

实验报告

学号: 202300130050

姓名: 王睿

班级: 数据 23

实验题目: 数据质量实践

实验学时: 2

实验日期: 2025.10.26

实验目标

本次实验主要围绕宝可梦数据集进行分析，考察在拿到数据后如何对现有的数据进行预处理清洗操作，建立起对于脏数据、缺失数据等异常情况的一套完整流程的认识。通过实践数据质量评估的四个维度（完整性、一致性、准确性、时效性/有效性），掌握数据清洗的基本方法和技巧，为后续的数据分析工作打下基础。

实验环境

- 操作系统: Windows
- 开发工具: VS Code / Jupyter N
- 编程语言: Python 3.8.20

实验内容

2.4.1 读取数据集

首先读取宝可梦数据集 (Pokemon.csv)，处理文件路径和编码问题：

- 自动查找当前目录下的 CSV 文件（支持不同大小写）
- 尝试多种编码方式 (GBK、UTF-8、Latin-1) 以确保正确读取
- 显示数据集的基本信息和前 5 行数据

数据集基本信息：

- 总行数：810 条记录
- 总列数：13 列（包括编号、名称、属性、各项能力值等）
- 数据类型：初始读取时大部分列为 object 类型，需要后续转换

2.4.2 完整性：缺失值处理

检查并处理数据集中的缺失值：

1. 检查缺失值：统计每列的缺失值数量
2. 缺失值填充：对 Type 2 列使用'None'字符串填充
3. 处理结果：填充后 Type 2 列的缺失值数量降为 0

2.4.3 一致性：重复值处理

检查并删除重复数据：

1. 检查完全重复行：发现 7 条完全重复的记录
2. 删除重复行：删除后数据集大小从 810 条减少到 803 条
3. 检查 Name 列重复：发现 1 个名称重复（可能是 Mega 进化等特殊情况，属于正常现象）

2.4.4 准确性：异常值处理

使用 IQR（四分位距）方法处理异常值，以 Attack（攻击力）列为例：

1. 数据类型转换：先将数值列（Total、HP、Attack、Defense 等）从 object 类型转换为数值类型
2. 计算 IQR：
3. 定义异常值边界：
4. 处理异常值：将超出边界的异常值替换为边界值，创建新列 Attack_cleaned 保存处理后的数据
5. 对比分析：比较处理前后的统计信息，验证异常值处理效果

2.4.5 时效性/有效性：数据格式和逻辑检查

检查数据的有效性和格式：

1. Legendary 列转换：将传说宝可梦列转换为布 bool 类型，确保数据类型正确
2. Generation 列范围检查：
3. 最终数据质量检查：输出处理后的数据集结构信息

实验结果

数据质量处理结果

1. 缺失值处理：

2. 重复值处理：

3. 异常值处理：

4. 数据类型转换：

```
PS D:\大数据分析> & D:/python/_python.exe d:/大数据分析/2/clean.py
原始宝可梦数据集（前5行）：
   #           Name Type 1  Type 2 Total  HP Attack Defense Sp. Atk Sp. Def Speed Generation Legendary
0  1       Bulbasaur  Grass  Poison  318  45    49    49    65    65    45      1    FALSE
1  2        Ivysaur  Grass  Poison  405  60    62    63    80    80    60      1    FALSE
2  3       Venusaur  Grass  Poison  525  80    82    83   100   100   80      1    FALSE
3  3  VenusaurMega Venusaur  Grass  Poison  625  80   100   123   122   120   80      1    FALSE
4  4     Charmander   Fire    NaN  309  39    52    43    60    50    65      1    FALSE

数据集信息：
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 810 entries, 0 to 809
Data columns (total 13 columns):
 #  Column      Non-Null Count Dtype  
--- 
 0  #           807 non-null   object  
 1  Name         807 non-null   object  
 2  Type 1      806 non-null   object  
 3  Type 2      424 non-null   object  
 4  Total         807 non-null   object  
 5  HP            806 non-null   object  
 6  Attack        807 non-null   object  
 7  Defense       807 non-null   object  
 8  Sp. Atk       807 non-null   object  
 9  Sp. Def       807 non-null   object  
 10 Speed          807 non-null   object  
 11 Generation    807 non-null   object  
 12 Legendary     807 non-null   object  
dtypes: object(13)
memory usage: 82.4+ KB
```

```
每列的缺失值数量:
```

```
#          3
Name       3
Type 1     4
Type 2    386
Total      3
HP         4
Attack     3
Defense    3
Sp. Atk    3
Sp. Def    3
Speed      3
Generation 3
Legendary   3
dtype: int64
```

```
填充 Type 2 缺失值后的缺失值数量:
```

```
#          3
Name       3
Type 1     4
Type 2     0
Total      3
HP         4
Attack     3
Defense    3
Sp. Atk    3
Sp. Def    3
Speed      3
Generation 3
Legendary   3
dtype: int64
```

```
完全重复行数量: 7
```

```
删除重复行后的数据集大小: 803
```

```
基于 Name 的重复数量 (用于检查数据一致性): 1
```

```
count    800.000000
mean     81.095000
std      53.245327
min      5.000000
25%     55.000000
50%     75.000000
75%     100.000000
max     1000.000000
Name: Attack, dtype: float64

'Attack_cleaned' 处理后统计信息:
count    800.000000
mean     79.110625
std      32.445670
min      5.000000
25%     55.000000
50%     75.000000
75%     100.000000
max     167.500000
Name: Attack_cleaned, dtype: float64

'Legendary' 数据类型转换完成: bool
Generation 范围: 1.0 到 6.0
```

数据质量处理后的数据集结构:

```
<class 'pandas.core.frame.DataFrame'>
Index: 803 entries, 0 to 808
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   #               801 non-null    object 
 1   Name             801 non-null    object 
 2   Type_1           800 non-null    object 
 3   Type_2           803 non-null    object 
 4   Total            800 non-null    float64
 5   HP               799 non-null    float64
 6   Attack           800 non-null    float64
 7   Defense          800 non-null    float64
```

并保存为 clean_data.csv,

数据质量改进

- **完整性**：通过填充缺失值，提高了数据的完整性
- **一致性**：通过删除重复记录，保证了数据的一致性
- **准确性**：通过异常值处理，提高了数据的准确性
- **有效性**：通过数据类型转换和范围检查，确保了数据的有效性

实验总结

通过本次实验，掌握了数据质量评估和清洗的完整流程：

1. **数据读取**：学会了处理不同编码格式的文件，以及如何自动查找和读取数据文件
2. **缺失值处理**：理解了缺失值的不同类型和处理方法，学会了根据业务逻辑选择合适的填充策略
3. **重复值处理**：掌握了识别和删除重复数据的方法，理解了完全重复和部分重复的区别
4. **异常值处理**：学会了使用 IQR 方法识别和处理异常值，理解了异常值处理对数据分析的重要性
5. **数据格式检查**：掌握了数据类型转换和逻辑范围检查的方法，确保数据的有效性

本次实验成功建立了一套完整的数据质量处理流程，为后续的数据分析工作打下了坚实的基础。通过实践，深入理解了数据清洗的重要性，以及如何根据不同的数据质量问题选择合适的处理方法。