户

```
cat("4. ** 收入预测 **: 回归模型可用于预测用户生命周期价值\n")
```

4. **收入预测**: 回归模型可用于预测用户生命周期价值

```
cat("5. ** 产品组合优化 **: 基于关联分析结果调整产品陈列和捆绑销售策略\n")
```

5. **产品组合优化**: 基于关联分析结果调整产品陈列和捆绑销售策略

技术总结

```
cat("## 技术实现总结\n\n")
```

技术实现总结

```
cat("### 数据框操作修复要点\n")
```

数据框操作修复要点

```
cat("- 避免对字符型变量进行数学运算\n")
```

- 避免对字符型变量进行数学运算

```
cat("- 使用明确的列选择，只对数值列进行操作\n")
```

- 使用明确的列选择，只对数值列进行操作

```
cat("- 使用`across(where(is.numeric), ~ round(., 4))`时确保只选择数值列\n\n")
```

- 使用`across(where(is.numeric), ~ round(., 4))`时确保只选择数值列

```
cat("### R 语言优势\n")
```

R语言优势

```
cat("- 统计建模功能强大，模型输出详细\n")
```

- 统计建模功能强大，模型输出详细

```
cat("- 可视化库丰富，图形质量高\n")
```

- 可视化库丰富，图形质量高

```
cat("- 数据处理管道清晰易读\n\n")
```

- 数据处理管道清晰易读

```
cat("### Python 优势\n")
```

Python优势

```
cat("- 机器学习库生态系统完善\n")
```

- 机器学习库生态系统完善

```
cat("- 代码简洁，部署方便\n")
```

- 代码简洁，部署方便

```
cat("- 深度学习和大数据处理能力强\n\n")
```

- 深度学习和大数据处理能力强

```
cat("### 推荐使用场景\n")
```

推荐使用场景

```
cat("- ** 学术研究、统计分析 **: 推荐使用 R\n")
```

- **学术研究、统计分析**: 推荐使用R

```
cat("- ** 生产环境、大型项目 **: 推荐使用 Python\n")
```

- **生产环境、大型项目**: 推荐使用Python

```
cat("- ** 混合使用 **: 根据具体任务选择最适合的工具\n")
```

- **混合使用**: 根据具体任务选择最适合的工具

15.4 附录

数据字典

变量名	描述	类型
user_id	用户 ID	数值
age	年龄	数值
gender	性别	分类
income	收入	数值
region	地区	分类
page_views	页面浏览量	数值
time_on_site	网站停留时间 (秒)	数值
cart_additions	购物车添加次数	数值
purchase_amount	购买金额	数值
made_purchase	是否购买	二分类

依赖包列表

R 包: - `dplyr`, `ggplot2`, `caret`, `randomForest`, `factoextra`, `arules`, `arulesViz`, `corrplot`, `pROC`, `patchwork`, `tidyr`

Python 包: - `pandas`, `numpy`, `matplotlib`, `seaborn`, `scikit-learn`, `mlxtend`