CoGrammar

Welcome to this session

Skills Bootcamp:

Data Manipulation and Cleaning (Practical Live Demonstration)

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Full Stack Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. We will be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: ***Feedback on Lectures.***
- Find all the lecture **content** in your **Lecture Backpack** on GitHub.
- If you are hearing impaired, kindly use your computer's function through Google chrome to enable captions.

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 1 - Initial Requirements

Specific achievements within the first two weeks of the program.

To meet this criterion, students need to, by no later than **01 December 2024 (C11)** or **22 December 2024 (C12):**

- **Guided Learning Hours** (GLH)**:** Attend a minimum of 7-8 GLH per week (lectures, workshops, or mentor calls) for a total minimum of **15 GLH**.
- **Task Completion:** Successfully complete the **first 4 of the assigned tasks**.

✅ Criterion 2 - Mid-Course Progress

Progress through the successful completion of tasks within the first half of the program.

To meet this criterion, students should, by no later than **12 January 2025 (C11)** or **02 February 2025 (C12):**

- **Guided Learning Hours** (GLH)**:** Complete at least **60 GLH**.
- **Task Completion :** Successfully complete the **first 13 of the assigned tasks**.

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3 – End-Course Progress

Showcasing students' progress nearing the completion of the course.

To meet this criterion, students should:

- **Guided Learning Hours** (GLH)**:** Complete the **total minimum required GLH,** by the **support end date.**
- **Task Completion : Complete all mandatory tasks**, including any necessary resubmissions, by the end of the bootcamp, **09 March 2025 (C11)** or **30 March 2025 (C12).**

✅ Criterion 4 - Employability

Demonstrating progress to find employment.

To meet this criterion, students should:

- **Record an Interview Invite:** Students are required to record proof of invitation to an interview by **30 March 2025 (C11)** or **04 May 2025 (C12)**.
  - **South Holland Students** are required to proof and interview by **17 March 2025.**
- **Record a Final Job Outcome :** Within 12 weeks post-graduation, students are required to record a job outcome.

CoGrammar

Stay Safe Series:

Mastering Online Safety One week at a Time

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the *Stay Safe Series* will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

Patch it Up:

The Importance of Regular Software Updates

1. Fixes security vulnerabilities in your system, preventing cyber attacks.
2. Ensures your devices are protected from newly discovered threats.
3. Improves software performance and reliability.
4. Enable automatic updates for operating systems and software where possible.
5. Regularly check for updates on apps that don’t update automatically.
6. Set reminders to check for updates on devices you use less frequently.
7. Prioritize updates labeled as critical or security updates.

Why is data cleaning important in data analysis?

A. To save storage space
B. To ensure accurate and reliable insights
C. To make data visually appealing
D. To remove unused data

What is a common issue when working with datasets?

A. Missing values
B. Overly clear formats
C. Consistently structured data
D. Homogeneous datasets

Learning Outcomes

- Discuss the importance of data cleaning in the data science pipeline
- Identify common data quality issues and their impact
- Explain key data cleaning techniques
- Explore data manipulation strategies for better analysis.

Lecture Overview

- ➔ Introduction
- ➔ Part 1: Fundamentals of Data Cleaning
- ➔ Break
- ➔ Part 2: Advanced Cleaning and Manipulation
- ➔ Assessment and Q&A

Why Data Cleaning Matters

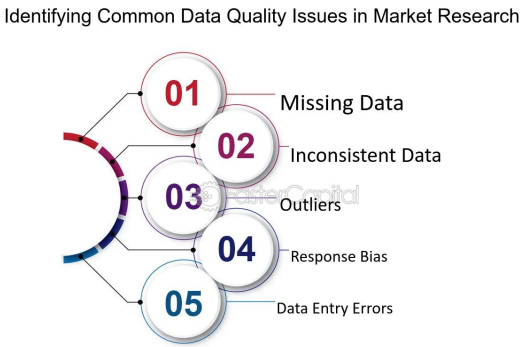
- 80% of data analysis involves cleaning and preparing data.
- Impact of Dirty Data:
  - Misleading insights.
  - Increased costs and errors.
- Example: Customer IDs missing in a sales dataset.

Why Data Cleaning Matters

- Ensures data accuracy and reliability.
- Eliminates biases and inconsistencies.
- Prepares data for meaningful analysis and modeling.
- Saves time and effort in downstream processes.
- Example:
  - "Imagine analyzing sales data with missing transaction amounts. This would distort the revenue calculations and lead to incorrect insights."

Common Data Quality Issues

- Missing Data:
  - Null or NaN values in datasets.
- Duplicates:
  - Redundant entries in data.
- Inconsistent Formats:
  - Non-standardized date formats, text inconsistencies.
- Outliers:
  - Extreme values that distort analysis.

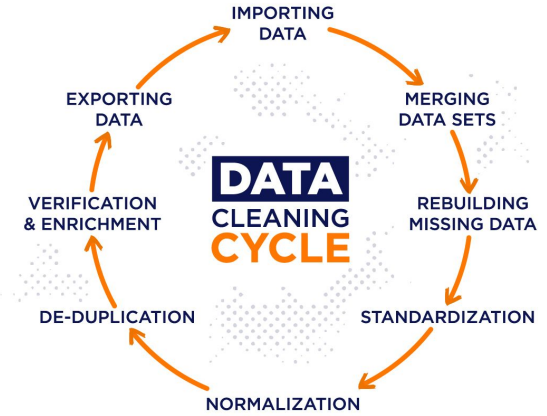


Tools for Data Cleaning

- Python Libraries:
  - Pandas: Data manipulation and cleaning.
  - NumPy: Numerical operations.
  - OpenRefine: Cleaning messy data.
  - Matplotlib/Seaborn: Visualizing data trends.
- Other Tools:
  - Excel/Google Sheets.
  - R (tidyverse).

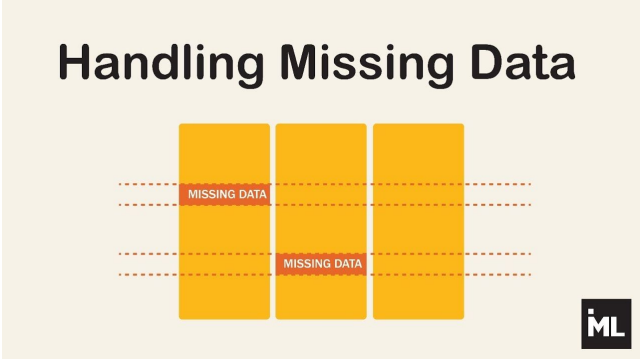
Overview of Data Cleaning Process

- Identify Issues:
  - Use exploratory data analysis (EDA).
- Handle Missing Data:
  - Impute, drop, or flag missing values.
- Standardize Data:
  - Ensure consistency in formats.
- Remove Duplicates:
  - Retain unique, valid entries.
- Address Outliers:
  - Detect and treat anomalies.



Handling Missing Data

- Strategies:
  - Deletion: Remove rows/columns with excessive missing values.
  - Imputation: Fill missing data with mean, median, or mode.
  - Flagging: Add an indicator column.
- Example:
  - Missing prices in a product dataset.



Step 1: Loading the Dataset

- Use pd.read_csv() to load CSV files.
- Inspect data with .head() and .info() to understand its structure.

```
[1]: import pandas as pd

[3]: data = pd.read_csv('Customer_Transactions.csv')
     print(data.head())

                  tranDate          custName           cardNum  zipCode  \
0  2023-09-15 20:32:41    Catherine Bell  2294637276392057     8642
1  2023-05-16 23:18:37     Parker Riddle   342160763812707    80349
2  2023-09-11 18:38:23      Brenda Baird  4137641055044779    34346
3  2023-08-04 21:42:37   Kimberly Carter  3546070762859922    47715
4  2023-09-22 08:27:40  Daniel Rodriguez   213170012973743    77790

   tranAmount
0         848
1         574
2         600
3         583
4        3636

[4]: print(data.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5500 entries, 0 to 5499
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   tranDate    5500 non-null   object
 1   custName    5500 non-null   object
 2   cardNum     5500 non-null   int64
 3   zipCode     5500 non-null   int64
 4   tranAmount  5500 non-null   int64
dtypes: int64(3), object(2)
memory usage: 215.0+ KB
None
```

Step 2: Identifying Data Issues

- Common Issues:
  - Missing Values: Check the count of nulls in each column.
  - Duplicates: Identify and quantify repeated rows.
  - Format Issues: Ensure dates, numbers, and text fields align with expected formats.

```
[5]: # Check for missing values
     print(data.isnull().sum())

     # Detect duplicates
     print(data.duplicated().sum())
```

```
tranDate      0
custName      0
cardNum       0
zipCode       0
tranAmount    0
dtype: int64
0
```

Step 3: Handling Missing Values

- Strategies:
  - Drop: Remove rows with essential missing data.
  - Impute: Fill with mean, median, or custom values for numerical data.
- Example:
  - "Imagine customer ages are missing — use the average to preserve trends in analysis."

[33]:
```
# Impute missing review ratings with median
data['Review Rating'].fillna(data['Review Rating'].median(), inplace=True)

# Drop rows with missing Customer ID
data.dropna(subset=['Customer ID'], inplace=True)
```

Dealing With Duplicates

- Causes:
  - Data entry errors, merging datasets.
- Solution:
  - Identify duplicates using Pandas.DataFrame.duplicated().
  - Drop duplicates while keeping the first or last occurrence.
- Example:
  - Customer orders with redundant IDs.

Step 4: Removing Duplicates

- Why?
  - Duplicates over-represent data and can skew results.
- How?
  - Use drop_duplicates().

```
[35]: # Remove duplicate rows
      data = data.drop_duplicates()
```

Standardizing Data Formats

- Focus Areas:
  - Dates: Convert to standard formats (e.g., YYYY-MM-DD).
  - Categorical Variables: Normalize text (e.g., Male/male/M).
  - Numerical Data: Ensure consistent units (e.g., USD vs. EUR).
- Tools:
  - Use Python libraries like Pandas or Excel for transformations.

Step 5: Standardizing Formats

- Ensure consistent formatting for:
- Dates: pd.to_datetime()
- Text: .str.lower() or .str.upper()

[38]:
```
# Standardize Season to uppercase
data['Season'] = data['Season'].str.upper()

# Standardize text to lowercase
data['Gender'] = data['Gender'].str.lower()

data.head()
```

[38]:

	Customer ID	Age	Gender	Item Purchased	Category	Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Payme Metho
0	CUST1000	51.25	female	Bag	Apparel	600.06	Phoenix	XL	White	SPRING	3.0	No	Standard	Yes	No	15.0	Cred Ca
1	CUST1001	19.30	female	Shoes	Apparel	458.98	Los Angeles	L	Blue	WINTER	1.2	Yes	Standard	Yes	No	49.0	Cred Ca
2	CUST1002	32.30	male	Shoes	Apparel	475.11	Phoenix	S	Green	WINTER	4.6	Yes	Express	Yes	Yes	25.0	Cas
3	CUST1003	29.61	male	Headphones	Books	168.08	Chicago	XL	Black	SPRING	3.7	No	Overnight	No	No	0.0	PayP
4	CUST1004	56.30	female	Smartphone	Accessories	660.31	Chicago	XL	Black	WINTER	4.6	No	Overnight	No	Yes	47.0	Cas

Let's take a break

CoGrammar

Data Manipulation Basics

- Transformations:
  - Normalize or scale data.
- Aggregations:
  - Group and summarize data.
- Combining Datasets:
  - Merge, join, or concatenate tables.

Transforming Data

- Why Transform Data?
  - To prepare it for analysis or machine learning models.
- Examples:
  - Min-max scaling for numerical features.
  - One-hot encoding for categorical variables.

Step 6: Transforming Data

- **Scaling Numerical Data:** Brings all values to a similar range.
- **Encoding Categorical Data:** Converts text categories into numbers.

[42]:
```
from sklearn.preprocessing import MinMaxScaler
# Scale Purchase Amount
data['ScaledAmount'] = MinMaxScaler().fit_transform(data[['Purchase Amount (USD)']])

# Encode Payment Method
data['PaymentMethodEncoded'] = data['Payment Method'].astype('category').cat.codes

data.head()
```

[42]:

Purchase Amount (USD)	Location	Size	Color	Season	Review Rating	Subscription Status	Shipping Type	Discount Applied	Promo Code Used	Previous Purchases	Payment Method	Frequency of Purchases	ScaledAmount	PaymentMethodEncoded
600.06	Phoenix	XL	White	SPRING	3.0	No	Standard	Yes	No	15.0	Credit Card	Quarterly	0.598170	1
458.98	Los Angeles	L	Blue	WINTER	1.2	Yes	Standard	Yes	No	49.0	Credit Card	Quarterly	0.456339	1
475.11	Phoenix	S	Green	WINTER	4.6	Yes	Express	Yes	Yes	25.0	Cash	Monthly	0.472555	0
168.08	Chicago	XL	Black	SPRING	3.7	No	Overnight	No	No	0.0	PayPal	Weekly	0.163889	3
660.31	Chicago	XL	Black	WINTER	4.6	No	Overnight	No	Yes	47.0	Cash	Monthly	0.658741	0

Aggregating Data

- Use Cases:
  - Summarizing sales by region.
  - Calculating average customer spend.
- Example:
  - Group data using Pandas' `groupby` function.

Combining Datasets

- Techniques:
  - Merging: Combine datasets with a common key.
  - Joining: SQL-like operations.
  - Concatenation: Stack datasets vertically or horizontally.
- Example:
  - Merge customer and transaction datasets.

Step 7: Aggregating Data

- **Why?**
  - Summarize data for trends and insights.
- **How?**
  - Use groupby() for grouping and summarizing.

```
[43]: # Group by Category and sum Purchase Amount
sales_summary = data.groupby('Category')['Purchase Amount (USD)'].sum().reset_index()
print(sales_summary)
```

```
      Category  Purchase Amount (USD)
0  Accessories              381195.33
1      Apparel              365914.94
2        Books              401722.59
3  Electronics              406972.11
4     Footwear              404123.90
```

CoGrammar

Step 7: Combining Data

- **Techniques:** Inner joins, outer joins, concatenation.
- **Example:** Merge shopping data with customer demographics.

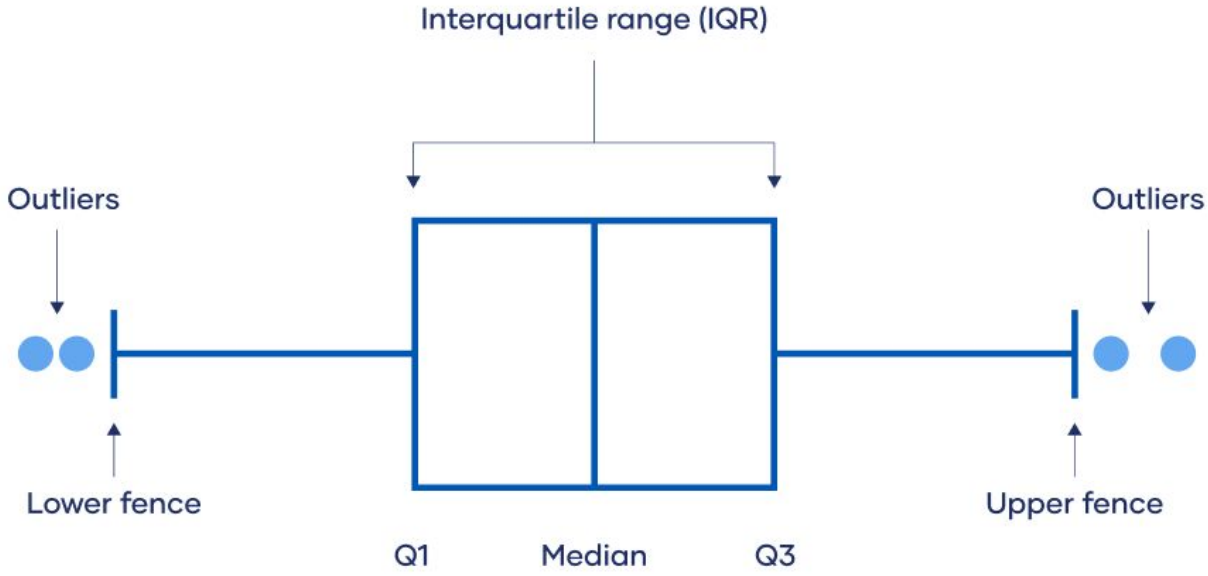
```
[60]: # Merge datasets on Customer ID
demographics = pd.read_csv('customer_demographics.csv')
combined_data = pd.merge(data, demographics, on='Customer ID', how='inner')
```

Outliers

- Outliers are data points that significantly deviate from the rest of the data distribution

CoGrammar

Detecting and Addressing Outliers



Scribbr

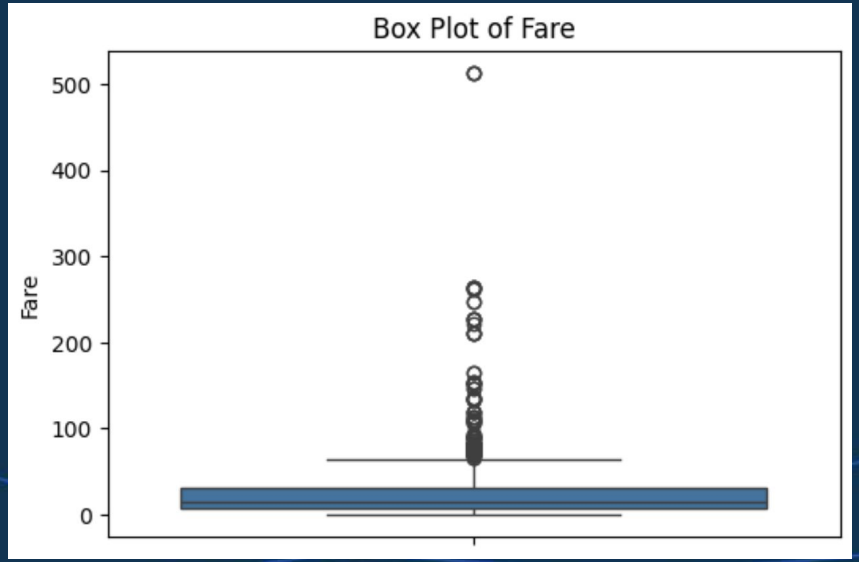
CoGrammar

Detecting and Addressing Outliers

- Methods:
  - Visualization: Use boxplots or scatterplots.
  - Statistical Measures: Z-scores, IQR.
  - Solutions: Treat or remove based on context.
- Example:
  - Sales data with an unusually high value.

Handling Outliers

Using a box plot to identify outliers.



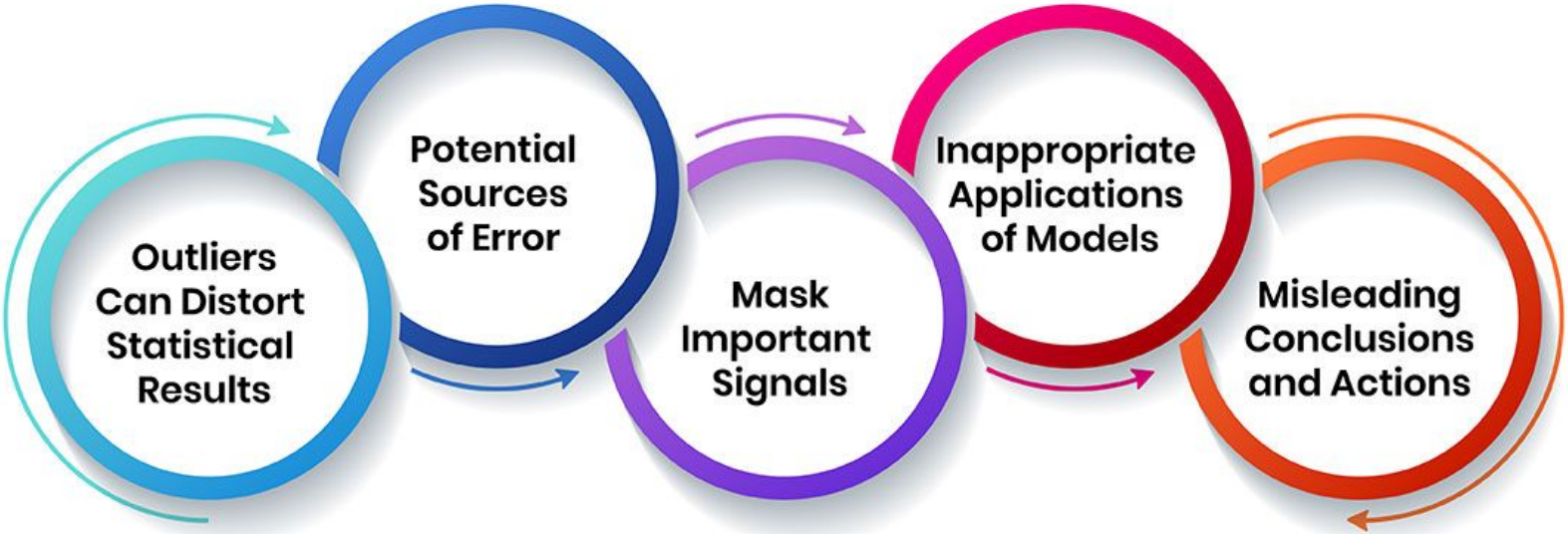
Handling Outliers

- Strategies for handling outliers:
  - **Removal:** Remove outliers from the dataset if they are erroneous or irrelevant to the analysis.
  - **Transformation:** Apply mathematical transformations (e.g., logarithmic, square root) to reduce the impact of outliers.
  - **Winsorisation:** Replace extreme values with the nearest non-outlier values.

```
data_winsorized['Salary'] = stats.mstats.winsorize(data_winsorized['Salary'], limits=0.2)
```

- Consider the context and domain knowledge when deciding how to handle outliers.

The Importance of Outlier Detection in Data Analysis



Best Practices for Data Cleaning

- Document Everything:
  - Track changes to data.
- Automate Repetitive Tasks:
  - Use scripts or pipelines.
- Validate Results:
  - Ensure changes improve data quality.
- Maintain Reproducibility:
  - Use version control for data.

Case Study: Cleaning a Dataset

- Dataset:
  - Sales data with missing values, duplicates, and inconsistent formats.
- Step-by-Step:
  - Identify and remove duplicates.
  - Impute missing sales amounts with the median.
  - Standardize date formats.
  - Handle outliers using IQR (Compute the interquartile range).

Real-world Challenges

- Unstructured Data:
  - Text, images, and logs.
- Large Datasets:
  - Memory constraints.
- Messy Data Sources:
  - Legacy systems or scraped data.
- **Tip**:
  - Always tailor cleaning to the context.

Recap Key Points

- Data cleaning ensures accurate and reliable insights.
- Techniques include handling missing data, duplicates, and outliers.
- Data manipulation prepares data for analysis and improves efficiency.

- "Your data is only as good as its quality!"

Further Learning

KDNuggets - Learn Data Cleaning and Preprocessing for Data Science with This Free eBook

Kaggle - Short Data Cleaning Course

Which method is best for handling missing data in small datasets?

A. Deleting rows with missing values
B. Imputing values with the dataset mean or median
C. Ignoring the missing data
D. Duplicating rows to fill gaps

What is the best way to detect outliers?

A. Sorting data manually
B. Using boxplots or statistical measures
C. Ignoring extreme values
D. Counting rows with missing values

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education