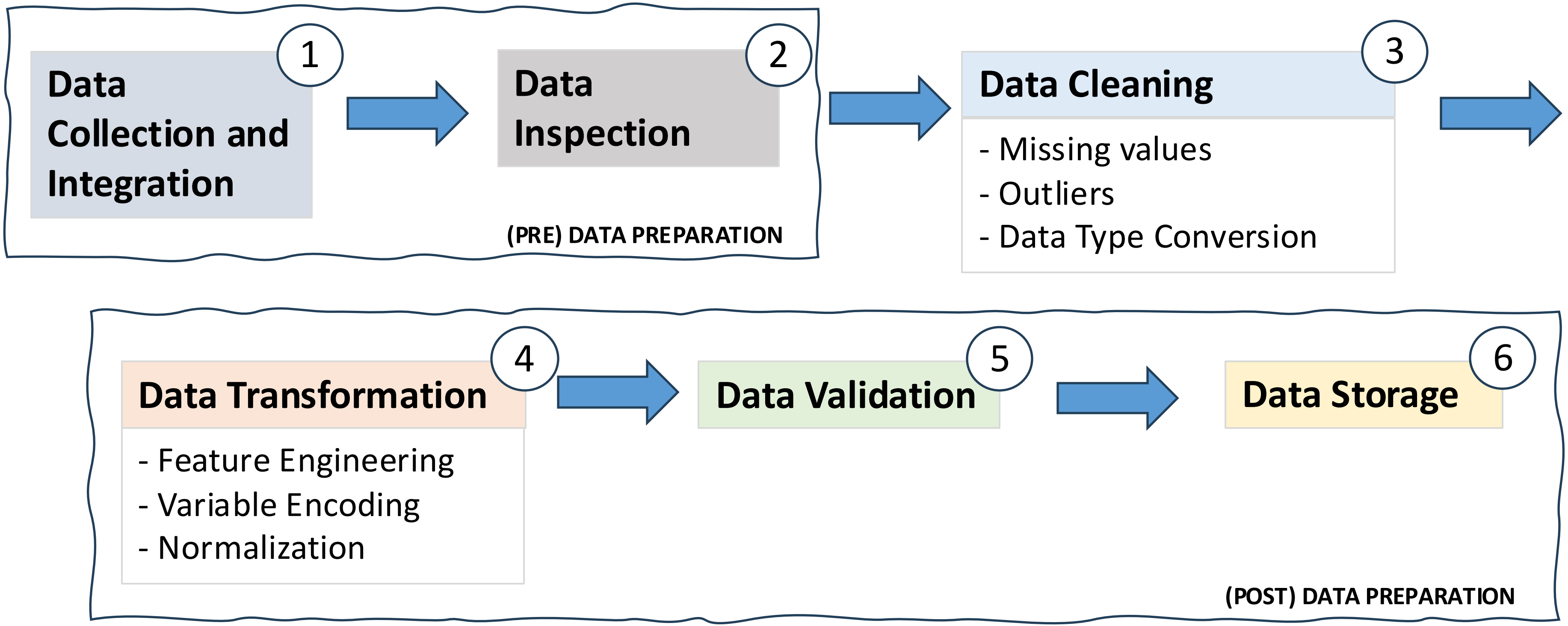
Quantitative Insights and Data Analytics II

SingStats

Data Processing: From Collection to Storage



The order of the stages can vary depending on the specific business problem and its structure

Data Inspection

The process of examining the collected data to understand its characteristics, quality, and potential issues before further processing.

KEY POINTS

- **Analyze data types and structures**
- **Identify missing or incomplete data**
- **Detect outliers and anomalies**
- **Assess data distribution and patterns**

EXAMPLE

A healthcare research team is analyzing patient data for a clinical trial. During data inspection, they:

- Discover that 5% of patient ages are missing
- Notice that some blood pressure readings are unrealistically high (e.g., systolic pressure > 300 mmHg)
- Find that gender is inconsistently recorded (e.g., "M", "Male", "1" all used)
- Observe an unexpected bimodal distribution in patient weight, suggesting potential data entry errors

Data Inspection: Understanding Your Dataset

KEY QUESTIONS

- What is the structure of the dataset?
- What does the data look like?
- Are there missing values?
- Are there duplicates?
- What are the summary statistics?
- What are the distributions of the variables?
- Are there any relationships between variables?
- Are there any inconsistencies or errors in the data?
- What is the context or domain knowledge?

EXAMPLE: Sales Transactions Dataset

Inspect transaction data for missing values, duplicates, consistency, outliers, and relationships.

Example: Inspection

- **df.head()**: Shows the first few rows of your dataset, giving you a quick look at the data structure and content.
- **df.info()**: Gives an overview of your DataFrame, including data types and missing values, helping you spot issues early.
- **df.describe()**: Provides summary statistics for numerical columns, showing key insights like averages and ranges.
- **df.isnull().sum()**: Tells you how many missing values are in each column, which is crucial for deciding how to handle them.
- **df.dtypes**: Shows the data types for each column, ensuring your data is in the right format for analysis.
- **df['Column'].unique()**: Lists all unique values in a column, helping you spot inconsistencies or prepare for data encoding.

```
import pandas as pd

# Load data into a DataFrame (e.g., from a CSV file)
df = pd.read_csv('data_inspection_test.csv')

# Display the first few rows of the DataFrame
print("First 5 rows of the data:")
print(df.head())

# Display summary information about the DataFrame
print("\nDataFrame Information:")
print(df.info())

# Display basic statistics for numerical columns
print("\nBasic Statistics:")
print(df.describe())

# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Check the data types of each column
print("\nData Types:")
print(df.dtypes)

# Display the unique values of a specific column (e.g., 'Category')
print("\nUnique values in 'Category' column:")
print(df['Category'].unique())
```

Data Cleaning

The process of identifying and correcting (or removing) inaccurate, incomplete, or irrelevant data from a dataset.

KEY POINTS

- **Handle missing values**
- **Remove duplicates**
- **Correct inconsistencies and errors**
- **Standardize data formats and units**

EXAMPLE

A marketing team is cleaning a customer database, They've identified and are addressing several data quality issues:

- **Missing Emails:** 1000 out of 100,000 records have missing email addresses.
- **Duplicate Records:** 500 duplicates exist due to variations in customer information.
- **Inconsistent Phone Numbers:** Phone numbers are formatted differently.
- **Typos in City Names:** Errors in city names are being corrected using a reference list.

Removing Duplicates

KEY QUESTIONS

- What constitutes a duplicate?
- Why do duplicates exist?
- How do duplicates impact the analysis?
- What is the context of the data?
- What information is lost if duplicates are removed?
- What are the business rules or domain-specific requirements?
- What are the implications for downstream processes?

EXAMPLE: Sales Transaction Dataset

Challenge: Duplicate transactions may exist due to system errors or manual data entry mistakes.

Solution: Use a unique identifier (e.g., transaction ID) to identify duplicates and remove them.

Handling Missing Values

KEY QUESTIONS

- What is the extent of missingness?
- What is the pattern of missingness?
- What is the reason for the missing data?
- How important is the column with missing data?
- How do missing values impact the analysis?
- What are the available options for handling missing values?
- What is the domain knowledge or business context?

EXAMPLE: Customer Churn Dataset

Challenge: Many customers have missing values for their last purchase date

Solution: Impute missing purchase dates with the maximum purchase date for that customer, assuming they are still active

Handling Outliers

KEY QUESTIONS

- What constitutes an outlier in your data?
- How do outliers impact your analysis or model?
- Are outliers due to data entry errors or legitimate variations?
- How should outliers be handled?
- What is the context or domain knowledge?

EXAMPLE: income Dataset

Challenge: A few individuals have extremely high incomes (e.g., millions of dollars) that could skew the analysis.

Solution: Use Z-score normalization to identify outliers and consider capping or removing them if they significantly impact the results.

Data Transformation

The process of converting data from its raw form into a format more suitable for analysis, modeling, or visualization.

KEY POINTS

- **Normalize or scale numerical data**
- **Encode categorical variables**
- **Create derived features**
- **Aggregate or summarize data**

EXAMPLE

A financial analyst is preparing data for a machine learning model to predict loan defaults

- **Normalization:** They standardize income and loan amounts to a common scale using z-score normalization
- **Encoding:** They convert job types into separate binary columns (one-hot encoding)
- **Feature Creation:** They calculate a new feature, "debt_to_income_ratio," to analyze financial health
- **Aggregation:** They summarize credit card transactions monthly for each customer.

Encoding Categorical Variables

KEY QUESTIONS

- What are the categorical variables in your dataset?
- Which encoding method is appropriate for each categorical variable?
- Are there any missing values in the categorical variables that need to be handled before encoding?
- How will the encoding affect the dataset size and model performance?

EXAMPLE: Sentiment Analysis

Challenge: A categorical variable like "sentiment" (positive, negative, neutral) needs to be encoded for classification

Solution: Use label encoding to assign numerical values to each sentiment category (e.g., positive=1, negative=-1, neutral=0)

Example: Transformation

- **Standardize Dates**: Convert date columns into a consistent datetime format for accurate time-based analysis.
- **Extract Date Components**: Break down dates into year and month to analyze time-based trends and patterns.
- **Create Calculated Columns**: Compute new metrics like profit from existing data to gain deeper insights.
- **Summarize Data**: Aggregate data by categories to produce summaries like total sales and revenue.

```
# Load the data
df = pd.read_csv(data_transformation_test.csv')

# 1. Clean the data
# Convert 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# 2. Transform date columns
# Extract year and month from the 'Date' column
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month

# 3. Create a new column based on existing data
# Calculate profit as Revenue - Sales * Unit Price (assuming unit price is $50)
df['Unit Price'] = 50 # Unit price assumption
df['Profit'] = df['Revenue'] - (df['Sales'] * df['Unit Price'])

# 4. Aggregate data
# Group by 'Product' and calculate total sales and total revenue
aggregated_df = df.groupby('Product').agg({
'Sales': 'sum',
'Revenue': 'sum',
'Profit': 'sum'
}).reset_index()
```

Data Validation

The process of ensuring the accuracy, consistency, and reliability of the processed data before it's used for analysis or modeling.

KEY POINTS

- **Verify data integrity and consistency**
- **Check for logical errors or inconsistencies**
- **Validate against business rules or constraints**
- **Perform cross-validation with external sources**

EXAMPLE

A government agency is validating census data:

- **ZIP Code Validation:** They ensure all ZIP codes are valid US postal codes.
- **Population Consistency:** They verify that population totals by age group match the overall population for each area.
- **Business Rule Enforcement:** They flag households with more than 20 members as potential exceptions.
- **Cross-Validation:** They compare population data with other sources to check for errors.

Example: Validation

- **Validate Email Addresses**: Ensures that email addresses follow a valid format, reducing the likelihood of errors when sending emails or processing user accounts.
- **Validate Age**: Confirms that age values are within a logical and acceptable range (e.g., between 0 and 120). This helps avoid data entry errors.
- **Check for Non-null Required Fields**: Ensures that essential fields like 'Name' and 'Email' are not missing. Missing values in required fields can lead to incomplete records and affect data quality.

```
# Load the data
df = pd.read_csv('data_validation_test.csv')

# 1. Validate Email Addresses
def is_valid_email(email):
# Simple regex for email validation
return re.match(r"[^@]+@[^@]+\.[^@]+", email) is not None

df['Email_Valid'] = df['Email'].apply(is_valid_email)

# 2. Validate Age
def is_valid_age(age):
# Ensure age is between 0 and 120
return pd.notna(age) and 0 <= age <= 120

df['Age_Valid'] = df['Age'].apply(is_valid_age)

# 3. Check for Non-null Required Fields
df['Name_Not_Null'] = df['Name'].notna()
df['Email_Not_Null'] = df['Email'].notna()
```

main.py

```python
x = 3

assert x == 3
print("x is equal to 3")
assert x == 5
print("Code finished!")
```

Run: main

```
C:\Users\ak111\PycharmProjects\pythonCourse\venv\Scripts\python.exe C:/Users/ak111/Pycha
x is equal to 3
Traceback (most recent call last):
  File "C:\Users\ak111\PycharmProjects\pythonCourse\main.py", line 5, in <module>
    assert x == 5
AssertionError

Process finished with exit code 1
```