数据预处理

缺失值处理实践工作手册

—— 参考答案与详解 ——

Answer Key & Detailed Explanations

⚠ 教师专用 - 请勿分发给学生

本文档包含所有练习题的参考答案、代码示例和评分要点

第一部分：理论复习 - 参考答案

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)

✓ 参考答案

定义：数据缺失的概率与任何观测到的或未观测到的变量都无关。缺失是完全随机发生的。

特点：缺失数据是原始数据的一个随机子集，删除缺失值后不会引入偏差。

例子：实验室设备随机故障导致的数据丢失。

(b) 随机缺失 (MAR)

✓ 参考答案

定义：数据缺失的概率与缺失值本身无关，但与其他已观测到的变量有关。

特点：可以通过其他观测变量来预测缺失模式，可以使用统计方法进行有效插补。

例子：年轻人更倾向于不填写邮箱地址，缺失与年龄有关但与邮箱本身无关。

(c) 非随机缺失 (MNAR)

✓ 参考答案

定义：数据缺失的概率与缺失值本身直接相关。

特点：最难处理的缺失类型，无法仅通过观测数据来纠正偏差，需要领域知识或特殊模型。

例子：抱郁症患者更可能不填写心理健康调查问卷，缺失与答案本身有关。

Q2. 对于下列场景，判断最可能的缺失数据机制类型： **[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。

✓ 参考答案

属于 MAR（随机缺失）。跳题通常是基于前面问题的回答而触发的，因此缺失与已观测到的其他变量有关，

而非与缺失答案本身直接相关。例如：回答“未婚”的人跳过婚姻相关问题。

(b) 收入调查中，高收入人群更倾向于不填写收入信息。

✓ 参考答案

属于 MNAR（非随机缺失）。缺失的原因直接与缺失值本身相关——收入越高，越不愿意披露，

因此缺失概率与收入值（即缺失值本身）直接相关。

(c) 传感器数据中，因设备随机故障导致的数据丢失。

✓ 参考答案

属于 MCAR（完全随机缺失）。设备故障是随机发生的，与被测量的值无关，

也与其他变量无关。数据丢失是纯粹的随机事件。

Q3. 简述缺失值处理的主要策略类别及其适用场景： **[基础]**

✓ 参考答案

1. 删除法：包括行删除和列删除。适用于 MCAR 且缺失比例小（通常 <5%）的情况，或样本量足够大的情况。

2. 统计填充法：均值、中位数、众数填充。适用于缺失比例中等、数据分布较均匀的情况。

3. 插值法：线性插值、前向/后向填充。适用于时间序列数据。

4. 模型插补法：KNN、MICE、回归插补。适用于 MAR 且需要保留变量间关系的场景。

5. 标记法：添加缺失指示变量。适用于 MNAR 或缺失本身含有信息的场景。

第二部分：缺失值检测与可视化 - 参考答案

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]**

```
# 方法一：基本方法
print('缺失值数量：')
print(df.isnull().sum())
print('\n缺失比例：')
print(df.isnull().mean().round(4) * 100, '%')


# 方法二：更美观的汇总方法
missing_info = pd.DataFrame({
    '缺失数量': df.isnull().sum(),
    '缺失比例(%)': (df.isnull().mean() * 100).round(2)
})
missing_info = missing_info[missing_info['缺失数量'] > 0]
missing_info = missing_info.sort_values('缺失比例(%)', ascending=False)
print(missing_info)
```

✓ 参考答案

预期输出示例（具体数值因随机种子而异）：

年龄：缺失 20 个 (10.0%)，英语成绩：缺失约 15-25 个，出勤率：缺失 15 个 (7.5%)，家庭收入：缺失约 30-40 个

评分要点：能同时显示缺失数量和比例，代码可运行，输出格式清晰。

Q5. 使用 missingno 库绘制缺失值矩阵图和柱状图： **[基础]**

```
import missingno as msno


# (a) 缺失值矩阵图
msno.matrix(df, figsize=(10, 6), fontsize=12)
plt.title('缺失值矩阵图', fontsize=14)
plt.tight_layout()
plt.show()


# (b) 缺失值柱状图
msno.bar(df, figsize=(10, 6), fontsize=12, color='steelblue')
plt.title('缺失值柱状图', fontsize=14)
```

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


# (c) 缺失值热力图（相关性）
msno.heatmap(df, figsize=(8, 6), fontsize=12)
plt.title('缺失值相关性热力图', fontsize=14)
plt.tight_layout()
plt.show()
```

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： **[中级]**

✓ 参考答案

参考分析要点：

1. 矩阵图观察：英语成绩的缺失模式与数学成绩较低的行存在关联（MAR 特征）。

2. 热力图观察：数学成绩与英语成绩的缺失可能存在正相关（因为我们设置了数学成绩低时英语更可能缺失）。

3. 年龄和出勤率的缺失与其他变量基本无关联（MCAR 特征）。

4. 家庭收入的缺失缺失比例较高，且与其本身的值相关（MNAR 特征）。

第三部分：删除法处理缺失值 - 参考答案

练习 3.1 行删除法

Q7. 使用 dropna() 方法执行以下操作： **[基础]**

(a) 删除含有任何缺失值的行：

```
df_any = df.dropna(how='any')
print(f'原始数据行数: {len(df)}')
print(f'删除后行数: {len(df_any)}')
print(f'删除了 {len(df) - len(df_any)} 行 ({(len(df)-len(df_any))/len(df)*100:.1f}%)')
```

(b) 仅删除所有值均为缺失的行：

```
df_all = df.dropna(how='all')
print(f'删除后行数: {len(df_all)}')
# 通常结果与原始相同，因为很少有整行全缺失
```

(c) 基于特定列删除：

```
df_subset = df.dropna(subset=['数学成绩', '英语成绩'])
print(f'删除后行数: {len(df_subset)}')
```

Q8. 记录并分析上述三种删除策略各自保留了多少行数据： **[中级]**

✓ 参考答案

参考分析：

(a) how='any' 会删除大量数据，因为多个列都有缺失值，可能仅保留 50-60% 的数据。过于激进。

(b) how='all' 几乎不会删除任何行，因为很难有整行全为 NaN 的情况。过于宽松。

(c) 基于关键列的删除是最合理的策略，因为它只关注分析所需的变量，平衡了数据完整性和样本量。

结论：应根据分析目的选择删除策略，而非一律使用 how='any'。

练习 3.2 列删除法

Q9. 自动识别并删除高缺失率的列： **[中级]**

```
# 设定缺失比例阈值
threshold = 0.5
```

```
# 计算每列缺失比例
missing_ratio = df.isnull().mean()

# 筛选高缺失率的列
high_missing_cols = missing_ratio[missing_ratio > threshold].index.tolist()
print(f'缺失比例超过 {threshold*100}% 的列: {high_missing_cols}')

# 删除高缺失率列
df_dropped = df.drop(columns=high_missing_cols)
print(f'删除前列数: {df.shape[1]}, 删除后列数: {df_dropped.shape[1]}')
```

Q10. 讨论删除法的优缺点： **[中级]**

✓ 参考答案

优点：(1) 简单易实现，不引入人为偏差；(2) 保持变量间的真实关系；(3) 在 MCAR 且缺失比例小时效果好。

缺点：(1) 可能大量减少样本量，降低统计力率；(2) 如果不是 MCAR，可能引入选择性偏差；(3) 浪费已收集的部分信息。

合理情况：MCAR 且缺失 <5%，或样本量非常大。

应避免：缺失比例高、MAR/MNAR 机制、样本量本就不大的情况。

第四部分：统计填充法 - 参考答案

练习 4.1 基本统计量填充

Q11. 使用均值、中位数和众数填充： **[基础]**

```
df_mean = df.copy()
df_median = df.copy()
df_mode = df.copy()


# (a) 均值填充
df_mean['年龄'] = df_mean['年龄'].fillna(df_mean['年龄'].mean())


# (b) 中位数填充
df_median['年龄'] = df_median['年龄'].fillna(df_median['年龄'].median())


# (c) 众数填充
df_mode['年龄'] = df_mode['年龄'].fillna(df_mode['年龄'].mode()[0])


# 比较结果
print('原始数据:')
print(df['年龄'].describe())
print(f'\n均值填充后 均值={df_mean["\u5E74\u9F84"].mean():.2f}, 标准差
={df_mean["\u5E74\u9F84"].std():.2f}')
print(f'中位数填充后 均值={df_median["\u5E74\u9F84"].mean():.2f}, 标准差
={df_median["\u5E74\u9F84"].std():.2f}')
print(f'众数填充后 均值={df_mode["\u5E74\u9F84"].mean():.2f}, 标准差
={df_mode["\u5E74\u9F84"].std():.2f}')
```

Q12. 比较均值填充和中位数填充后数据的标准差变化： **[中级]**

✓ 参考答案

两种方法都会使标准差减小（因为缺失值被替换为集中趋势的值，减少了数据离散度）。

均值填充：不改变均值，但会显著减少标准差，可能低估变异性。

中位数填充：对标准差的影响与均值填充类似，但对偶数分布更稳健，不受极端值影响。

结论：对于偏态分布的数据，中位数填充对分布的影响更小；对于近似正态分布，两者差异不大。

练习 4.2 分组填充

Q13. 按性别分组，用各组中位数填充数学成绩： **[中级]**

```
df_grouped = df.copy()


# 方法一：groupby + transform
df_grouped['数学成绩'] = df_grouped.groupby('性别')['数学成绩'].transform(
    lambda x: x.fillna(x.median())
)


# 验证
print(df_grouped['数学成绩'].isnull().sum())  # 应为 0


# 查看分组统计
print('填充前分组统计:')
print(df.groupby('性别')['数学成绩'].describe())
print('\n填充后分组统计:')
print(df_grouped.groupby('性别')['数学成绩'].describe())
```

Q14. 为什么分组填充通常比全局填充更合理： **[中级]**

✓ 参考答案

分组填充能够保留组间差异。例如如果男生和女生的数学成绩分布不同，全局均值填充会模糊这种差异。

分组填充用各组自己的中位数，更精确地反映该组的典型值，减少填充引入的偏差。

本数据集中，男女生的成绩分布可能存在差异，按性别分组填充可以更好地保留这种差异。

练习 4.3 前向/后向填充

Q15. 前向填充和后向填充处理出勤率： **[基础]**

```
# 前向填充 (forward fill)
df_ffill = df.copy()
df_ffill['出勤率'] = df_ffill['出勤率'].ffill()


# 后向填充 (backward fill)
df_bfill = df.copy()
df_bfill['出勤率'] = df_bfill['出勤率'].bfill()
```

```
# 检查结果
print(f'前向填充后缺失数: {df_ffill["\u51FA\u52E4\u7387"].isnull().sum()}')
print(f'后向填充后缺失数: {df_bfill["\u51FA\u52E4\u7387"].isnull().sum()}')
```

Q16. 前向/后向填充最适合什么类型的数据？在本数据集中是否合适： **[中级]**

✓ 参考答案

前向/后向填充最适合时间序列数据，其中相邻时间点的值具有连续性和相似性。

典型应用：股票价格、温度记录、传感器数据等时间序列数据。

本数据集中不太合适，因为学生数据按 ID 排序，相邻学生的出勤率之间没有逻辑上的连续性关系，

前一个学生的出勤率不能代表下一个学生的出勤率。应优先考虑均值/中位数填充或分组填充。

第五部分：高级插补方法 - 参考答案

练习 5.1 KNN 插补

Q17. 使用 KNNImputer 对数值型变量插补： **[高级]**

```
from sklearn.impute import KNNImputer


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()


# (a) 创建 KNNImputer
imputer = KNNImputer(n_neighbors=5)


# (b) 执行插补
df_knn_imputed = pd.DataFrame(
    imputer.fit_transform(df_knn),
    columns=num_cols
)


# (c) 插补后描述统计
print('插补前:')
print(df_knn.describe())
print('\n插补后:')
print(df_knn_imputed.describe())
print('\n缺失值检查:', df_knn_imputed.isnull().sum().sum())
```

Q18. 尝试不同的 k 值，比较插补结果： **[高级]**

```
results = {}
for k in [3, 5, 10, 20]:
    imp = KNNImputer(n_neighbors=k)
    imputed = pd.DataFrame(imp.fit_transform(df_knn), columns=num_cols)
    results[k] = {
        'mean': imputed.mean(),
        'std': imputed.std()
    }
    print(f'\nk={k}:')
    print(f'  年龄 均值={imputed["\u5E74\u9F84"].mean():.2f}, 标准差
={imputed["\u5E74\u9F84"].std():.2f}')
    print(f'  家庭收入 均值={imputed["\u5BB6\u5EAD\u6536\u5165"].mean():.2f}, 标准差
```

```
={imputed["\u5BB6\u5EAD\u6536\u5165"].std():.2f}')
```

✓ 参考答案

k 值对结果的影响：

k 较小（如 k=3）：插补值受局部邻居影响大，结果波动较大，可能过拟合局部模式。

k 较大（如 k=20）：插补值趋向于全局均值，结果更平滑，但可能丢失局部特征。

建议：通常 k=5 或 k=√n 是一个合理的起点，可通过交叉验证确定最优 k 值。

练习 5.2 多重插补 (MICE)

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

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


# (a) 创建 IterativeImputer
mice_imputer = IterativeImputer(
    max_iter=10,
    random_state=42,
    sample_posterior=False
)


# (b) 执行插补
df_mice = pd.DataFrame(
    mice_imputer.fit_transform(df_knn),
    columns=num_cols
)


# (c) 比较 MICE 与 KNN
print('原始数据描述统计:')
print(df_knn.describe().round(2))
print('\nKNN 插补后:')
print(df_knn_imputed.describe().round(2))
print('\nMICE 插补后:')
print(df_mice.describe().round(2))


# 相关性比较
print('\n原始相关矩阵:')
```

```
print(df_knn.corr().round(3))
print('\nMICE 插补后相关矩阵:')
print(df_mice.corr().round(3))
```

Q20. 比较 KNN 和 MICE 的优缺点： **[高级]**

✓ 参考答案

KNN 插补：

优点 - 直观易理解，不需要假设数据分布，能捕捉非线性关系。

缺点 - 计算复杂度高（$O(n^2)$），对异常值和维度敏感，需要特征缩放。

适用 - 数据量中等，变量间存在非线性关系的场景。

MICE (IterativeImputer)：

优点 - 能保留变量间的统计关系，可处理不同类型变量，理论基础扎实。

缺点 - 计算开销大，需要多次迭代，假设变量间存在线性关系（默认用回归模型）。

适用 - MAR 数据，变量间有明确线性关系，需要高质量插补结果的场景。

第六部分：综合实战练习 - 参考答案

练习 6.1 完整的缺失值处理流程

Q21. 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

```
# 数据全貌探索
print('数据形状:', df.shape)
print('\n数据类型:')
print(df.dtypes)
print('\n缺失值统计:')
missing_stats = pd.DataFrame({
    '缺失数量': df.isnull().sum(),
    '缺失比例(%)': (df.isnull().mean() * 100).round(2),
    '数据类型': df.dtypes
})
print(missing_stats[missing_stats['缺失数量'] > 0])
print('\n描述统计:')
print(df.describe())
```

步骤 2：缺失值可视化分析

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


# 图1：缺失值比例柱状图
missing_pct = df.isnull().mean() * 100
missing_pct = missing_pct[missing_pct > 0].sort_values(ascending=False)
axes[0].barh(missing_pct.index, missing_pct.values, color='steelblue')
axes[0].set_xlabel('缺失比例 (%)')
axes[0].set_title('各变量缺失比例')


# 图2：缺失值热力图
sns.heatmap(df.isnull(), cbar=True, yticklabels=False, ax=axes[1])
axes[1].set_title('缺失值模式热力图')


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

步骤 3：制定处理策略

变量名称	缺失机制	处理策略	选择理由

年龄	MCAR	中位数填充或 KNN 插补	MCAR 且缺失比例中等，中位数对偏态数据更稳健
英语成绩	MAR	分组填充或 MICE 插补	MAR 与数学成绩相关，需保留变量间关系
出勤率	MCAR	均值/中位数填充	MCAR 且缺失比例较小，简单方法即可
家庭收入	MNAR	标记法 + KNN 插补	MNAR 需谨慎，可添加缺失指示变量再插补

步骤 4：实施处理方案

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


df_clean = df.copy()


# 1. 年龄：中位数填充 (MCAR)
df_clean['年龄'] = df_clean['年龄'].fillna(df_clean['年龄'].median())


# 2. 英语成绩：按性别分组填充 (MAR)
df_clean['英语成绩'] = df_clean.groupby('性别')['英语成绩'].transform(
    lambda x: x.fillna(x.median())
)


# 3. 出勤率：均值填充 (MCAR, 缺失少)
df_clean['出勤率'] = df_clean['出勤率'].fillna(df_clean['出勤率'].mean())


# 4. 家庭收入：添加缺失指示 + KNN插补 (MNAR)
df_clean['收入缺失标记'] = df['家庭收入'].isnull().astype(int)
num_cols_income = ['年龄', '数学成绩', '英语成绩', '出勤率', '家庭收入']
imputer = KNNImputer(n_neighbors=5)
df_clean[num_cols_income] = pd.DataFrame(
    imputer.fit_transform(df_clean[num_cols_income]),
    columns=num_cols_income,
    index=df_clean.index
)
```

步骤 5：结果验证与评估

```
# 确认无缺失值
```

```
print('处理后缺失值数量:')
print(df_clean.isnull().sum())


# 比较处理前后的描述统计
print('\n处理前:')
print(df.describe().round(2))
print('\n处理后:')
print(df_clean.describe().round(2))


# 分布变化可视化
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
cols_to_check = ['年龄', '英语成绩', '出勤率', '家庭收入']
for i, col in enumerate(cols_to_check):
    ax = axes[i//2, i%2]
    df[col].hist(alpha=0.5, label='原始', ax=ax, bins=20)
    df_clean[col].hist(alpha=0.5, label='处理后', ax=ax, bins=20)
    ax.set_title(f'{col} 分布对比')
    ax.legend()
plt.tight_layout()
plt.show()
```

第七部分：总结与反思 - 参考答案

练习 7.1 方法对比总结

Q22. 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	简单快速，不引入人为偏差，保持真实关系	减少样本量，非 MCAR 时可能引入偏差	MCAR、缺失<5%、样本量大
均值/中位数填充	简单易实现，保持样本量，不改变均值	减少变异性，影响分布形状，忽略变量间关系	缺失比例小、数据分布均匀
分组填充	保留组间差异，比全局填充更精确	需要合适的分组变量，组内仍使用单一统计量	MAR、存在明确分组变量
KNN 插补	考虑多变量关系，不假设分布，能捕捉非线性	计算量大，对异常值敏感，需要特征缩放	变量间存在非线性关系、中等数据量
MICE 多重插补	保留变量间关系，理论严谨，可处理多种类型	计算开销大，需多次迭代，默认假设线性关系	MAR、多变量缺失、需高质量插补

练习 7.2 思考题

Q23. 在实际项目中的决策流程： **[高级]**

✓ 参考答案

参考决策流程：

1. 识别缺失模式：统计缺失比例、可视化缺失模式、分析缺失之间的相关性。
2. 判断缺失机制：通过统计检验（如 Little's MCAR test）或领域知识判断是 MCAR/MAR/MNAR。
3. 评估缺失比例：<5% 可用简单方法，5-20% 需谨慎选择，>20% 需考虑删除或高级插补。
4. 选择方法：根据缺失机制和数据特征选择合适的方法。
5. 验证效果：比较处理前后的分布变化、相关性变化、对下游任务的影响。
6. 记录决策：记录选择的理由和处理过程，以便复现和审查。

Q24. 缺失值处理对机器学习模型的影响： **[高级]**

✓ 参考答案

主要影响：

1. 偏差引入：不当的填充方法会改变数据分布，导致模型学习到错误的模式。

   例如：用均值填充家庭收入（MNAR）会低估高收入群体的平均收入。

2. 方差低估：均值填充会减少变量方差，导致置信区间过窄，显著性检验过于乐观。

3. 相关性破坏：简单填充可能破坏变量间的真实关系，影响特征选择和模型性能。

4. 样本量减少：过度删除导致训练数据不足，模型泛化能力下降。

建议：始终在训练集上拟合插补器，用相同参数变换测试集，避免数据泄漏。

Q25. 学习本章后的三个最重要知识点： **[基础]**

✓ 参考答案

参考答案（开放性问题，以下为示例）：

1. 理解缺失机制是选择处理方法的前提：不同缺失机制适用不同方法，盲目处理可能引入偏差。

2. 没有万能的缺失值处理方法：每种方法都有其优缺点和适用场景，必须根据具体问题选择。

3. 验证和评估不可少：处理后必须检查数据质量，确保处理效果符合预期。

评分要点：学生能够说出自己的理解和反思即可，不需要与参考答案完全一致。

附录：评分指南

评分原则

- 代码题：代码可运行 (40%) + 逻辑正确 (30%) + 代码规范 (15%) + 注释说明 (15%)
- 分析题：观点正确 (40%) + 逻辑清晰 (30%) + 有具体例子 (20%) + 表达完整 (10%)
- 开放性问题：论述合理即可，不要求与参考答案完全一致

题目难度分布与分值建议

难度等级	题目数量	建议分值	占比	对应题号
基础	8 题	每题 3 分	24%	Q1,Q3,Q4,Q5,Q7,Q11,Q15,Q25
中级	10 题	每题 4 分	40%	Q2,Q6,Q8,Q9,Q10,Q12,Q13,Q14,Q16,Q22
高级	7 题	每题 5 分	35%	Q17,Q18,Q19,Q20,Q21,Q23,Q24

✓ 参考答案

总分建议：99 分（基础 24 + 中级 40 + 高级 35），可按实际需要调整。

让学生在缺失值处理的思维过程和实践能力上得到充分锻炼。