Swiss UMEF
UNIVERSITY OF APPLIED SCIENCES INSTITUTE

Big Data Analytics
大数据分析

# 04: In-Memory Analytics with Pandas. Data Cleaning and Preparation
04：用Pandas进行内存分析 数据清洗与准备

Instructor: Oleh Tymchuk
讲师：奥勒·特姆恰克

#04: Agenda 课程安排

● Introduction to Data Cleaning	**数据清洗**简介
● Handling Missing Data	处理缺失数据
● Detecting and Removing Duplicates	检测和移除重复项
● Fixing Data Types and Formatting	**修正数据**类型和格式化
● Dealing with Outliers and Extreme Values	处理离群值和极端值
● Practical cases	实际**案例**
● Useful Links	实用链接

Introduction to Data Cleaning
数据清洗简介

Why Data Cleaning is Important? 为什么数据清洗很重要？

- Real-world data is often messy: missing values, duplicates, inconsistencies
- 真实世界的数据通常是混乱的：缺失值、重复项、不一致性
- Data Cleaning is a process of detecting and correcting (or removing) errors, inconsistencies, and inaccuracies in datasets
- 数据清洗是检测和纠正（或移除）数据集中错误、不一致性和不准确性的过程
- Clean data ensures data quality and reliability for accurate analysis
- 清洁的数据确保了数据质量和可靠性，以便进行准确的分析
- Saves time and resources by preventing errors in downstream tasks
- 通过防止下游任务中的错误，节省时间和资源
- A crucial step in any data-driven workflow
- 任何数据驱动工作流程中的关键步骤

Common Data Issues 常见的数据问题

● Missing or null values	缺失或空值
● Duplicates and inconsistencies	重复项和不一致性
● Incorrect data types	错误的数据类型
● Outliers and incorrect formatting	离群值和错误的格式化

Handling Missing Data
处理缺失数据

Identifying Missing Data 识别缺失数据

- Missing data refers to the absence of values in a dataset
- 缺失数据指的是数据集中值的缺失
- Represented as NaN (Not a Number) or None in Pandas
- 在Pandas中表示为NaN（非数字）或None
- Why Identify Missing Data? 为什么要识别缺失数据？
  - Missing data can lead to incorrect analysis or modeling results.
  - 缺失数据可能导致分析或建模结果错误
  - Helps decide the appropriate strategy for handling it.
  - 有助于确定恰当策略来处理缺失数据
- How to Identify Missing Data 如何识别缺失数据

check for missing values: 检查缺失值：

```
df.isnull()
```

count missing values per column: 按列计数缺失值：

```
df.isnull().sum()
```

Strategies for Handling Missing Data 处理缺失数据的策略

Remove Missing Data 移除缺失数据

```
df.dropna()                          # Drop rows with any missing
values                               #移除包含任何缺失值的行

df.dropna(axis=1)            # Drop columns with any missing
values                       # 移除包含任何缺失值的列
```

Use when missing data is minimal and doesn’t affect analysis

当缺失数据很少且不影响分析时使用这种方法

Strategies for Handling Missing Data 处理缺失数据的策略

Fill Missing Data 填充缺失数据

- Fill with a constant value **用常数值填充**

```
df.fillna(0)   # Fill with 0   #用0填充
```

- Fill with statistical measures **用统计方法填充**

```
df.fillna(df.mean())   # Fill with column mean#用列均值填充
df.fillna(df.median())   # Fill with column median#用列中位数填充
```

- Forward or backward fill **前向或后向填充**

```
df.fillna(method='ffill')   # Forward fill #向前填充
df.fillna(method='bfill')   # Backward fill#向后填充
```

Strategies for Handling Missing Data 处理缺失数据的策略

Estimate missing values using interpolation 使用插值估计缺失值

```
df.interpolate()
```

Use machine learning models to predict missing values

使用机器学习模型预测缺失值

Example: K-Nearest Neighbors (KNN) imputation

举例：K最近邻（KNN）插补

Best Practices 最佳实践

- Understand the reason for missing data (e.g., random or systematic)
- 了解缺失数据的原因（例如，随机或系统性）
- Choose a strategy based on the context and impact on analysis
- 根据上下文和对分析的影响选择策略

Example. Mean 举例 均值

```
data = pd.Series([30, np.nan, 10, 40, np.nan, 20, np.nan])
```

```
filled_mean = data.fillna(data.mean())
```

- **How it works?** **工作原理？**
  - The mean is calculated as:
    均值计算为：

$$\frac{30+10+40+20}{4} = 25$$

  - Every `NaN` is replaced with `25`. 所有NaN替换为25
  - *Result:** 结果：

```
0     30.0
1     25.0
2     10.0
3     40.0
4     25.0
5     20.0
6     25.0
dtype: float64
```

*When to use it?**

✅ If data is **evenly distributed** without extreme values (outliers).

✅ Example: **Filling missing test scores when calculating class averages.**

什么时候用？
如果数据**均匀分布**且没有极端值（离群值）。
例如：**在计算班级平均分时填补缺失的考试成绩**。

Example. Median 举例 中位数

```
data = pd.Series([30, np.nan, 10, 40, np.nan, 20, np.nan])
```

```
filled_median = data.fillna(data.median())
```

- **How it works? 工作原理？**
  - The median is the middle value of the sorted list: `[10, 20, 30, 40]`. 中位数是排序列表的中间值：[10, 20, 30, 40]。
  - Since we have **an even number of values**, the median is: 由于我们有偶数个数值，中位数是：

$$\frac{20+30}{2}=25$$

  - Every `NaN` is replaced with `25`. 所有NaN替换为25

Result: 结果：

```
0    30.0
1    25.0
2    10.0
3    40.0
4    25.0
5    20.0
6    25.0
dtype: float64
```

When to use it?

✅ If data has **outliers** (e.g., extreme high or low values).

✅ Example: **Filling missing student salaries in a dataset (since some may have very high salaries, affecting the mean).**

什么时候用这种方法？

如果数据有**离群值**（例如，极高或极低的值）。

例如：**填补数据集里缺失的学生工资（因为有些人可能工资非常高，影响平均值）**。

Example. Interpolate 举例 插值

```
data = pd.Series([30, np.nan, 10, 40, np.nan, 20, np.nan])
filled_interpolate = data.interpolate()
```

- **How it works? 工作原理？**
  - Missing values are **estimated based on nearby values**. 缺失值**基于邻近值估算**。
  - Calculation: 计算：
    1. Between 30 and 10: 30和10之间

$$\frac{30+10}{2} = 20.0$$

    2. Between 10 and 40: 10和40之间

$$10 + \frac{40-10}{2} = 25.0$$

    3. Between 40 and 20: 40和20之间

$$40 + \frac{20-40}{2} = 30.0$$

Result: 结果：

```
0    30.0
1    20.0
2    10.0
3    40.0
4    25.0
5    20.0
6    30.0
dtype: float64
```

When to use it? 什么时候用这种方法？

✅ If data represents **a continuous process**, such as time series data. 如果数据表示一个连续过程，例如时间序列数据。

✅ Example: **Filling missing temperature readings from a weather station.** 例如：**填补气象站缺失的温度读数。**

Comparison Table 比较表

Method	When to Use?	Example
`mean()` (average)	Data is evenly distributed, no outliers	Student test scores
`median()`	There are outliers, need a "central" value	Student salaries
`interpolate()`	Data changes smoothly over time	Temperature sensor data

方法	什么时候用？	举例
`mean()` (av 均值	**数据均匀分布，没有离群值**	**学生考试成绩**
`median()` 中位数	**存在离群值，需要一个“中间”值**	**学生工资**
`interpolat` 插值	**数据随时间平滑变化**	**温度传感器数据**

Comparison Table 比较表

Method	How it works?	When to use?
`fillna(value)`	Replaces `NaN` with a fixed value	When `NaN` means missing data (e.g., warehouse stock)
`fillna(method='ffill')`	Copies the previous value	Time series, such as temperature or sales data
`fillna(method='bfill')`	Copies the next value	When future values are more reliable (e.g., expected payments)
`fillna(limit=n, method=...)`	Limits the number of copied values	When long gaps shouldn't be blindly filled
`dropna()`	Removes rows with `NaN`	When missing data is minimal and can be ignored

方法	工作原理	什么时候用？
`fillna(value)`	用固定值替换NaN	当NaN表示缺失数据时（例如，仓库库存）
`fillna(method='ffill')`	复制前一个值	时间序列数据，如温度或销售数据
`fillna(method='bfill')`	复制下一个值	当未来值更可靠时（例如，预期付款）
`fillna(limit=n, method=...)`	限制复制值的数量	当不应盲目填充长时间间隔时
`dropna()`	删除包含NaN的行	当缺失数据很少且可以忽略时

Detecting and Removing Duplicates
检测和移除重复项

Identifying and Removing Duplicates 识别和移除重复项

What are Duplicates? 什么是重复项？

- Duplicates are repeated rows in a dataset. 重复项是数据集中重复的行
- They can occur due to data entry errors, merging datasets, or other reasons.
- 出现的原因可能是数据输入错误、合并数据集或其他原因

Why Remove Duplicates? 为什么移除重复项？

- Duplicates can skew analysis and lead to incorrect results.
- 重复项可能扭曲分析并导致错误结果
- They waste storage and computational resources.
- 重复项浪费存储和计算资源

Detecting duplicates: `df.duplicated()` 检测重复项

Removing duplicates: `df.drop_duplicates()` 移除重复项

Fixing Data Types and Formatting
修正数据类型和格式化

Handling Incorrect Data Types

What are Incorrect Data Types? 什么是错误数据类型？

- Data stored in a format that doesn’t match its intended type (e.g., numbers stored as strings, dates stored as text)
- 数据以与其预期类型不匹配的格式存储（例如，数字存储为字符串，日期存储为文本）

Why Fix Data Types? 为什么修正数据类型？

- Ensures proper analysis and computation (e.g., arithmetic operations on numeric data)
- 确保正确的分析和计算（例如，对数值数据进行算术运算）
- Enables use of specialized functions (e.g., date operations)
- 启用使用专用函数（例如，日期操作）

Common Data Type Issues 常见的数据类型问题

- Numeric data stored as strings (e.g., "123" instead of 123) 数值数据存储为字符串（例如，“123”而不是123）
- Dates stored as strings (e.g., "2023-10-01" instead of datetime)
- 日期存储为字符串（例如，“2023-10-01”而不是日期时间）
- Categorical data stored as strings or numbers 分类数据存储为字符串或数字

How to Fix Data Types 如何修正数据类型

- Converting data types: .astype() 转换数据类型：.astype()
- Parsing dates: pd.to_datetime() 解析日期：pd.to_datetime()
- Handling categorical data: .astype('category') 处理分类数据：.astype('category')

Fixing Inconsistent Data　修正不一致的数据

What is Inconsistent Data? 什么是不一致数据？

- Data that doesn’t follow a standard format or convention (e.g., mixed cases, extra spaces, inconsistent units)
- 数据不遵循标准格式或惯例（例如，混合大小写，多余空格，不一致的单位）

Why Fix Inconsistent Data?为什么要修正不一致数据？

- Ensures uniformity for accurate analysis and reporting 确保分析和报告的一致性
- Improves readability and usability of the dataset 提高数据集的可读性和可用性

Common Inconsistencies 常见的不一致性

- Text: Mixed cases ("New York" vs. "new york"), extra spaces 文本：混合大小写，多余空格
- Units: Inconsistent measurements (e.g., "kg" vs. "lbs")　单位：不一致的测量方法（例如，“千克”与“磅”）
- Categories: Different spellings or representations (e.g., "USA" vs. "U.S.A.")
  类别：不同的拼写或表示（例如，“USA”与“U.S.A.”）

How to Fix Inconsistent Data 如何修正不一致数据

- Standardizing text format (lowercase, trimming spaces) 标准化文本格式（小写，修剪空格）
- Correcting categorical inconsistencies (e.g., "USA" vs. "United States")
- 纠正分类不一致（例如，“USA”与“United States”）
- Replacing incorrect values: .replace()　替换错误值：.replace()

Dealing with Outliers and Extreme Values
处理离群值和极端值

Detecting and Handling Outliers 检测和处理离群值

What are Outliers? 什么是离群值？

- Outliers are data points that are significantly different from the rest of the data
- They can occur due to errors, variability, or rare events
- 离群值是与其余数据显著不同的数据点
- 离群值可能由于错误、变异性或罕见事件而出现

Why Handle Outliers? 为什么处理离群值？

- Outliers can skew analysis, affect statistical measures (e.g., mean, variance), and impact machine learning models
- 离群值可能扭曲分析，影响统计量（例如，均值、方差）并影响机器学习模型
- Deciding whether to keep, remove, or transform outliers depends on the context
- 决定是否保留、移除或转换离群值取决于上下文

Detecting Outliers 检测离群值

📊 **Visual Methods: 可视化方法：**

✅ Boxplots – Outliers appear as points outside the "whiskers"

箱线图 – 离群值出现在“须”外的点

📉 **Statistical Methods: 统计方法：**

✅ Z-score Z分数

Measures how many standard deviations a value is from the mean

Common threshold: |Z| > 3 (potential outlier)

衡量一个值与均值的标准差距离

常见阈值：|Z| > 3（潜在离群值）

✅ Interquartile Range (IQR): 四分位距（IQR）：

IQR = Q3 - Q1 (Q1 = 25th percentile, Q3 = 75th percentile)

IQR = Q3 - Q1 (Q1 = 第25百分位数，Q3 = 第75百分位数)

Outliers are values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR

离群值是低于Q1 - 1.5IQR或高于Q3 + 1.5IQR的值

Handling Outliers 处理离群值

- **Remove Outliers 移除离群值**
  Drop rows containing outliers 删除包含离群值的行
- **Cap/Floor Outliers 限制离群值**
  Replace outliers with a maximum or minimum threshold value
  用最大或最小阈值替换离群值
- **Transform Data 转换数据**
  Apply log transformation or scaling to reduce the impact of outliers
  应用对数转换或缩放以减少离群值的影响
- **Keep Outliers 保留离群值**
  Retain outliers if they are meaningful (e.g., rare but valid events)
  如果离群值有意义（例如，罕见但有效的事件）则保留

Data Cleaning Best Practices 数据清洗最佳实践

- Always check the dataset before analysis
- 总是在分析前检查数据集
- Keep track of changes for reproducibility
- 追踪改动，以实现可重复性
- Validate results after cleaning
- 清洗后验证结果

Practical cases
实际案例

Useful Links 实用链接

Pandas Cheat Sheet Pandas	速查表
Working with missing data	处理缺失数据
Outlier Detection Techniques in Python	Python中的离群值检测方法