数据预处理

缺失值处理实践工作手册

Missing Value Handling Practice Workbook

编程语言	Python
核心库	pandas, numpy, scikit-learn, missingno
适用对象	数据预处理课程学生
难度等级	基础 / 中级 / 高级

姓名		**日期**	

学习目标

完成本工作手册后，你将能够：

1. 识别数据集中的缺失值类型（MCAR、MAR、MNAR）
2. 使用 Python 检测和可视化缺失值模式
3. 根据数据特征选择合适的缺失值处理策略
4. 实现删除法、统计填充法和高级插补法
5. 评估不同缺失值处理方法对数据质量的影响
6. 综合运用多种方法处理复杂的缺失数据场景

第一部分：理论复习

练习 1.1 缺失值基本概念

Q1. 请简要解释以下三种缺失数据机制的定义和区别： **[基础]**

(a) 完全随机缺失 (MCAR - Missing Completely At Random)

(b) 随机缺失 (MAR - Missing At Random)

(c) 非随机缺失 (MNAR - Missing Not At Random)

Q2. 对于下列场景，判断最可能的缺失数据机制类型，并说明理由： **[中级]**

(a) 问卷调查中，部分受访者因跳题导致某些问题没有回答。

(b) 收入调查中，高收入人群更倾向于不填写收入信息。

(c) 传感器数据中，因设备随机故障导致的数据丢失。

Q3. 简述缺失值处理的主要策略类别及其适用场景： **[基础]**

第二部分：缺失值检测与可视化

练习 2.1 环境准备

请先运行以下代码，导入必要的库并创建练习数据集：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


# 创建模拟数据集
np.random.seed(42)
n = 200
data = {
    '学生ID': range(1, n+1),
    '年龄': np.random.randint(18, 30, n).astype(float),
    '性别': np.random.choice(['男', '女'], n),
    '数学成绩': np.random.normal(75, 15, n),
    '英语成绩': np.random.normal(70, 12, n),
    '出勤率': np.random.uniform(0.6, 1.0, n),
    '家庭收入': np.random.lognormal(10, 0.8, n),
}
df = pd.DataFrame(data)


# 人为引入缺失值（模拟不同缺失机制）
# MCAR: 年龄随机缺失
mcar_idx = np.random.choice(n, 20, replace=False)
df.loc[mcar_idx, '年龄'] = np.nan


# MAR: 数学成绩低的学生英语成绩更可能缺失
mar_prob = (df['数学成绩'] < 60).astype(float) * 0.5 + 0.05
mar_mask = np.random.binomial(1, mar_prob).astype(bool)
df.loc[mar_mask, '英语成绩'] = np.nan


# MNAR: 高收入家庭更可能隐藏收入
```

```
mnar_mask = (df['家庭收入'] > df['家庭收入'].quantile(0.7))
mnar_mask = mnar_mask & (np.random.random(n) < 0.6)
df.loc[mnar_mask, '家庭收入'] = np.nan


# 出勤率随机缺失一些
df.loc[np.random.choice(n, 15, replace=False), '出勤率'] = np.nan
```

练习 2.2 缺失值检测

Q4. 编写代码，输出数据集中每列的缺失值数量和缺失比例： **[基础]**

> **提示**
>
> 使用 isnull().sum() 和 shape[0] 计算缺失数量和比例

请在下方编写代码：

```

```

运行结果记录：

Q5. 使用 missingno 库（或 seaborn/matplotlib）绘制缺失值矩阵图和柱状图： **[基础]**

```
# 安装 missingno: pip install missingno
import missingno as msno


# 请在下方补充代码：
# (a) 绘制缺失值矩阵图




# (b) 绘制缺失值柱状图




# (c) 绘制缺失值热力图（相关性）
```

Q6. 根据可视化结果，分析各变量缺失值之间是否存在关联性？请说明你的发现： **[中级]**

第三部分：删除法处理缺失值

练习 3.1 行删除法 (Listwise Deletion)

Q7. 使用 dropna() 方法执行以下操作并比较结果： **[基础]**

(a) 删除含有任何缺失值的行：

(b) 仅删除所有值均为缺失的行：

(c) 基于特定列（数学成绩、英语成绩）删除缺失行：

Q8. 记录并分析上述三种删除策略各自保留了多少行数据？讨论哪种策略最合适及原因： **[中级]**

练习 3.2 列删除法

Q9. 如果某列缺失比例超过 50%，通常建议删除该列。编写代码自动识别并删除高缺失率的列： **[中级]**

> **提示**
>
> 使用 isnull().mean() 获取每列缺失比例，设定阈值进行筛选

Q10. 讨论删除法的优缺点。在什么情况下删除法是合理的？什么情况下应该避免使用？ **[中级]**

第四部分：统计填充法

练习 4.1 基本统计量填充

Q11. 对数值型变量使用均值、中位数和众数进行填充，并比较结果： **[基础]**

```
# 以 '年龄' 列为例
df_mean = df.copy()
df_median = df.copy()
df_mode = df.copy()


# 请在下方补充填充代码：
# (a) 均值填充




# (b) 中位数填充




# (c) 众数填充




# 比较填充后的统计特征
print('原始数据描述统计：')
print(df['年龄'].describe())
```

Q12. 比较均值填充和中位数填充后数据的标准差变化，哪种方法对数据分布的影响更小？为什么？

[中级]

练习 4.2 分组填充

Q13. 按性别分组，使用各组的中位数填充 '数学成绩' 列的缺失值： **[中级]**

> **提示**
>
> 使用 groupby + transform 方法可以便捷地实现分组填充

Q14. 为什么分组填充通常比全局填充更合理？请用本数据集的例子说明： **[中级]**

练习 4.3 前向/后向填充

Q15. 假设数据按学生ID排序，使用前向填充和后向填充处理 '出勤率' 列： **[基础]**

```
# 前向填充 (forward fill)
df_ffill = df.copy()




# 后向填充 (backward fill)
df_bfill = df.copy()




# 思考：这种方法适合当前数据吗？为什么？
```

Q16. 前向/后向填充最适合什么类型的数据？在本数据集中使用是否合适？请解释原因： **[中级]**

第五部分：高级插补方法

练习 5.1 KNN 插补

Q17. 使用 scikit-learn 的 KNNImputer 对数值型变量进行插补： **[高级]**

```
from sklearn.impute import KNNImputer


# 选择数值型列
num_cols = ['年龄', '数学成绩', '英语成绩', '出勤率', '家庭收入']
df_knn = df[num_cols].copy()


# 请补充以下代码：
# (a) 创建 KNNImputer (n_neighbors=5)




# (b) 执行插补




# (c) 输出插补后的数据描述统计
```

Q18. 尝试不同的 k 值 (k=3, 5, 10, 20)，比较插补结果。k 值对结果有什么影响？ **[高级]**

```

```

分析记录：

练习 5.2 多重插补 (MICE)

Q19. 使用 sklearn 的 IterativeImputer 实现多重插补： **[高级]**

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer


# 请补充以下代码：
# (a) 创建 IterativeImputer




# (b) 对数值型数据执行插补
```

```
# (c) 比较 MICE 与 KNN 插补的结果差异
```

Q20. 比较 KNN 插补和 MICE 方法的优缺点，讨论各自适用的场景： **[高级]**

第六部分：综合实战练习

练习 6.1 完整的缺失值处理流程

Q21. 请按照以下步骤完成一个完整的缺失值处理流程： **[高级]**

步骤 1：数据探索与缺失值诊断

编写代码完成数据全貌探索，包括：数据形状、数据类型、缺失值统计、基本描述统计。

步骤 2：缺失值可视化分析

创建至少两种不同的可视化图表来展示缺失值模式。

步骤 3：制定处理策略

根据分析结果，为每个含缺失值的变量制定处理策略，填写下表：

变量名称	缺失机制	处理策略	选择理由
年龄			
英语成绩			
出勤率			
家庭收入			

步骤 4：实施处理方案

根据上表的策略，编写完整的处理代码：

步骤 5：结果验证与评估

编写代码验证处理后的数据质量，包括：确认无缺失值、比较处理前后的分布变化、数据完整性检查。

第七部分：总结与反思

练习 7.1 方法对比总结

Q22. 填写以下缺失值处理方法对比表： **[中级]**

方法	优点	缺点	适用场景
直接删除			
均值/中位数填充			
分组填充			
KNN 插补			
MICE 多重插补			

练习 7.2 思考题

Q23. 在实际项目中，你如何决定使用哪种缺失值处理方法？请列出你的决策流程： **[高级]**

Q24. 缺失值处理会对后续的机器学习模型产生什么影响？举例说明不当处理可能导致的问题： **[高级]**

Q25. 学习本章后，你认为最重要的三个知识点是什么？在实践中遇到了哪些困难？ **[基础]**

附录：常用代码速查表

A. 缺失值检测

```
df.isnull().sum()                # 每列缺失数量
df.isnull().mean()               # 每列缺失比例
df.isnull().sum().sum()          # 总缺失数量
df.info()                        # 数据概览
df[df.isnull().any(axis=1)]      # 含缺失值的行
```

B. 缺失值可视化

```
import missingno as msno
msno.matrix(df)                  # 缺失值矩阵图
msno.bar(df)                     # 缺失值柱状图
msno.heatmap(df)                 # 缺失值相关性热力图
msno.dendrogram(df)              # 缺失值聚类树状图


# seaborn 热力图
sns.heatmap(df.isnull(), cbar=True, yticklabels=False)
```

C. 删除法

```
df.dropna()                      # 删除含缺失值的行
df.dropna(how='all')             # 删除全为缺失值的行
df.dropna(subset=['col1'])       # 基于特定列删除
df.dropna(thresh=3)              # 保留至少3个非空值的行
df.drop(columns=['col1'])        # 删除列
```

D. 填充法

```
df['col'].fillna(df['col'].mean())      # 均值填充
df['col'].fillna(df['col'].median())    # 中位数填充
df['col'].fillna(df['col'].mode()[0])   # 众数填充
df['col'].ffill()                       # 前向填充
df['col'].bfill()                       # 后向填充
df['col'].interpolate()                 # 线性插值


# 分组填充
df['col'] = df.groupby('group')['col'].transform(
    lambda x: x.fillna(x.median())
```

```
)
```

E. 高级插补

```
# KNN 插补
from sklearn.impute import KNNImputer
imputer = KNNImputer(n_neighbors=5)
df_imputed = pd.DataFrame(
    imputer.fit_transform(df[num_cols]),
    columns=num_cols
)


# MICE (迭代插补)
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
imputer = IterativeImputer(max_iter=10, random_state=42)
df_imputed = pd.DataFrame(
    imputer.fit_transform(df[num_cols]),
    columns=num_cols
)
```