Data Acquisition and Cleaning: Part I

July 7th, 2025

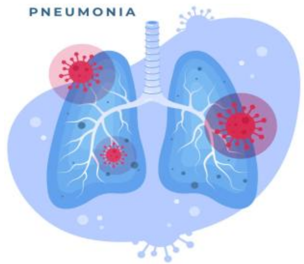
Introduction to Data Acquisition

- **Definition:** Data acquisition is the process of collecting, measuring, and storing data for analysis. It's the first step in any data-driven project.
- **Importance:** High-quality data is essential. Poor-quality input leads to misleading results ("garbage in, garbage out").

- **What makes data good?**

1. Accuracy: Reflects real-world truth
2. Completeness: No critical data is missing
3. Consistency: Logical coherence within and across datasets
4. Timeliness: Up-to-date for the context of analysis
5. Validity: Follows expected formats and value ranges
6. Uniqueness: No unjustified duplicates
7. Relevance: Answers the research or business question
8. Traceability: Clearly documented sources and processing steps

Example: The Hidden Danger: When ML Learns the Wrong Lesson



- A team of researchers at Columbia University developed a machine learning model to help decide whether patients with pneumonia should be hospitalized or sent home. They trained the model using real historical patient data.
- The model worked well — except for one critical mistake: it learned that asthma made pneumonia patients *less* likely to die and thus recommended sending them home.
- But in reality, **asthmatic patients were only surviving because they were immediately sent to intensive care** — not because their risk was low.
- Just because someone *survived* cancer due to early chemotherapy, doesn’t mean cancer isn’t dangerous.

An experiment

🔍 Step 2 – What's Wrong with the Model?

- Asthmatic patients were always sent to intensive care → low death rates
- The data learned: asthma = low risk → send home ❌
- It missed the reason: survival was due to ICU care
- The data confused correlation with causation
- This is a classic example of biased training data

✅ Step 4 – Fixing the Labels

- severity > 60 → high risk
  - age > 80 → high risk
  - asthma → high risk

- This simulates what would happen *without* ICU
- The data now correctly treats asthma as dangerous
- Key idea: Outcomes ≠ Risk — context matters!

Methods of Data Acquisition

1. Manual Collection
2. File-Based Acquisition
3. API Access (Programmatic)
4. Web Scraping
5. Database Queries
6. Sensor and IoT Streams
7. Simulation and Synthetic Data
8. Screen Scraping and OCR
9. Crowd-Sourced or Community-Contributed Platforms

📋 1. Manual Collection

- Manual data collection involves **direct, human-led methods** to record information — often used in field research or qualitative studies where automation is impractical.

Common Techniques:

- **Surveys and Questionnaires**
  - Paper-based or online (e.g., Google Forms, Qualtrics)
  - Used in social science, education, market research
- **Interviews and Focus Groups**
  - Verbal responses recorded as transcripts or notes
  - Rich, context-sensitive data; requires transcription
- **Manual Logs**
  - Recording sensor readings, observations, or timestamps by hand
  - Used in experiments or resource-limited settings

Hands-On Activity: Mini Manual Survey & Data Entry

- Step 1: Design Your Survey (3-5 short questions)
- Step 2: Collect Responses
- Step 3: Enter & Preview the Data

Discussion

- Did everyone interpret your questions the same way?
- Were there any invalid or unclear answers?
- What changes would improve data quality next time?

Take Away

- Inconsistencies, missing values, and subjectivity may arise
- Clear question wording and data validation

✅ Method 2: File-Based Acquisition

- File-based acquisition refers to gathering data stored in files — often in structured formats like CSV, Excel, or JSON — and loading it into software for analysis.

📁 **Common File Formats:**

- **CSV (Comma-Separated Values)** — Simple, universal, human-readable
- **Excel (.xlsx)** — Often used in business and government
- **JSON / XML** — Semi-structured, common in APIs and config files
- **TXT / Log files** — Often line-by-line records or unstructured data

Examples

- Load the dataset, understand its structure, and identify early quality issues
- **Files are the most common format** for sharing and storing small-to-medium datasets — especially in CSV and Excel formats.
- **Data quality is not guaranteed.** Files may contain typos, missing values, bad formats, or inconsistent column naming — even from “official” sources.
- **Always inspect before analysis.** Use .info(), .describe(), .head(), and .isnull().sum() to assess the dataset's structure and issues.
- **Documentation may be missing.** Files often lack metadata — you must infer structure or seek external documentation.
- **Cleaning is often necessary.** Standardizing column names, fixing types, and handling missing data are essential before analysis.
- **Small files are easy to misuse.** Copy-pasting from Excel or manual editing can break data integrity. Always script your processing steps for reproducibility.

✅ Method 3: API Access (Programmatic)

- APIs (Application Programming Interfaces) allow programs to access data on demand over the internet — often in **semi-structured formats like JSON or XML.**

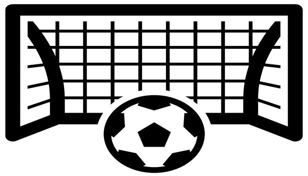
🔗 **Common API Types:**

- **REST APIs** (most common): Use HTTP verbs like GET, POST
- **GraphQL APIs**: Flexible querying, newer
- **Streaming APIs**: Push data in real time (e.g., Twitter stream)

🌐 **Real-World Examples:**

- 📈 **CoinDesk API** — cryptocurrency prices
- 🌥 **OpenWeatherMap API** — weather data
- 🚀 **NASA APIs** — missions, satellite imagery
- 🦠 **COVID-19 APIs** — Johns Hopkins, WHO

Examples



- Many APIs (like OpenWeatherMap or Google Maps) require you to register and use an API key.
- Requires API keys or authentication
- Rate limits: limited requests per minute/day
- May change without notice (versioning, fields)

✅ Method 4: Web Scraping

- Web scraping is the automated extraction of data from websites by parsing the HTML structure (the underlying code of web pages).

🧰 Tools Commonly Used:

- **requests** — to fetch raw HTML content
- **BeautifulSoup** — to parse and navigate HTML elements
- **Selenium** — for dynamic websites that require JavaScript rendering
- **lxml** — fast HTML/XML parser (used with BeautifulSoup)

Useful when no API is provided
Can extract structured or semi-structured data (e.g., product listings, headlines)
Flexible and scriptable

Example : Scraping Book Titles

Easy to scrape:

•Clean, consistent HTML structure (e.g., regular div or table patterns)
•Semantic HTML tags (<h1>, <article>, <ul>)
•Static content (loaded in the initial HTML, no JavaScript rendering required)
•No anti-scraping mechanisms (e.g., no CAPTCHAs or login walls)

Hard to scrape:

•Content loaded via JavaScript (requires Selenium or a headless browser)
•Frequent HTML changes or random class names
•Content hidden behind logins or infinite scroll
•Pages with CAPTCHAs or bot detection

✅ Method 5: Database Queries

- A **database** is a structured system for storing, managing, and retrieving data efficiently. You use **queries** (e.g., SQL) to extract subsets of interest.

🧰 **Common Types:**

- **Relational databases (SQL):** Tables with rows/columns; relationships enforced by keys
  👉 Tools: SQLite, PostgreSQL, MySQL
- **NoSQL databases:** Flexible schema (key-value, documents, etc.)
  👉 Tools: MongoDB, Firebase

✅ Method 6: Sensor and IoT Streams

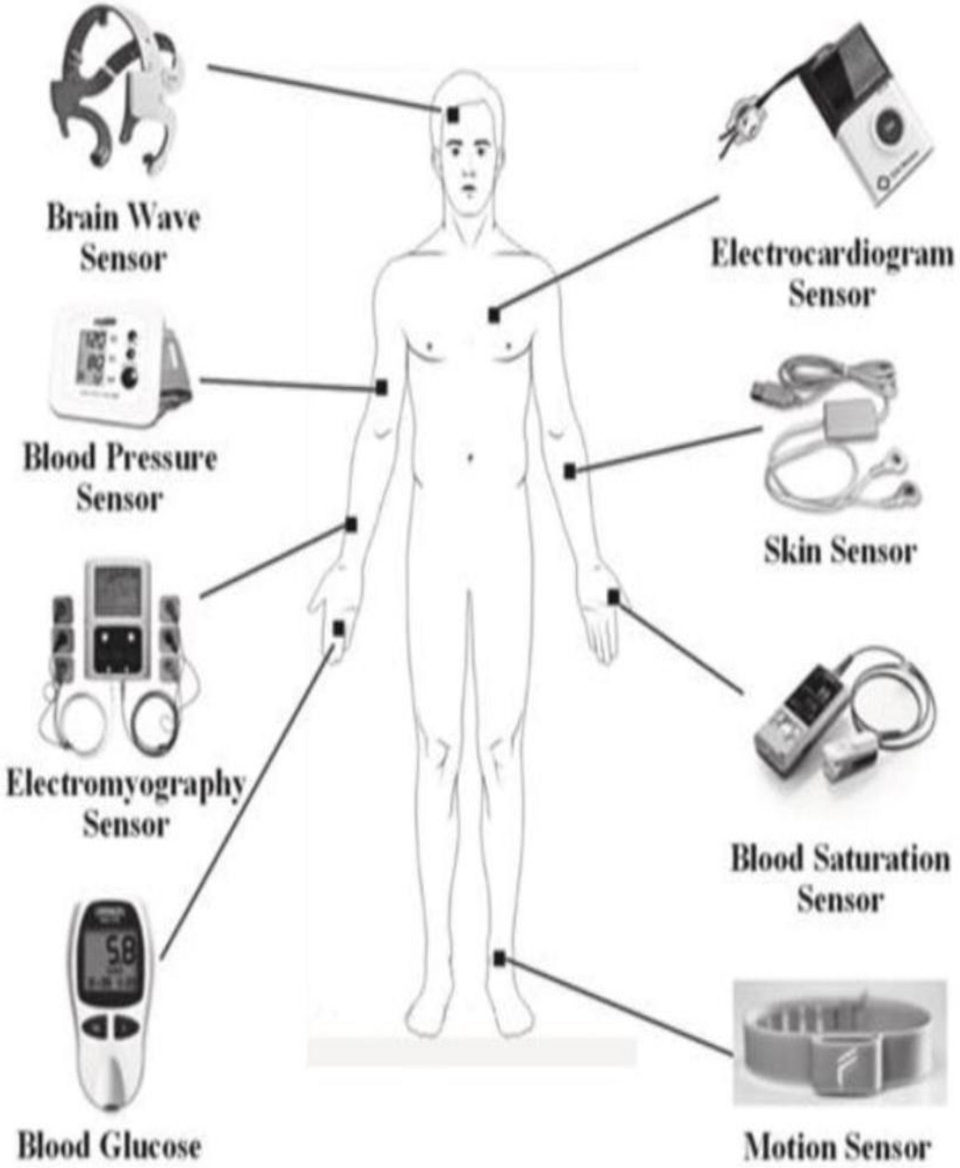
- **Sensor data acquisition** involves collecting real-time measurements from physical devices such as thermometers, GPS modules, cameras, or heart rate monitors. These streams are often continuous and need to be handled live or buffered.

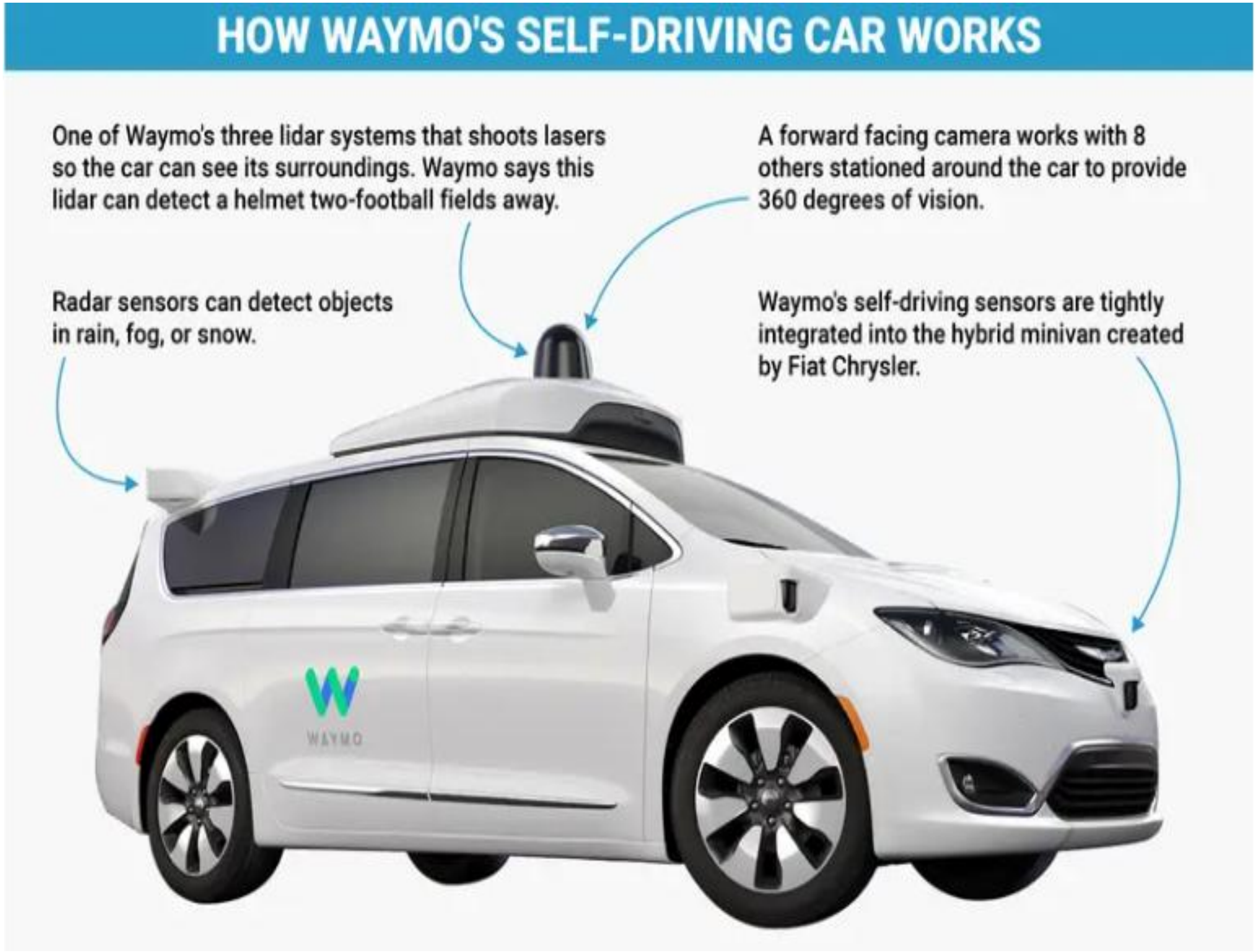
📡 **Common Sensor Types:**

- **Environmental:** temperature, humidity, air quality
- **Motion:** accelerometer, gyroscope
- **Biometric:** heart rate, EEG, pulse oximeter
- **Location:** GPS (latitude, longitude)
- **Industrial:** pressure, vibration, voltage, light

🔌 **Common IoT Protocols:**

- **MQTT, HTTP, WebSocket, Bluetooth**
- **Serial or USB** connections for local sensor kits (e.g., Arduino)
- Often sends JSON, CSV, or binary packets


Brain Wave Sensor
Electrocardiogram Sensor
Blood Pressure Sensor
Skin Sensor
Electromyography Sensor
Blood Saturation Sensor
Blood Glucose
Motion Sensor


HOW WAYMO'S SELF-DRIVING CAR WORKS
One of Waymo's three lidar systems that shoots lasers so the car can see its surroundings. Waymo says this lidar can detect a helmet two-football fields away.
A forward facing camera works with 8 others stationed around the car to provide 360 degrees of vision.
Radar sensors can detect objects in rain, fog, or snow.
Waymo's self-driving sensors are tightly integrated into the hybrid minivan created by Fiat Chrysler.
WAYMO

✅ Method 7: Simulation and Synthetic Data

- **Synthetic data** is data that is **artificially generated** rather than collected from real-world events. It mimics the structure and patterns of real data without containing real individual records.

🧪 **How It's Created:**

•Using **random generators** (e.g., numpy, random)

•**Simulation models** based on assumptions (e.g., disease spread, market behavior)

•**Libraries like Faker** to generate realistic-looking personal data (names, dates, emails)

•Advanced: **GANs** or ML-based tabular synthesis (e.g., SDV)

Example: Generate a Synthetic Health Dataset

- We'll simulate a dataset of patients with height, weight, age, and disease status.

Discussion:

- How realistic is this dataset? What’s missing?
- What risks would occur if someone used this as real medical data?
- How could we make this more realistic (e.g., add noise, missingness, or comorbidities)?

Example: Digit generation

- We’ll use the built-in **digits dataset** from sklearn.datasets
- Apply simple **image transformations** (e.g., noise, rotation) to simulate new examples
- Basic form of image-based synthetic data generation
- Increase samples for further study

✅ Method 8: Screen Scraping and OCR

- **Screen scraping** involves extracting data from content **rendered visually on screen**, including scanned documents, PDFs, screenshots, or old legacy systems. **OCR (Optical Character Recognition)** turns text in images into machine-readable strings.

🧰 **Tools Commonly Used:**

- **Tesseract OCR** (pytesseract) — free OCR engine by Google
- **OpenCV** — for preprocessing (e.g., thresholding, resizing)
- **PDF-to-image** libraries — e.g., pdf2image, PyMuPDF
- **Adobe OCR API**, **AWS Textract**, or **Google Vision API** — cloud-based options

🔍 **Real-World Use Cases:**

- Converting scanned PDFs to searchable text
- Processing receipts for expense reports
- Extracting names/dates from lab reports or invoices
- Forensic science

✅ Method 9: Crowd-Sourced or Community-Contributed Platforms

- These platforms collect data from **the public** or a **community of volunteers**, often across many domains. The data is openly shared for analysis, research, or collaboration.

Platform	Description	Example Use Cases
Wikipedia	Encyclopedia content from volunteers	Text, links, metadata
OpenStreetMap	Global map data created by the public	Routing, GIS, infrastructure studies
StackExchange	Forum Q&A data across disciplines	NLP, education, tech behavior
Kaggle Datasets	Shared by users for analysis challenges	ML, visualization, public projects
GitHub Repos	Open datasets and code by community	Reproducibility, audit trails

Practical consideration for Big Data

Visualization and Summarization of Big Data

- You **cannot plot millions of rows directly**. Big data visualization is about **aggregating, sampling, filtering**, and choosing the **right tool** for scale.

📉 Why Visualization Fails at Scale

•Interactive plotting libraries (like matplotlib, seaborn) choke on millions of rows

•Browser-based dashboards crash or freeze

•Summary statistics may become **misleading** due to skew or missing values

Try it......

- What Happens When You Try to Plot 10 Million Rows

⚽

- 🧠 **What Happens:**
- **Freezing:** Notebook or IDE becomes unresponsive
- **Memory Error:** Kernel crashes if RAM is insufficient
- **Browser Timeouts:** If done in a dashboard app

Practical method

- **Sampling**

  Use **random sampling** to show a representative subset

  Can stratify by category to preserve class balance

  Uniformly sampling
- **Aggregation**

  Group by key dimensions before plotting

  Use time buckets, geographic bins, or user cohorts

- Example: **Hourly traffic volume** for I-94 near Minneapolis–Saint Paul (2012–2018), with weather and holiday features. It’s 48,204 rows