山东大学 计算机科学与技术 学院

大数据安全 课程实验报告

学号：202300130090	姓名：杨笑语	班级：数据班
实验题目：数据质量实践		
实验学时：2	实验日期： 2025/9/26	
实验要求：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。		
硬件环境： 计算机一台		
软件环境： python3.9, jupyter notebook 编码格式：GBK（适配数据集中文编码）		

实验步骤与内容：

（一）实验准备：环境配置与数据加载

库安装与升级由于读取 Excel 文件需依赖 xlrd 库，且旧版本可能存在兼容性问题，首先执行升级命令：

```
[1]: !pip install xlrd --upgrade
```

```
Requirement already satisfied: xlrd in d:\anaconda\lib\site-packages (2.0.1)
Collecting xlrd
  Using cached xlrd-2.0.2-py2.py3-none-any.whl.metadata (3.5 kB)
Using cached xlrd-2.0.2-py2.py3-none-any.whl (96 kB)
Installing collected packages: xlrd
  Attempting uninstall: xlrd
    Found existing installation: xlrd 2.0.1
    Uninstalling xlrd-2.0.1:
      Successfully uninstalled xlrd-2.0.1
Successfully installed xlrd-2.0.2
```

导入核心库导入数据处理、数值计算及可视化所需库：

```
[2]: import pandas as pd
     from pandas import DataFrame
     import numpy as np
```

数据路径验证与加载指定宝可梦数据集（Pokemon.xls）的本地路径，先验证路径有效性，再读取数据：

```
import os
file_path = 'D:\\Pokemon.xls'
print(os.path.exists(file_path))  # 检查路径是否存在
print(os.path.isfile(file_path))  # 检查是否是文件（不是文件夹）
```

```
True
True
```

[12]:
```
df=pd.read_excel('D:\Pokemon.xls')
df
```

[12]:

	#	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
805	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	1
806	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined
807	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined
808	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
809	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

810 rows × 13 columns

（二）数据清洗：问题识别与处理

1. 处理无意义尾行

数据集末尾存在 4 行无意义空值或无效数据，干扰后续分析。处理代码通过切片删除末尾 4 行数据：

[13]:
```
df=df[:-4]
df
```

[13]:

	#	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
801	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	1
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	1
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	1
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	1
805	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	1

806 rows × 13 columns

2. 清理 Type 2 列的异常数值

Type 2 列（宝可梦第二属性）应是字符串类型（如 “Poison”“Flying”），但存在整数 / 浮点数类型的异常值（如 “273”）。处理代码通过 lambda 函数判断数据类型，将整数 / 浮点数类型的值替换为 NaN（表示缺失值，符合属性逻辑）：

[14]:
```
condition=df['Type 2'].apply(lambda x: isinstance(x, (int, float)))
df.loc[condition,'Type 2']=np.nan
df
```

```
C:\Users\53207\AppData\Local\Temp\ipykernel_2492\2867539432.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.loc[condition,'Type 2']=np.nan
```

[14]:

	#	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
801	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	1
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	1
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	1
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	1
805	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	1

806 rows × 13 columns

3. 去除重复数据

数据集中存在重复记录（如同一宝可梦重复录入），可能导致分析结果偏差。以 Name 列（宝可梦名称）为唯一标识，删除重复行，并验证去重效果：

[15]:
```
df_clean=df.drop_duplicates(subset=['Name'])
df_clean
```

[15]:

	#	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...
801	719	Diancie	Rock	Fairy	600	50	100	150	100	150	50	6	1
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110	110	6	1
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130	70	6	1
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130	80	6	1
805	721	Volcanion	Fire	Water	600	80	110	120	130	90	70	6	1

801 rows × 13 columns

[16]:
```
has_dup=df_clean.duplicated().any()
if has_dup:
    print("仍有重复值")
else:
    print("无重复值")
```

```
无重复值
```

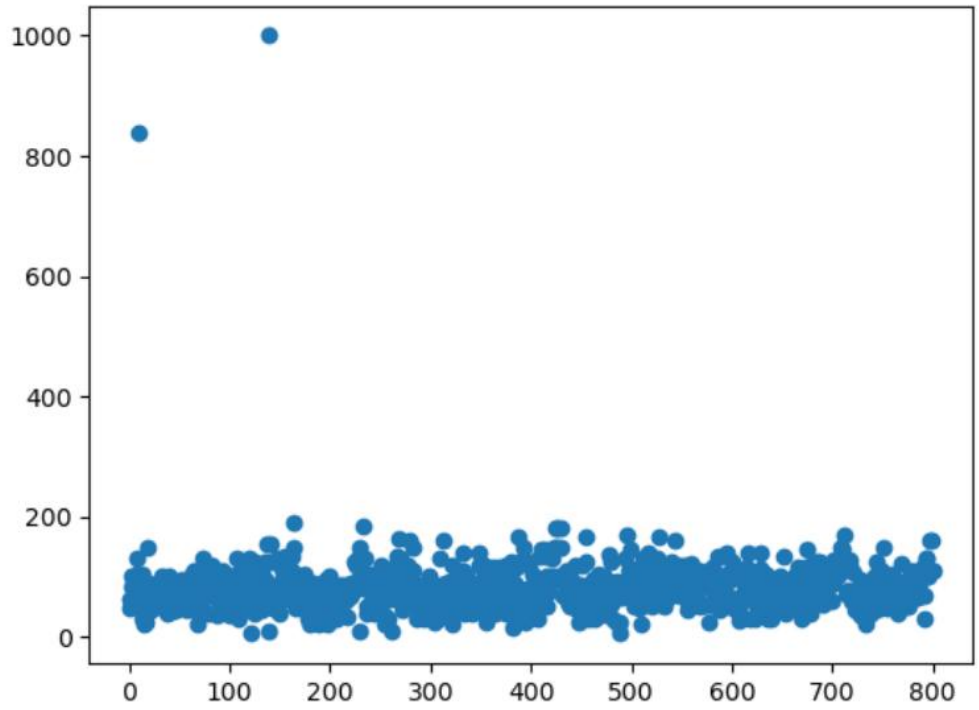
4. 处理 Attack 列的异常值

问题识别：Attack 列（宝可梦攻击值）存在远超合理范围的异常值（如数值过大），需通过

可视化识别并过滤。处理步骤：
可视化识别异常值：通过散点图直观观察 Attack 列的数值分布，发现极端异常点：

[17]:
```
import matplotlib.pyplot as plt
plt.scatter(range(0,df_clean.shape[0]),df_clean.iloc[:,6])
```

[17]:
```
<matplotlib.collections.PathCollection at 0x212ffa42310>
```



数据类型转换与异常值过滤：先确保 Attack 列为数值类型，再过滤掉攻击值大于 400 的异常数据，并删除缺失值：

[18]:
```
df_clean['Attack'] = pd.to_numeric(df_clean['Attack'], errors='coerce')
df_clean_2 = df_clean[df_clean['Attack'] <= 400].dropna(subset=['Attack'])
df_clean_2
```

```
C:\Users\53207\AppData\Local\Temp\ipykernel_2492\2484456789.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_clean['Attack'] = pd.to_numeric(df_clean['Attack'], errors='coerce')
```

[18]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49.0	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62.0	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82.0	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100.0	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52.0	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
801	719	Diancie	Rock	Fairy	600	50	100.0	150	100	150	50	6	1
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160.0	110	160	110	110	6	1
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110.0	60	150	130	70	6	1
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160.0	60	170	130	80	6	1
805	721	Volcanion	Fire	Water	600	80	110.0	120	130	90	70	6	1

798 rows × 13 columns

5. 修复 Generation 与 Legendary 列的错位数据

问题识别：部分行的 Generation 列（世代）与 Legendary 列（是否传说宝可梦）数据错位（如 Generation 列出现布尔值，Legendary 列出现整数）。处理步骤：

定位错位行：判断 Generation 列是否为整数类型，找出非整数类型的错位行索引：

```
[19]: condition=df_clean_2['Generation'].apply(lambda x: isinstance(x,(int)))
      row_n=df_clean_2.index[condition==False].tolist()
      row_n
```

```
[19]: [771]
```

交换错位列的值：定义函数交换指定行的 Generation 与 Legendary 列值，修复数据：

```
[20]: def swap_values(df,row1,col1,col2):
          df.loc[row1,col1],df.loc[row1,col2]=df.loc[row1,col2],df.loc[row1,col1]
      swap_values(df_clean_2,771,'Generation','Legendary')
      df_clean_2
```

[20]:

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	1	Bulbasaur	Grass	Poison	318	45	49.0	49	65	65	45	1	False
1	2	Ivysaur	Grass	Poison	405	60	62.0	63	80	80	60	1	False
2	3	Venusaur	Grass	Poison	525	80	82.0	83	100	100	80	1	False
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100.0	123	122	120	80	1	False
4	4	Charmander	Fire	NaN	309	39	52.0	43	60	50	65	1	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
801	719	Diancie	Rock	Fairy	600	50	100.0	150	100	150	50	6	1
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160.0	110	160	110	110	6	1
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110.0	60	150	130	70	6	1
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160.0	60	170	130	80	6	1
805	721	Volcanion	Fire	Water	600	80	110.0	120	130	90	70	6	1

798 rows × 13 columns

（三）数据清洗结果验证

清洗完成后，通过 df_clean_2.info()和 df_clean_2.describe()查看数据质量：

```
# 查看数据基本信息（列类型、非空值数量）
print(df_clean_2.info())
# 查看数值型列的统计描述（均值、标准差、最值等）
print(df_clean_2.describe())
```

验证结果：

无无意义尾行、重复行；Type 2 列无异常数值，仅保留合理字符串或 NaN；Attack 列无极端异常值，数值分布合理；Generation 与 Legendary 列无错位数据，类型正确。

结论分析与体会：

（一）实验收获

我通过本次实验掌握了数据清洗的核心流程：从数据加载、问题识别（缺失值、异常值、重复值、错位数据）到针对性处理，形成了系统化的脏数据解决方案。

熟练使用 pandas 库进行数据操作（如 drop_duplicates 去重、loc 定位修改、apply 函数批量处理），以及 matplotlib 可视化辅助识别异常值。

理解了数据质量对后续分析的重要性 —— 清洗后的数据可以有效避免分析偏差，为建模（如宝可梦属性相关性分析、战斗力预测）提供可靠基础。

（二）问题与改进方向

实验可以在这些方面改进：

1.对 Type 2 列的异常值直接替换为 NaN，未进一步结合业务逻辑（如宝可梦属性规则）填充合理值。

2.异常值过滤（如 Attack≤400）依赖经验阈值，可进一步通过四分位法（IQR）实现自动化异常值检测。

改进方向：

1.基于宝可梦属性的业务规则，对 Type 2 列的缺失值进行合理填充（如根据 Type 1 列关联填充）。

2.使用四分位法（IQR = Q3 - Q1，过滤小于 Q1-1.5IQR 或大于 Q3+1.5IQR 的值）替代经验阈值，提升异常值处理的客观性。