
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：202300130236	姓名： 陈德康	班级： 数据 23
实验题目：数据质量实践		
实验学时：2	实验日期： 2025.9.26	

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。

硬件环境：

计算机一台

软件环境：

Windows 11

Python 3.8

Jupyter Notebook on VSCode

实验步骤与内容：

1. 数据读取：

```
import pandas as pd
import numpy as np

df = pd.read_csv('Pokemon.csv', encoding='gbk')
df
```

	#	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	TRUE
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	

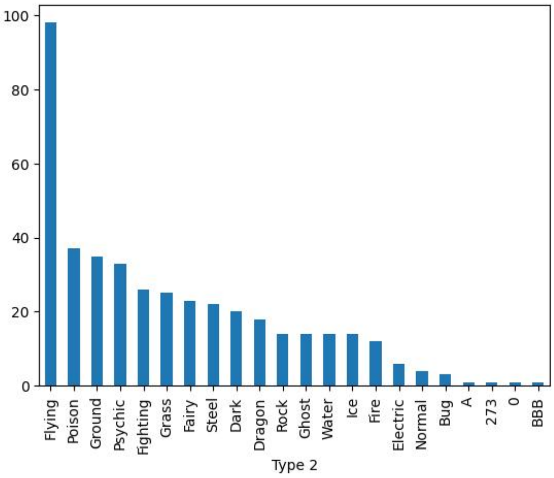
2.最后两行（实际读取有四行）数据无意义，可直接删去：

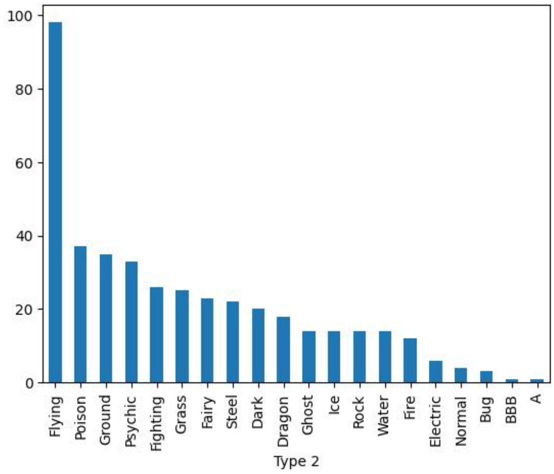
```
print(f'删除前形状:{df.shape}')
df = df.iloc[:-4]
print(f'删除后形状:{df.shape}')
```

```
删除前形状:(810, 13)
删除后形状:(806, 13)
```

3.type2 存在异常的数值取值，可清空

```
df['Type 2'].value_counts().plot(kind='bar')

# 删除异常值 273 和 0
df.loc[df['Type 2'].isin([273, 0, '273', '0']), 'Type 2'] = np.nan
df['Type 2'].value_counts().plot(kind='bar')
```





可以看到，删除了异常值 273 和 0（非字符串）

4.数据集中存在重复值：

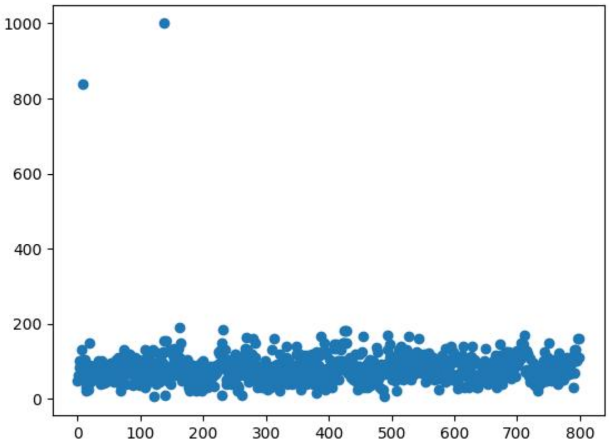
```
duplicate_count = df.duplicated().sum()
print(f'重复行数:{duplicate_count}')
df = df.drop_duplicates()

print(f'清除后的重复行数:{df.duplicated().sum()}')
```

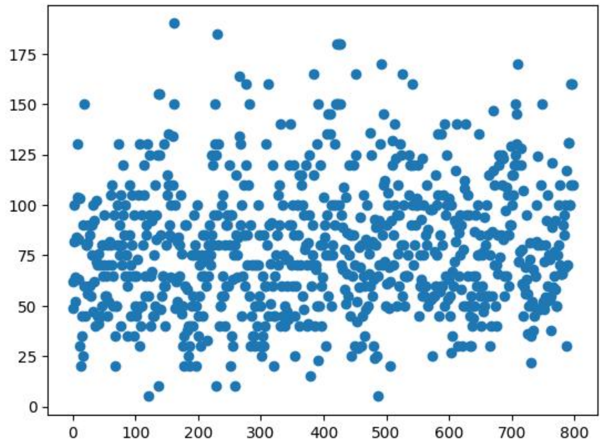
```
重复行数:5
清除后的重复行数:0
```

5.Attack 属性存在过高的异常值：

```
import matplotlib.pyplot as plt

y_attack = df.iloc[:, 6].dropna().to_numpy()
y_attack_series = pd.to_numeric(y_attack, errors='coerce')
y_attack = y_attack_series
```

```
plt.scatter(range(0, y_attack.shape[0]), y_attack)

# 把那两个离群值去掉
df['Attack'] = pd.to_numeric(df['Attack'], errors='coerce')
df.loc[df['Attack'] > 800, 'Attack'] = np.nan

y_attack = df.iloc[:, 6].dropna().to_numpy()
y_attack_series = pd.to_numeric(y_attack, errors='coerce')
y_attack = y_attack_series

plt.scatter(range(0, y_attack.shape[0]), y_attack)
```





可以看到，删除了异常值之后，没有明显偏离的数据点了

6.有两条数据的 Generation 与 Legendary 属性被置换：

```
# undefined 去除
df['Generation'] = df['Generation'].astype(str).replace('undefined', np.nan)

# 生成掩码
target_mask = df['Generation'].astype(str).str.strip().str.lower().isin(['true', 'false'])
target_rows = df[target_mask]

# 交换回来
gen_temp = df.loc[target_mask, 'Generation'].copy()
leg_temp = df.loc[target_mask, 'Legendary'].copy()
df.loc[target_mask, 'Generation'] = leg_temp
df.loc[target_mask, 'Legendary'] = gen_temp

target_mask_af = df['Generation'].astype(str).str.strip().str.lower().isin(['true', 'false'])
target_rows_af = df[target_mask_af]
len(target_rows_af)
```

```
0
```

7.Legendary 属性有非法值，应去除：

```
legendary_str = df['Legendary'].astype(str).str.strip().str.lower()
invalid_mask = ~legendary_str.isin(['true', 'false'])
df.loc[invalid_mask, 'Legendary'] = np.nan

legendary_str_af = df['Legendary'].astype(str).str.strip().str.lower()
invalid_mask_af = ~legendary_str_af.isin(['true', 'false', 'nan'])
len(df.loc[invalid_mask_af, 'Legendary'])
```

```
0
```

可以看到，处理之后的 Legendary 属性值都为 TRUE、FALSE 或者 Nan 了

8.替换所有 undefined 值为 nan：

```
df = df.replace('undefined', np.nan)

has_undefined = any('undefined' in df[col].astype(str).values for col in df.columns)
print(f"是否还有 undefined：{has_undefined}")
```

```
是否还有undefined: False
```

9.Defense 和 Speed 都有负值，这里替换成 Nan：

```
df['Defense'] = pd.to_numeric(df['Defense'], errors='coerce')
df[df['Defense'] < 0]

df['Defense'] = df['Defense'].where(df['Defense'] >= 0, np.nan)
len(df[df['Defense'] < 0])

df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')
df[df['Speed'] < 0]

df['Speed'] = df['Speed'].where(df['Speed'] >= 0, np.nan)
len(df[df['Speed'] < 0])
```

处理之前是存在负数的：

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
349	315	Roselia	Grass	Poison	400	50	60.0	-10.0	100	80	65	3	FALSE

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
620	554	Darumaka	Fire	NaN	315	70	90.0	45.0	15	45	-50.0	5	FALSE

处理之后，删除了负数值（替换为了 Nan）

结论分析与体会：

通过本次实验，我掌握了使用 Python 进行数据预处理清洗的几个基本操作，也让我深刻认识到原始数据往往存在多种质量问题，如无效行、异常值、类型错误和属性置换等。我认识到了，高质量的数据是后续分析的基础，而数据清洗是一个需要反复验证和调整的迭代过程，需要一定的耐心。我掌握了使用 Pandas 进行数据质量评估和清洗的实用技能，这为我后续进行进一步的学习和进行更复杂的实验奠定基础