数据预处理

异常值处理 Python 实习

Outlier Handling - Hands-on Practice

授课教师：李阳源

2026-03-03

>>> 实习环境：Python 3.x / Jupyter Notebook 或 Google Colab <<<

目录（mù lù）

实习 1：环境准备与数据生成

实习 2：基础统计量计算

实习 3：Z-分数方法检测异常值

实习 4：IQR 方法检测异常值

实习 5：可视化异常值检测（箱线图 / 直方图 / 散点图）

实习 6：异常值处理方法对比

实习 7：综合案例 - 信用卡欺诈检测模拟

实习 8：课后练习

实习 1：环境准备与数据生成

首先，我们需要导入（dǎo rù）必要的 Python 库，并生成一组包含异常值的示例数据。

Step 1-1：导入库

我们需要用到以下几个库：

numpy - 数值计算

pandas - 数据处理

matplotlib - 数据可视化（kě shì huà）

scipy - 科学计算（包含 Z-score 函数）

```
# === Step 1-1: 导入必要的库 ===
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats


# 设置中文显示（如果需要）
plt.rcParams['font.sans-serif'] = ['SimHei', 'Arial Unicode MS']
plt.rcParams['axes.unicode_minus'] = False


print('所有库导入成功！')
```

>>> 老师提示：如果在 Google Colab 中运行，这些库都已经预装好了，直接运行就行。

Step 1-2：生成示例数据

我们先用课堂上讲过的简单数据集来练习，然后再生成一个更大的数据集。

```
# === Step 1-2: 生成示例数据 ===
```

```
# (1) 课堂示例数据（和幻灯片一样的数据）
simple_data = np.array([10, 12, 15, 18, 20, 22, 25, 28, 30, 100])
print('课堂示例数据：', simple_data)

# (2) 生成更大的数据集（模拟真实场景）
np.random.seed(42)  # 设置随机种子，保证每次运行结果一致

# 正常数据：均值50，标准差10，共100个
normal_data = np.random.normal(loc=50, scale=10, size=100)

# 异常值：5个明显偏大的值，5个明显偏小的值
outliers_high = np.array([120, 135, 150, 110, 125])
outliers_low = np.array([-20, -15, -30, -10, -25])

# 合并所有数据
full_data = np.concatenate([normal_data, outliers_high, outliers_low])

# 创建 DataFrame
df = pd.DataFrame({
    'value': full_data,
    'type': ['normal']*100 + ['outlier_high']*5 + ['outlier_low']*5
})

print(f'总数据量：{len(df)}')
print(f'正常数据：{len(df[df["type"]=="normal"])}')
print(f'高异常值：{len(df[df["type"]=="outlier_high"])}')
print(f'低异常值：{len(df[df["type"]=="outlier_low"])}')
```

预期输出（yù qī shū chū）：

```
课堂示例数据： [ 10  12  15  18  20  22  25  28  30 100]
总数据量：110
正常数据：100
高异常值：5
低异常值：5
```

>>> 同学们注意：我们故意加入了10个异常值，等一下我们用各种方法来看能不能把它们找出来。

实习 2：基础统计量计算

在检测异常值之前，我们先来看看数据的基本统计信息。

这一步非常重要，因为只有了解了数据的基本特征，才能更好地判断哪些值是异常的。

Step 2-1：用课堂示例数据计算

```
# === Step 2-1: 课堂示例数据的统计量 ===


data = simple_data  # [10, 12, 15, 18, 20, 22, 25, 28, 30, 100]


# 手动计算（和幻灯片一致）
n = len(data)
mean = np.mean(data)
std_pop = np.std(data)           # 总体标准差（除以 N）
std_sample = np.std(data, ddof=1)  # 样本标准差（除以 N-1）
median = np.median(data)
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1


print('='*50)
print('       课堂示例数据 统计摘要')
print('='*50)
print(f'样本数量 (N)    : {n}')
print(f'平均值 (mu)     : {mean:.2f}')
print(f'总体标准差 (sigma): {std_pop:.2f}')
print(f'样本标准差       : {std_sample:.2f}')
print(f'中位数 (Q2)     : {median:.2f}')
print(f'Q1 (25%)        : {q1:.2f}')
print(f'Q3 (75%)        : {q3:.2f}')
print(f'IQR             : {iqr:.2f}')
print('='*50)
```

预期输出：

```
==================================================
       课堂示例数据 统计摘要
==================================================
样本数量 (N)    : 10
```

```
平均值 (mu)     : 28.00
总体标准差 (sigma): 24.78
样本标准差       : 26.12
中位数 (Q2)     : 21.00
Q1 (25%)        : 15.50
Q3 (75%)        : 27.25
IQR             : 11.75
==================================================
```

>>> 同学们思考：平均值是28，但中位数只有21。两者差距很大，说明什么？

>>> 说明数据被那个100拉偏了！这就是异常值对统计量的影响。

Step 2-2：用完整数据集计算

```
# === Step 2-2: 完整数据集的 describe ===


print('完整数据集 描述统计：')
print(df['value'].describe().round(2))
print()


# 对比：只看正常数据的统计量
print('仅正常数据 描述统计：')
print(df[df['type']=='normal']['value'].describe().round(2))
```

>>> 对比两组统计量，看看异常值是怎么影响平均值和标准差的。

实习 3：Z-分数方法检测异常值

现在我们用 Z-分数方法来检测异常值。

还记得公式吗？Z = (X - mu) / sigma，当 |Z| > 3 时判定为异常值。

Step 3-1：手动计算 Z-分数（课堂示例）

```
# === Step 3-1: 手动计算 Z-分数 ===


data = simple_data
mean = np.mean(data)
std = np.std(data)  # 总体标准差


# 计算每个数据点的 Z-分数
z_scores = (data - mean) / std


# 创建结果表格
result = pd.DataFrame({
    '数据 (X)': data,
    '差值 (X-mu)': data - mean,
    'Z-分数': z_scores.round(3),
    '是否异常 (|Z|>3)': ['是' if abs(z) > 3 else '否' for z in z_scores]
})


print('Z-分数计算结果：')
print(result.to_string(index=False))
print()
print(f'检测到的异常值数量：{sum(abs(z_scores) > 3)}')
```

> >> 注意：100的Z-分数约为2.905，没有超过3的阈值（yù zhí），所以Z-分数方法在这个小数据集上没能检测到异常值。

Step 3-2：用 scipy 计算 Z-分数（完整数据）

```
# === Step 3-2: 用 scipy.stats.zscore 计算 ===
```

```
# scipy 的 zscore 函数可以直接计算
df['z_score'] = stats.zscore(df['value'])


# 标记异常值（|Z| > 3）
df['z_outlier'] = abs(df['z_score']) > 3


# 查看结果
print('Z-score 方法检测结果：')
print(f'  检测到异常值数量：{df["z_outlier"].sum()}')
print(f'  实际异常值数量：{len(df[df["type"]!="normal"])}')
print()


# 看看被检测为异常的数据
print('被检测为异常的数据点：')
print(df[df['z_outlier']==True][['value','type','z_score']].to_string())
```

>>> 思考题：Z-score 方法找到了几个异常值？有没有遗漏（yí lòu）的？为什么？

Step 3-3：可视化 Z-分数

```
# === Step 3-3: Z-score 可视化 ===


fig, ax = plt.subplots(figsize=(12, 5))


colors = ['red' if outlier else 'steelblue'
          for outlier in df['z_outlier']]


ax.scatter(range(len(df)), df['z_score'], c=colors, alpha=0.6, s=30)
ax.axhline(y=3, color='red', linestyle='--', label='Z=3 (上界)')
ax.axhline(y=-3, color='red', linestyle='--', label='Z=-3 (下界)')
ax.axhline(y=0, color='gray', linestyle='-', alpha=0.3)


ax.set_xlabel('数据索引')
ax.set_ylabel('Z-score')
ax.set_title('Z-score 异常值检测结果')
ax.legend()


plt.tight_layout()
plt.show()
```

>>> 图中红色的点是被Z-score方法判定为异常的数据，蓝色的是正常数据。红色虚线是±3

的阈值线。

实习 4：IQR 方法检测异常值

接下来我们用 IQR 方法。

IQR = Q3 - Q1，异常值边界是 Q1-1.5*IQR 和 Q3+1.5*IQR。

Step 4-1：手动计算 IQR（课堂示例）

```
# === Step 4-1: 手动 IQR 计算 ===


data = simple_data
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1


lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr


print('=== IQR 方法计算过程 ===')
print(f'Q1 (25%)     : {q1:.2f}')
print(f'Q3 (75%)     : {q3:.2f}')
print(f'IQR          : {iqr:.2f}')
print(f'下界 (Lower) : {lower_bound:.2f}')
print(f'上界 (Upper)  : {upper_bound:.2f}')
print()


# 检测异常值
outliers = data[(data < lower_bound) | (data > upper_bound)]
print(f'检测到的异常值：{outliers}')
```

>>> 看！用IQR方法，100被成功检测为异常值了。对比Z-score方法（没检测到），IQR方法更加稳健（wěn jiàn）。

Step 4-2：完整数据集的 IQR 检测

```
# === Step 4-2: 完整数据的 IQR 检测 ===


q1 = df['value'].quantile(0.25)
```

```
q3 = df['value'].quantile(0.75)
iqr = q3 - q1


lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr


df['iqr_outlier'] = (df['value'] < lower) | (df['value'] > upper)


print(f'IQR 边界: [{lower:.2f}, {upper:.2f}]')
print(f'IQR 检测到异常值数量：{df["iqr_outlier"].sum()}')
print(f'实际异常值数量：{len(df[df["type"]!="normal"])}')
print()


# 对比两种方法
print('=== 方法对比 ===')
print(f'Z-score 检测到：{df["z_outlier"].sum()} 个')
print(f'IQR 检测到     ：{df["iqr_outlier"].sum()} 个')
print(f'实际异常值     ：10 个')
```

>>> 比较一下：哪种方法检测到了更多的异常值？哪种更准确？

实习 5：可视化异常值检测

可视化是检测异常值最直观（zhí guān）的方法。我们来画三种图。

Step 5-1：箱线图（Box Plot）

```
# === Step 5-1: 箱线图 ===

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# 左图：课堂示例数据
axes[0].boxplot(simple_data, vert=True)
axes[0].set_title('课堂示例数据 箱线图')
axes[0].set_ylabel('值')

# 右图：完整数据集
axes[1].boxplot(df['value'], vert=True)
axes[1].set_title('完整数据集 箱线图')
axes[1].set_ylabel('值')

plt.tight_layout()
plt.show()

print('提示：箱子外面的圆点就是异常值！')
```

>>> 同学们观察：箱线图中的小圆点就是异常值。它们明显远离（yuǎn lí）箱子的位置。

Step 5-2：直方图（Histogram）

```
# === Step 5-2: 直方图 ===

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# 左图：课堂数据
axes[0].hist(simple_data, bins=10, color='steelblue',
             edgecolor='white', alpha=0.7)
axes[0].set_title('课堂示例数据 直方图')
```

```
axes[0].set_xlabel('值')
axes[0].set_ylabel('频次')

# 右图：完整数据
axes[1].hist(df['value'], bins=20, color='steelblue',
             edgecolor='white', alpha=0.7)
axes[1].axvline(x=upper, color='red', linestyle='--',
                label=f'IQR上界={upper:.1f}')
axes[1].axvline(x=lower, color='red', linestyle='--',
                label=f'IQR下界={lower:.1f}')
axes[1].set_title('完整数据集 直方图')
axes[1].set_xlabel('值')
axes[1].set_ylabel('频次')
axes[1].legend()

plt.tight_layout()
plt.show()
```

>>> 直方图中，正常数据集中在中间形成一个高峰，异常值在两端形成孤立的小柱子。

Step 5-3：散点图（Scatter Plot）

```
# === Step 5-3: 散点图（带颜色标记）===

fig, ax = plt.subplots(figsize=(12, 6))

# 用不同颜色标记正常值和异常值
colors = {'normal': 'steelblue',
          'outlier_high': 'red',
          'outlier_low': 'red'}

for t in df['type'].unique():
    subset = df[df['type'] == t]
    ax.scatter(subset.index, subset['value'],
               c=colors[t], label=t, alpha=0.6, s=40)

# 添加 IQR 边界线
ax.axhline(y=upper, color='orange', linestyle='--',
           label=f'IQR 上界 ({upper:.1f})')
ax.axhline(y=lower, color='orange', linestyle='--',
           label=f'IQR 下界 ({lower:.1f})')
```

```
ax.set_xlabel('数据索引')
ax.set_ylabel('值')
ax.set_title('数据分布散点图（红色=异常值）')
ax.legend()


plt.tight_layout()
plt.show()
```

>>> 红色的点是我们故意加入的异常值，橙色虚线是IQR边界。正常数据在两条线之间。

实习 6：异常值处理方法对比

检测到异常值后，我们来实习四种不同的处理方法，并比较（bǐ jiào）它们的效果。

Step 6-1：方法一 - 删除（Deletion）

```
# === Step 6-1: 删除异常值 ===


df_deleted = df[~df['iqr_outlier']].copy()


print('删除异常值后：')
print(f'  原始数据量：{len(df)}')
print(f'  删除后数据量：{len(df_deleted)}')
print(f'  删除了 {len(df) - len(df_deleted)} 个数据点')
```

Step 6-2：方法二 - 均值替换（Mean Imputation）

```
# === Step 6-2: 用均值替换异常值 ===


df_mean = df.copy()
mean_val = df_mean[~df_mean['iqr_outlier']]['value'].mean()
df_mean.loc[df_mean['iqr_outlier'], 'value'] = mean_val


print(f'用均值 {mean_val:.2f} 替换了异常值')
```

Step 6-3：方法三 - 中位数替换（Median Imputation）

```
# === Step 6-3: 用中位数替换异常值 ===


df_median = df.copy()
median_val = df_median[~df_median['iqr_outlier']]['value'].median()
df_median.loc[df_median['iqr_outlier'], 'value'] = median_val


print(f'用中位数 {median_val:.2f} 替换了异常值')
```

Step 6-4：方法四 - 对数转换（Log Transformation）

```
# === Step 6-4: 对数转换 ===
# 注意：对数转换要求数据为正数，需要先做偏移

df_log = df.copy()
min_val = df_log['value'].min()
shift = abs(min_val) + 1 if min_val <= 0 else 0
df_log['value_log'] = np.log1p(df_log['value'] + shift)

print(f'偏移量: {shift:.2f}')
print(f'转换前范围: [{df["value"].min():.2f}, {df["value"].max():.2f}]')
print(f'转换后范围: [{df_log["value_log"].min():.2f},
{df_log["value_log"].max():.2f}]')
```

Step 6-5：四种方法效果对比

```
# === Step 6-5: 对比四种方法 ===

fig, axes = plt.subplots(2, 2, figsize=(14, 10))

datasets = [
    ('原始数据', df['value']),
    ('删除异常值', df_deleted['value']),
    ('均值替换', df_mean['value']),
    ('中位数替换', df_median['value']),
]

for ax, (title, data) in zip(axes.flat, datasets):
    ax.hist(data, bins=20, color='steelblue',
            edgecolor='white', alpha=0.7)
    ax.set_title(title)
    ax.set_xlabel('值')
    ax.set_ylabel('频次')
    mean_v = data.mean()
    ax.axvline(x=mean_v, color='red', linestyle='--')
    ax.text(mean_v, ax.get_ylim()[1]*0.9,
            f'  mean={mean_v:.1f}', color='red')

plt.suptitle('四种处理方法效果对比', fontsize=16, fontweight='bold')
plt.tight_layout()
plt.show()

# 打印统计对比表
```

```
print('\n=== 统计量对比 ===')
print(f'{"方法":<15}{"平均值":>10}{"标准差":>10}{"中位数":>10}')
print('-'*45)
for name, data in datasets:

print(f'{name:<15}{data.mean():>10.2f}{data.std():>10.2f}{data.median():>10.2f}')
```

>>> 对比不同方法对平均值和标准差的影响。哪种方法让数据分布最接近正态分布?

实习 7：综合案例 - 信用卡欺诈检测模拟

现在我们来做一个更接近实际的综合案例——模拟信用卡交易数据的异常检测。

Step 7-1：生成模拟信用卡交易数据

```
# === Step 7-1: 生成模拟数据 ===


np.random.seed(123)


n_normal = 1000  # 正常交易
n_fraud = 20     # 欺诈交易


# 正常交易：金额 50-500，频率 1-10次/天
normal_amount = np.random.exponential(scale=100, size=n_normal) + 20
normal_freq = np.random.poisson(lam=3, size=n_normal) + 1


# 欺诈交易：金额异常大 或 频率异常高
fraud_amount = np.random.uniform(2000, 10000, size=n_fraud)
fraud_freq = np.random.poisson(lam=15, size=n_fraud) + 5


# 合并
credit_df = pd.DataFrame({
    'amount': np.concatenate([normal_amount, fraud_amount]),
    'frequency': np.concatenate([normal_freq, fraud_freq]),
    'is_fraud': [0]*n_normal + [1]*n_fraud
})


print(f'数据集大小：{len(credit_df)}')
print(f'正常交易：{n_normal}, 欺诈交易：{n_fraud}')
print()
print('数据预览：')
print(credit_df.describe().round(2))
```

Step 7-2：多方法异常检测

```
# === Step 7-2: 用三种方法检测 ===


# 方法1: Z-score
```

```
credit_df['z_amount'] = abs(stats.zscore(credit_df['amount']))
credit_df['z_detect'] = credit_df['z_amount'] > 3


# 方法2: IQR
q1 = credit_df['amount'].quantile(0.25)
q3 = credit_df['amount'].quantile(0.75)
iqr = q3 - q1
credit_df['iqr_detect'] = (
    (credit_df['amount'] < q1-1.5*iqr) |
    (credit_df['amount'] > q3+1.5*iqr)
)


# 方法3: 综合（金额+频率 同时异常）
z_freq = abs(stats.zscore(credit_df['frequency']))
credit_df['combined_detect'] = (
    (credit_df['z_amount'] > 2.5) |
    (z_freq > 2.5)
)


# 评估各方法
methods = ['z_detect', 'iqr_detect', 'combined_detect']
names = ['Z-score', 'IQR', '综合方法']


print('=== 检测性能对比 ===')

print(f'{"方法":<12}{"检测数":>8}{"真欺诈":>8}{"准确率":>10}')
print('-'*40)


for name, method in zip(names, methods):
    detected = credit_df[method].sum()
    true_pos = credit_df[credit_df[method] &
               (credit_df['is_fraud']==1)].shape[0]
    precision = true_pos/detected*100 if detected>0 else 0
    print(f'{name:<12}{detected:>8}{true_pos:>8}{precision:>9.1f}%')
```

Step 7-3：可视化检测结果

```
# === Step 7-3: 散点图可视化 ===


fig, ax = plt.subplots(figsize=(12, 8))


# 正常交易
normal = credit_df[credit_df['is_fraud']==0]
ax.scatter(normal['frequency'], normal['amount'],
           c='steelblue', alpha=0.4, s=20, label='正常交易')
```

```
# 欺诈交易
fraud = credit_df[credit_df['is_fraud']==1]
ax.scatter(fraud['frequency'], fraud['amount'],
           c='red', s=100, marker='X', label='欺诈交易',
           edgecolors='darkred', linewidth=1.5)

# 添加 IQR 边界
ax.axhline(y=q3+1.5*iqr, color='orange', linestyle='--',
           alpha=0.8, label='IQR 金额上界')

ax.set_xlabel('交易频率 (次/天)', fontsize=12)
ax.set_ylabel('交易金额 (元)', fontsize=12)
ax.set_title('信用卡交易异常检测', fontsize=14, fontweight='bold')
ax.legend(fontsize=11)

plt.tight_layout()
plt.show()

print('红色 X 标记的就是欺诈交易。')
print('它们的特点：金额异常高 且/或 频率异常高。')
```

>>> 综合案例总结：在实际应用中，通常需要结合多种方法、多个特征来进行异常检测，单一方法往往不够。

实习 8：课后练习

请同学们独立完成以下练习题，加深（jiā shēn）对异常值处理的理解。

练习 1：修改阈值

修改 Z-score 的阈值（从3改为2.5），看看检测结果有什么变化。

```
# 练习1 提示：
threshold = 2.5  # 尝试不同的阈值
z_outlier_new = abs(stats.zscore(df['value'])) > threshold
print(f'阈值={threshold} 时检测到 {z_outlier_new.sum()} 个异常值')
# TODO: 试试 threshold = 2.0, 2.5, 3.0, 3.5
```

练习 2：使用 Winsorize 方法

Winsorize 是把极端值截断（jié duàn）到指定分位数（fēn wèi shù）的方法。

```
# 练习2 提示：
from scipy.stats.mstats import winsorize


# 将最高5%和最低5%的数据截断
data_winsorized = winsorize(df['value'], limits=[0.05, 0.05])


# TODO: 比较 winsorize 前后的统计量和直方图
# TODO: 尝试不同的 limits 参数
```

练习 3：探索 Isolation Forest

用机器学习方法（Isolation Forest）来检测异常值。

```
# 练习3 提示：
from sklearn.ensemble import IsolationForest


# 训练模型
iso_forest = IsolationForest(
    contamination=0.1,  # 预期异常值比例
```

```
    random_state=42
)


# 用两个特征训练（适用于 credit_df）
X = credit_df[['amount', 'frequency']]
credit_df['iso_predict'] = iso_forest.fit_predict(X)


# TODO: 可视化 Isolation Forest 的检测结果
# TODO: 与 Z-score 和 IQR 的结果对比
```

练习 4：自由探索

尝试用今天学到的方法，对以下真实数据集进行异常值分析：

(1) Boston Housing 数据集 - 房价数据

(2) Iris 数据集 - 鸢尾花数据

(3) 或者使用你自己感兴趣的数据

```
# 练习4 提示：
from sklearn.datasets import load_iris


iris = load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)


# TODO: 对每个特征分别进行异常值检测
# TODO: 可视化每个特征的箱线图
# TODO: 尝试多变量异常值检测（如马哈拉诺比斯距离）
```

>>> 提交要求：请将代码和运行结果截图提交到课程平台。截止日期：下周上课前。<<<

实习结束 - 谢谢大家！加油！