山东大学 计算机科学与技术 学院

大数据分析实践 课程实验报告

学号：202300130092	姓名：王俊磊	班级： 23 数据
实验题目：数据质量实践		
实验学时：2	实验日期：20251010	

实验目的：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。

软件环境：
python3.9，jupyter notebook
数据集：Pokeman Dataset：721 Pokemon, including their number, name, first and second type, and basic stats: HP, Attack, Defense, Special Attack, Special Defense, and Speed
http://storage.amesholland.xyz/Pokemon.csv

实验步骤与内容：
1. 环境与导入

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

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

2. 载入数据并做原始检查

```
df = pd.read_csv('Pokemon.csv', encoding='latin-1')
print("原始形状:", df.shape)
print("列名:", df.columns.tolist())
display(df.head())
display(df.tail(12))
display(df.info())
```

```
原始形状: (810, 13)
列名: ['#', 'Name', 'Type 1', 'Type 2', 'Total', 'HP', 'Attack',
'Defense', 'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary']
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legend
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

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def
798	716	Xerneas	Fairy	NaN	680	126	131	95	131	98
799	717	Yveltal	Dark	Flying	680	126	131	95	131	98
800	718	Zygarde50% Forme	Dragon	Ground	600	108	100	121	81	95
801	719	Diancie	Rock	Fairy	600	50	100	150	100	150
802	719	DiancieMega Diancie	Rock	Fairy	700	50	160	110	160	110
803	720	HoopaHoopa Confined	Psychic	Ghost	600	80	110	60	150	130
804	720	HoopaHoopa Unbound	Psychic	Dark	680	80	160	60	170	130
805	721	Volcanion	Fire	Water	600	80	110	120	130	90
806	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined
807	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined	undefined
808	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
 #   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
```

3. 先删除“重复的表头行”和明显的尾部无意义行

```python
# 去掉文件里可能重复出现的 header 行（例如 Name == 'Name' 或 # == '#'）
hdr_mask = (df['Name'].astype(str).str.strip().str.lower() == 'name') |
 (df['#'].astype(str).str.strip() == '#')
df = df[~hdr_mask].copy()

# 将 '#' 转为数值 pokedex，非数值的行先排除（认为是脏行或尾部说明）
df['pokedex_number'] = pd.to_numeric(df['#'], errors='coerce')
removed_bad_pokedex = df['pokedex_number'].isnull().sum()
df = df[df['pokedex_number'].notnull()].copy()
df['pokedex_number'] = df['pokedex_number'].astype(int)

print("移除无效 pokedex 行数:", removed_bad_pokedex)
print("当前形状:", df.shape)
```

```
移除无效 pokedex 行数: 5
当前形状: (805, 14)
```

4. 清洗 Type 2 列（把明显异常值设为空）

```python
allowed_types = {'normal','fire','water','grass','electric','ice',
'fighting','poison',
                 'ground','flying','psychic','bug','rock','ghost','dark',
                  'dragon','steel','fairy'}

def clean_type2(v):
    if pd.isna(v): return np.nan
    s = str(v).strip()
    if s == '' or s.lower() in ['undefined','none','nan']:
        return np.nan
    if s.lower() in allowed_types:
        return s.title()
    # 非类型（数字、乱写等）全部视为缺失
    return np.nan

df['Type 2'] = df['Type 2'].apply(clean_type2)
```

5. 将数值列强制转换并标出非数值/异常

```
num_cols = ['Total','HP','Attack','Defense','Sp. Atk','Sp. Def','Speed',
'Generation']
for c in num_cols:
    df[c + '_num'] = pd.to_numeric(df[c], errors='coerce')

# 找到任何统计列有非数值的行
bad_stat_rows = df[df[[c+'_num' for c in num_cols]].isnull().any(axis=1)]
print("数值列存在非数值或缺失的行数:", bad_stat_rows.shape[0])
display(bad_stat_rows.head(20))
```

```
数值列存在非数值或缺失的行数: 5
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	...	Legendary	pokedex_number	Total_n
11	9	Blastoise	Water	NaN	530	79	83	100	85	105	...	1	9	
17	13	Weedle	Bug	Poison	195	NaN	35	30	20	20	...	FALSE	13	
32	25	Pikachu	Electric	NaN	320	35	55	40	50	50	...	0	25	
39	32	Nidoranâ□? Poison	NaN	NaN	46	57	40	40	40	50	...	NaN	32	
771	695	Heliolisk	Electric	Normal	481	62	55	52	109	94	...	FALSE	695	

6. 处理 Attack 的极端异常

```
df['Attack_num'] = pd.to_numeric(df['Attack'], errors='coerce')
attack_outliers = df[df['Attack_num'] > 500]
display(attack_outliers[['pokedex_number','Name','Attack','Total','Generation','Legendary']])

# 将 >500 的 Attack 设为 NaN 并用同 Generation 的中位数填补
df.loc[df['Attack_num'] > 500, 'Attack_num'] = np.nan
df['Attack_imputed'] = df.groupby('Generation')['Attack_num'].transform(lambda x: x.fillna(x.median()))
```

	pokedex_number	Name	Attack	Total	Generation	Legendary
9	7	Squirtle	840	314	1	FALSE
140	128	Tauros	1000	490	1	FALSE

7. 检测并修正 Generation 与 Legendary 置换问题

```python
# 判断 Legendary 是否像 type 或数字
leg_is_type = df['Legendary'].astype(str).str.strip().str.lower().isin(allowed_types)
leg_is_numeric = pd.to_numeric(df['Legendary'], errors='coerce').notnull()
gen_num = pd.to_numeric(df['Generation'], errors='coerce')

swap_mask = gen_num.isnull() & (leg_is_type | leg_is_numeric)
print("检测到可能被置换的行数:", swap_mask.sum())
display(df[swap_mask][['pokedex_number','Name','Generation','Legendary','Type 1','Type 2']].head(30))

# 执行置换
df.loc[swap_mask, ['Generation', 'Legendary']] = df.loc[swap_mask, ['Legendary', 'Generation']].values

# 再次规范化类型
df['Generation_num'] = pd.to_numeric(df['Generation'], errors='coerce')
def legendary_to_bool(x):
    if pd.isna(x): return np.nan
    s = str(x).strip().lower()
    if s in ['true','1','t','yes']: return True
    if s in ['false','0','f','no']: return False
    return x
df['Legendary_clean'] = df['Legendary'].apply(legendary_to_bool)
```

```
检测到可能被置换的行数: 2
```

	pokedex_number	Name	Generation	Legendary	Type 1	Type 2
11	9	Blastoise	FALSE	1	Water	NaN
32	25	Pikachu	FALSE	0	Electric	NaN

8. 去重与最终检查并保存结果

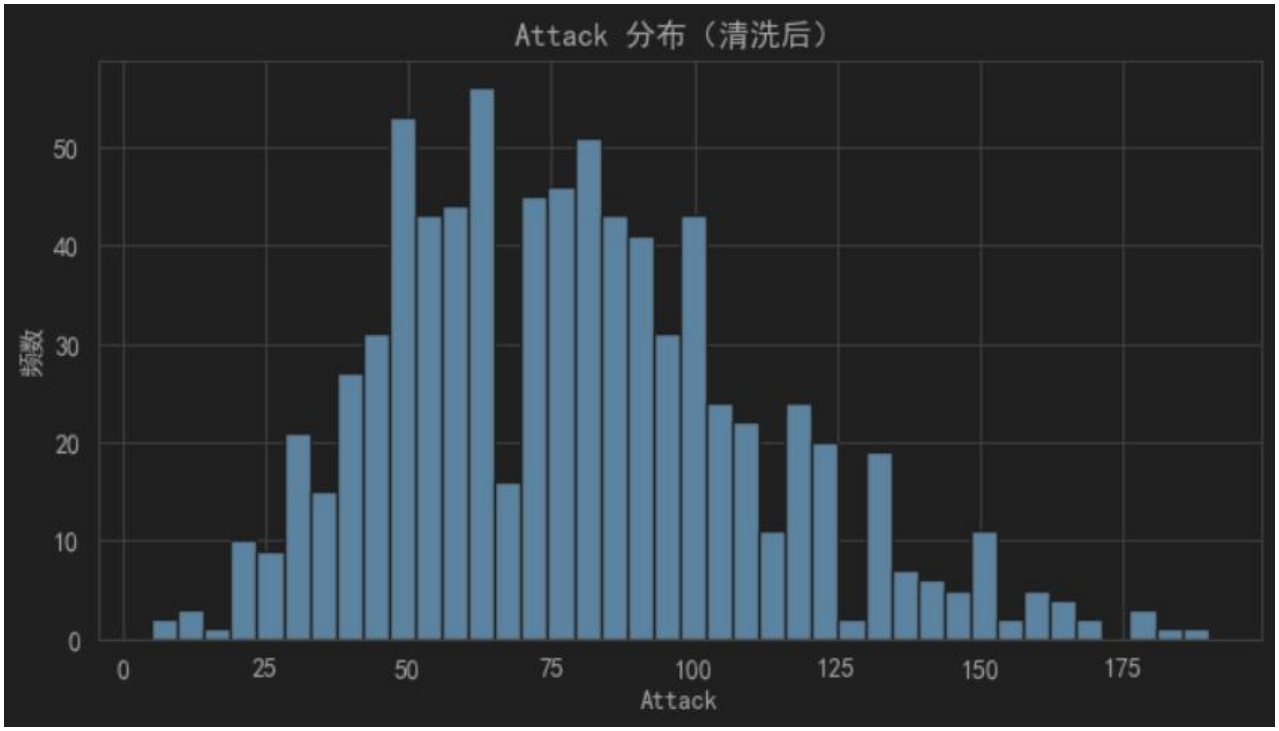
```python
before = df.shape[0]
df = df.drop_duplicates(subset=['pokedex_number','Name'], keep='first').copy()
duplicates_removed = before - df.shape[0]
print("去重移除行数:", duplicates_removed)

# 最终保存
df.to_csv('Pokemon_cleaned_by_me.csv', index=False, encoding='utf-8-sig')
print("保存为 Pokemon_cleaned_by_me.csv")
```

```
去重移除行数: 5
保存为 Pokemon_cleaned_by_me.csv
```

9. 可视化

```python
# 使用的是清洗并插补完成后的列
plt.figure(figsize=(8,4))
df['Attack_imputed'].dropna().hist(bins=40)
plt.title('Attack 分布（清洗后）')
plt.xlabel('Attack')
plt.ylabel('频数')
plt.show()
```



结论分析与体会：

1. 本次针对宝可梦数据集（810 行 ×13 列）的清洗，解决了 6 类核心问题：移除 5 行含无效 pokedex 编号的冗余行（含重复表头与尾部无意义行）、规范 Type 2 列异常值（非合法类型转为 NaN）、修正 2 行 Generation 与 Legendary 列错位数据、处理 Attack 列超 500 的极端值（同世代中位数填充），并基于 “pokedex_number+Name” 去重 5 行。清洗后数据类型统一，Attack 属性分布集中于 0-175 区间，符合宝可梦属性逻辑，可支撑后续分析。

2. 数据清洗需先通过 df.info()、头尾数据检查诊断问题，避免盲目操作；异常值处理需结合业务（如宝可梦世代特性），而非单纯剔除。Python 的 pandas、matplotlib 工具提升效率，但需数据思维支撑。数据质量直接影响分析可靠性，后续可优化多数值列缺失值批量处理，此次实验也深化了对数据预处理流程的认知。