

# Class 2: Python ML Project

Master Course:  
Data-driven Systems Engineering (ML Operations)  
440MI and 305SM

# Agenda & Learning Goals

- Why Python dominates ML
- Alternative ML languages
- ML pipeline & KDD
- Data quality & EDA
- PCA & feature engineering
- From prototype → pickle → service
- Real-world ML ecosystem

# Python in Machine Learning (Overview)

- General-purpose, interpreted, dynamic language
- Rich ecosystem: NumPy, pandas, scikit-learn, TensorFlow, PyTorch
- Huge open-source community support



[Reference: Van Rossum, G. Python Programming Language, 1991]

# Why Python?

- Readability and simplicity
- Massive number of ML libraries
- Community-driven innovation
- Great integration with visualization tools

[Reference: Müller & Guido, Introduction to Machine Learning with Python, 2016]

# Advantages of Python for ML

- Cross-platform and open-source
- High productivity (less boilerplate than Java/C++)
- Easy prototyping and research → production

# Advantages of Python for ML

## API design for machine learning software: experiences from the scikit-learn project

Lars Buitinck<sup>1</sup>, Gilles Louppe<sup>2</sup>, Mathieu Blondel<sup>3</sup>, Fabian Pedregosa<sup>4</sup>,  
Andreas C. Müller<sup>5</sup>, Olivier Grisel<sup>6</sup>, Vlad Niculae<sup>7</sup>, Peter Prettenhofer<sup>8</sup>,  
Alexandre Gramfort<sup>4,9</sup>, Jaques Grobler<sup>4</sup>, Robert Layton<sup>10</sup>, Jake Vanderplas<sup>11</sup>,  
Arnaud Joly<sup>2</sup>, Brian Holt<sup>12</sup>, and Gaël Varoquaux<sup>4</sup>

<sup>1</sup> ILPS, Informatics Institute, University of Amsterdam

<sup>2</sup> University of Liège

<sup>3</sup> Kobe University

<sup>4</sup> Parietal, INRIA Saclay

<sup>5</sup> University of Bonn

<sup>6</sup> Independent consultant

<sup>7</sup> University of Bucharest

<sup>8</sup> Ciuvo GmbH

<sup>9</sup> Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI

<sup>10</sup> University of Ballarat

<sup>11</sup> University of Washington

<sup>12</sup> Samsung Electronics Research Institute

## 2 Core API

All objects within *scikit-learn* share a uniform common basic API consisting of three complementary interfaces: an *estimator* interface for building and fitting models, a *predictor* interface for making predictions and a *transformer* interface for converting data. In this section, we describe these three interfaces, after reviewing our general principles and data representation choices.

### 2.1 General principles

As much as possible, our design choices have been guided so as to avoid the proliferation of framework code. We try to adopt simple conventions and to limit to a minimum the number of methods an object must implement. The API is designed to adhere to the following broad principles:

**Consistency.** All objects (basic or composite) share a consistent interface composed of a limited set of methods. This interface is documented in a consistent manner for all objects.

**Inspection.** Constructor parameters and parameter values determined by learning algorithms are stored and exposed as public attributes.

**Non-proliferation of classes.** Learning algorithms are the only objects to be represented using custom classes. Datasets are represented as NumPy arrays or SciPy sparse matrices. Hyper-parameter names and values are represented as standard Python strings or numbers whenever possible. This keeps *scikit-learn* easy to use and easy to combine with other libraries.


**Composition.** Many machine learning tasks are expressible as sequences or combinations of transformations to data. Some learning algorithms are also naturally viewed as meta-algorithms parametrized on other algorithms. Whenever feasible, such algorithms are implemented and composed from existing building blocks.

**Sensible defaults.** Whenever an operation requires a user-defined parameter, an appropriate default value is defined by the library. The default value should cause the operation to be performed in a sensible way (giving a baseline solution for the task at hand).

[Reference: Buitinck et al.,  
API design for machine  
learning software, 2013]

# Limitations of Python

- Slower runtime (interpreted, GIL issue)
- Memory-intensive for very large datasets
- Deployment challenges in high-performance environments



The Python Global Interpreter Lock or **GIL**, in simple words, is a mutex (or a lock) that allows only one thread to hold the control of the Python interpreter.

This means that only one thread can be in a state of execution at any point in time. The impact of the GIL isn't visible to developers who execute single-threaded programs, but it can be a performance bottleneck in CPU-bound and multi-threaded code.

## Comparison Table

- Python: easy, fast prototyping, wide libraries
- Java: robust, enterprise-ready, but verbose
- R: stats-focused, less scalable
- Julia: fast, new ecosystem
- C++: powerful, steep learning curve



## Most popular technologies (Stackoverflow 2025)

### Programming, scripting, and markup languages

After more than a decade of steady growth, Python's adoption has accelerated significantly. It saw a 7 percentage point increase from 2024 to 2025; this speaks to its ability to be the go-to language for AI, data science, and back-end development.

2 Which programming, scripting, and markup languages have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the language and want to continue to do so, please check both boxes in that row.)

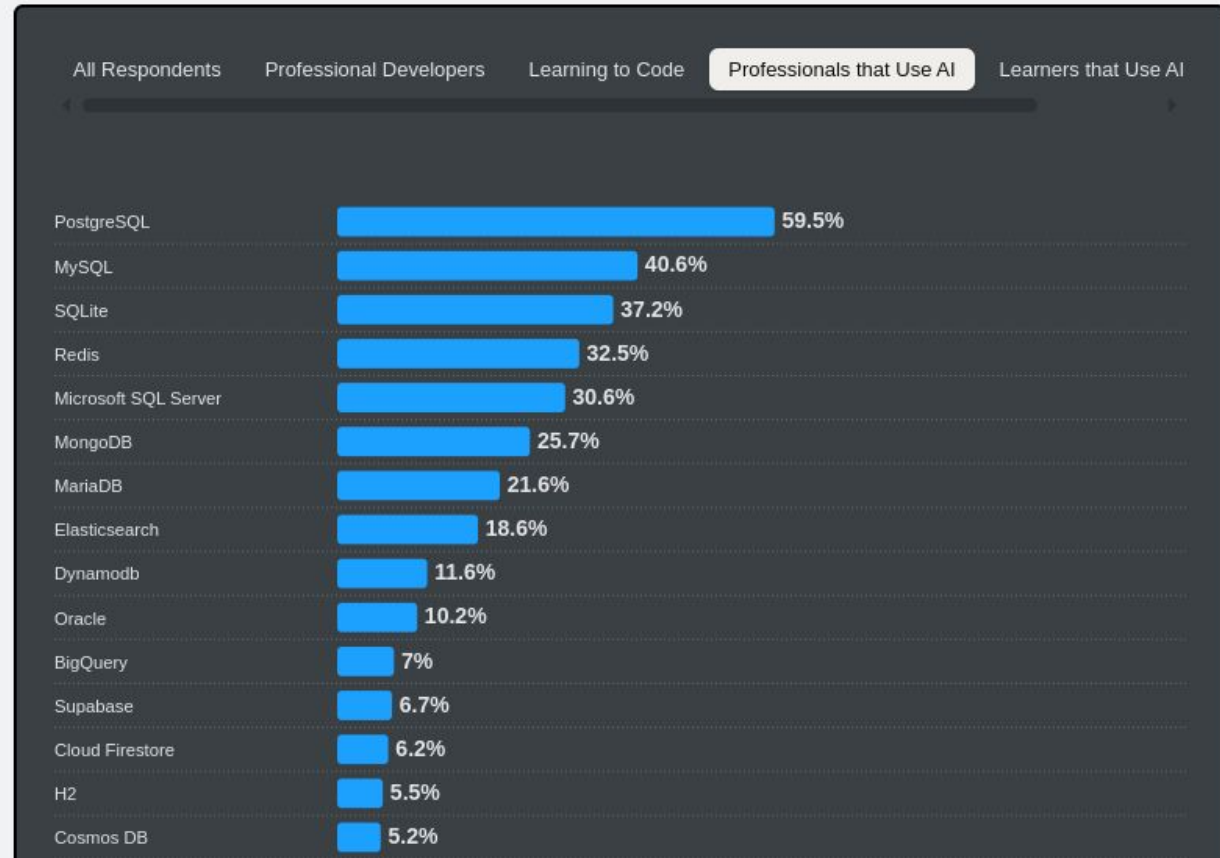


## Most popular technologies (Stackoverflow 2025)

### Databases

The significant growth in usage for Redis (+8%) highlights its growing importance. As applications become more complex, the need for high-speed, in-memory caching and data structures has made Redis an essential part of the modern tech stack.

? Which **database environments** have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the database and want to continue to do so, please check both boxes in that row.)



## Most popular technologies (Stackoverflow 2025)

### Cloud development

Docker has moved from a popular tool to a near-universal one. After years of growth, it experienced a +17 point jump in usage from 2024 to 2025, the largest single-year increase of any technology surveyed. This is partially due to consolidating some technology sectors in this year's survey.

? Which cloud platforms, containerization/orchestration tools, package managers, build tools, and infrastructure as code solutions have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the platform and want to continue to do so, please check both boxes in that row.)



## Most popular technologies (Stackoverflow 2025)

### Web frameworks and technologies

The +5 point increase for FastAPI is one of the most significant shifts in the web framework space. This signals a strong trend towards using Python for building performant APIs and reflects the overall strength of the Python ecosystem.

? Which web frameworks and web technologies have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the framework and want to continue to do so, please check both boxes in that row.)



## Most popular technologies (Stackoverflow 2025)

### Dev IDEs

Subscription-based, AI-enabled IDEs weren't able to topple the dominance of Visual Studio and Visual Studio Code this year. Both maintained their top spots for the fourth year while relying on extensions as optional, paid AI services.

? Which development environments and AI-enabled code editing tools did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.



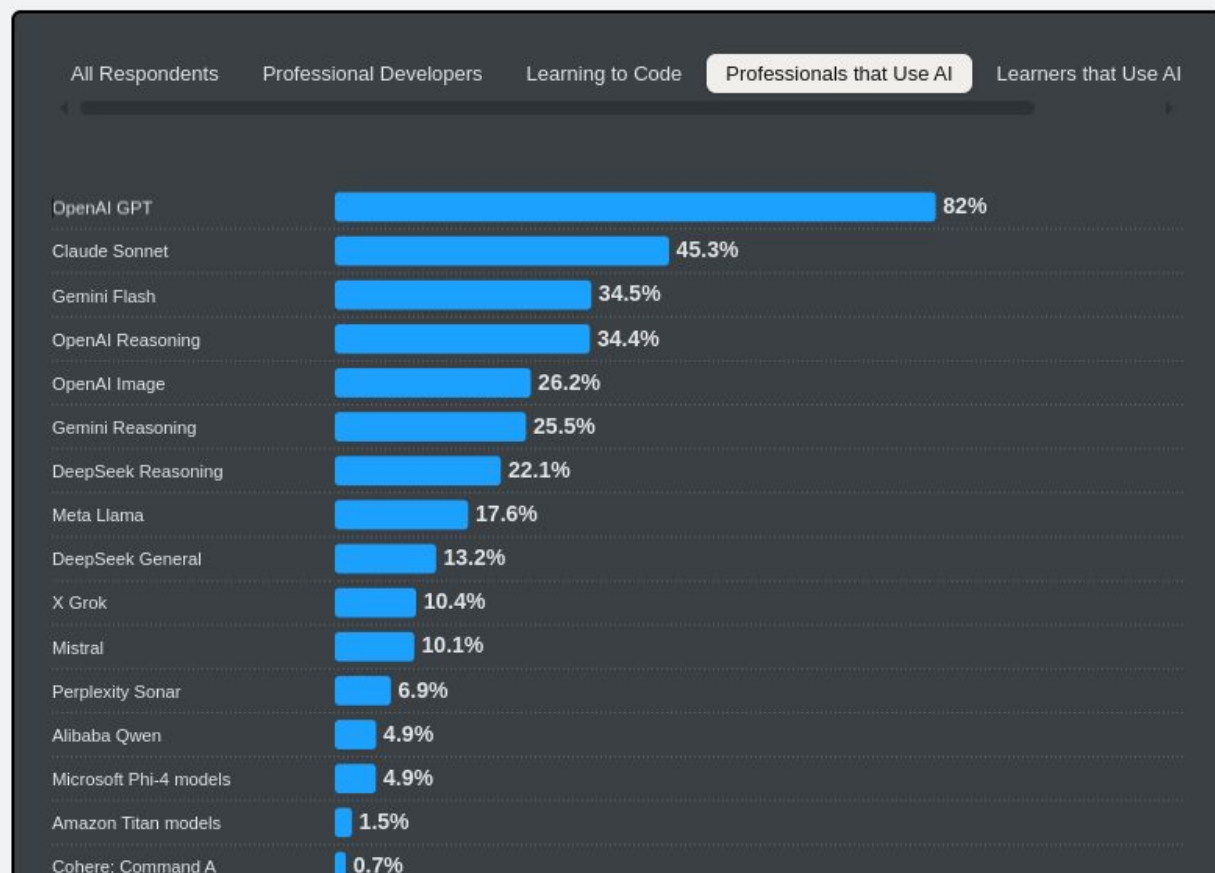


## Most popular technologies (Stackoverflow 2025)

### Large language models

OpenAI's GPT models top the large language model list with 82% of developers indicating they used them for development work in the past year. Anthropic's Claude Sonnet models are used more by professional developers (45%) than by those learning to code (30%).

? Which LLM models for AI tools have you used for development work in the past year, and which would you like to use next year? Select all that apply.



## Most popular technologies (Stackoverflow 2025)

### Code documentation and collaboration tools

Professional developers that use AI compared to all professional developers show a slight preference for Miro (17% vs. 16%) and Notion (19% vs. 17%) - could this be due to the AI integrations both tools have recently incorporated?

? Which collaborative work management and/or code documentation tools did you use regularly over the past year, and which do you want to work with over the next year? Select all that apply



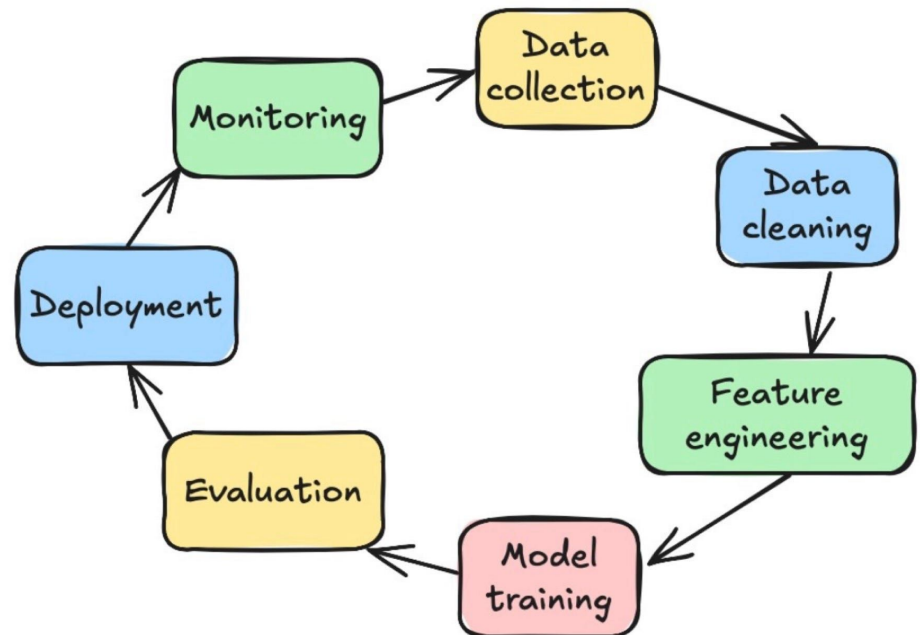
# What is a Machine Learning Pipeline?

- End-to-end process from  
data ingestion → model training → deployment
- Automates repetitive tasks
- Ensures reproducibility



# Typical Pipeline Steps

- Data collection
- Preprocessing (cleaning, transformation)
- Feature engineering
- Model training
- Evaluation
- Deployment
- Monitoring!?

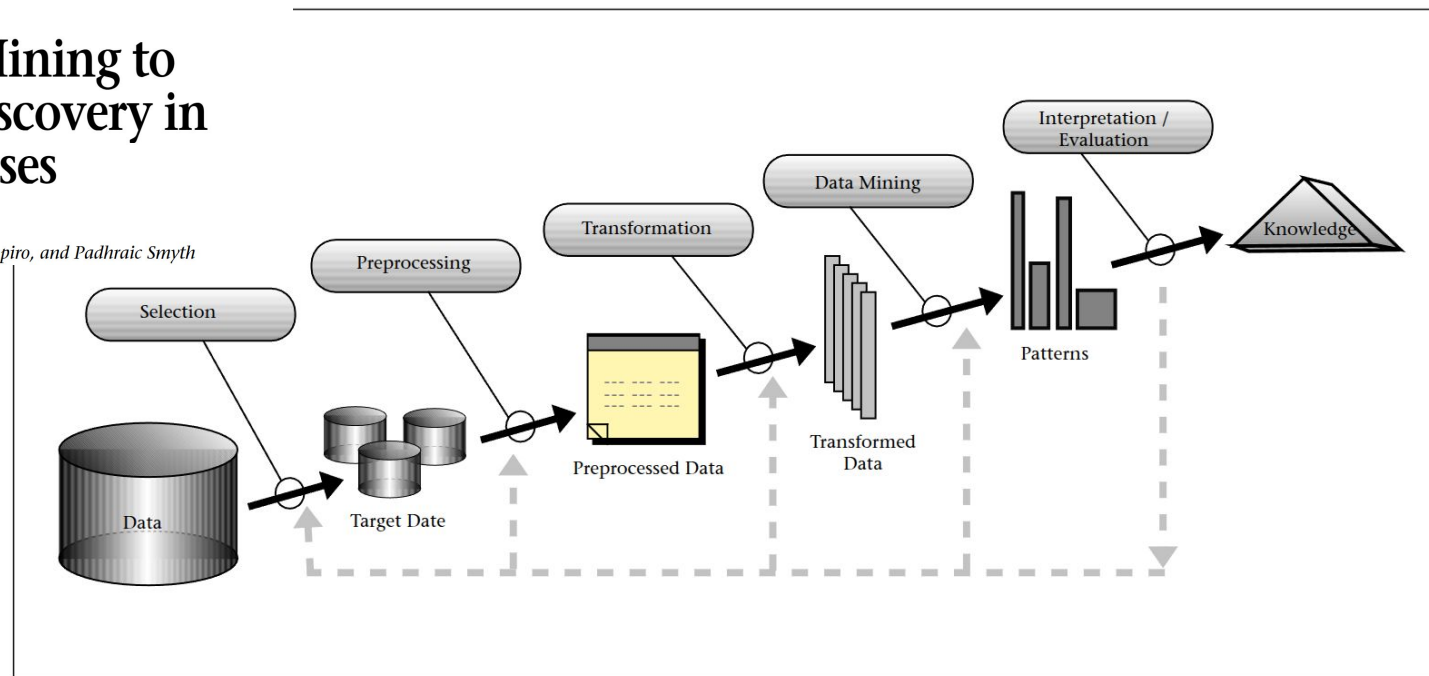


# KDD (Knowledge Discovery in Databases)

- Process of extracting useful knowledge from large datasets

## From Data Mining to Knowledge Discovery in Databases

*Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth*



*Figure 1. An Overview of the Steps That Compose the KDD Process.*

[Reference: Fayyad et al., 1996]

## KDD vs. CRISP-DM

- KDD: emphasizes data-driven discovery
- CRISP-DM: business understanding first, structured methodology
- Both widely used frameworks

[Reference: Wirth & Hipp, 2000]

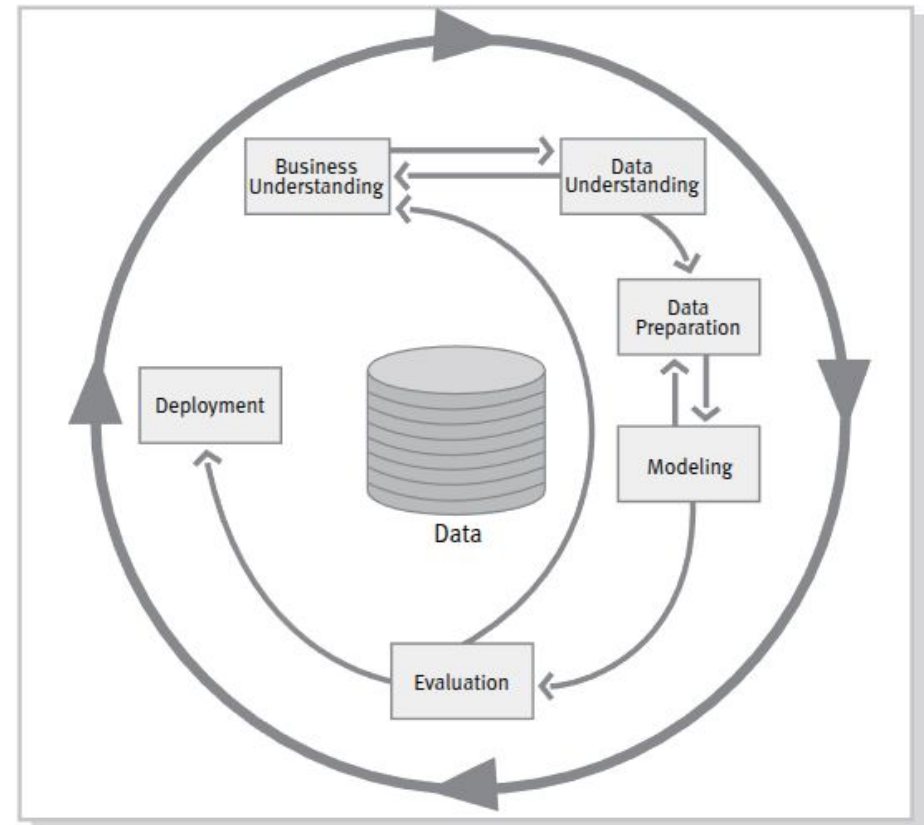


Figure 2: Phases of the CRISP-DM reference model

# CRISP-DM

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
<b>Determine Business Objectives</b> <i>Background</i> <i>Business Objectives</i> <i>Business Success Criteria</i>	<b>Collect Initial Data</b> <i>Initial Data Collection Report</i>	<b>Select Data</b> <i>Rationale for Inclusion/Exclusion</i>	<b>Select Modeling Techniques</b> <i>Modeling Technique</i> <i>Modeling Assumptions</i>	<b>Evaluate Results</b> <i>Assessment of Data Mining Results w.r.t. Business Success Criteria</i> <i>Approved Models</i>	<b>Plan Deployment</b> <i>Deployment Plan</i>
<b>Assess Situation</b> <i>Inventory of Resources</i> <i>Requirements, Assumptions, and Constraints</i> <i>Risks and Contingencies</i> <i>Terminology</i> <i>Costs and Benefits</i>	<b>Describe Data</b> <i>Data Description Report</i>	<b>Clean Data</b> <i>Data Cleaning Report</i>	<b>Generate Test Design</b> <i>Test Design</i>	<b>Review Process</b> <i>Review of Process</i>	<b>Plan Monitoring and Maintenance</b> <i>Monitoring and Maintenance Plan</i>
<b>Determine Data Mining Goals</b> <i>Data Mining Goals</i> <i>Data Mining Success Criteria</i>	<b>Explore Data</b> <i>Data Exploration Report</i>	<b>Construct Data</b> <i>Derived Attributes</i> <i>Generated Records</i>	<b>Build Model</b> <i>Parameter Settings</i> <i>Models</i> <i>Model Descriptions</i>	<b>Determine Next Steps</b> <i>List of Possible Actions</i> <i>Decision</i>	<b>Produce Final Report</b> <i>Final Report</i> <i>Final Presentation</i>
<b>Produce Project Plan</b> <i>Project Plan</i> <i>Initial Assessment of Tools and Techniques</i>	<b>Verify Data Quality</b> <i>Data Quality Report</i>	<b>Integrate Data</b> <i>Merged Data</i>	<b>Assess Model</b> <i>Model Assessment</i> <i>Revised Parameter Settings</i>		<b>Review Project</b> <i>Experience</i> <i>Documentation</i>
		<b>Format Data</b> <i>Reformatted Data</i>			
		<i>Dataset</i> <i>Dataset Description</i>			

Figure 3: Generic tasks (bold) and outputs (italic) of the CRISP-DM reference model

# Role of Datasets

- Foundation of ML systems
- Types:
  - structured (tables);
  - unstructured (images, text, audio).
- Sources:
  - databases;
  - sensors;
  - APIs.

# Data Challenges

- Missing values
- Data bias
- Noise and inconsistency
- Imbalanced datasets

# What is Data Quality?

- **Accuracy (no model accuracy!!!)**
  - How closely the data values reflect the real-world entities or events they represent.
  - Example: Customer age recorded as 29 when the actual age is 29 (accurate) vs. 92 (inaccurate).
- **Completeness**
  - The degree to which all required data is present.
  - Example: If a dataset of customers has missing phone numbers for 20% of entries, it lacks completeness.
- **Consistency**
  - Ensuring that data is the same across different sources or systems.
  - Example: A customer's address is recorded as "123 Main St." in one database and "123 Main Street" in another → inconsistent.
- **Timeliness**
  - How up-to-date the data is and whether it is available when needed.
  - Example: A stock price dataset must be updated in real-time; using yesterday's prices reduces timeliness.



# Why Data Quality Matters

- Garbage in → garbage out
- Poor quality → poor models
- Data quality is 70–80% of ML effort

Example:

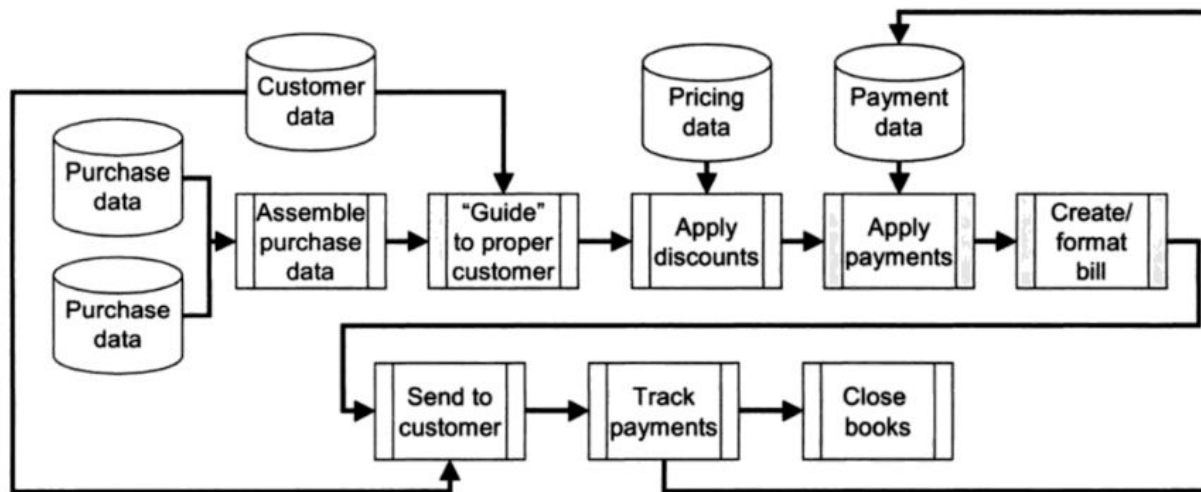


Figure 3.3 *A billing chain involves many hand-offs*

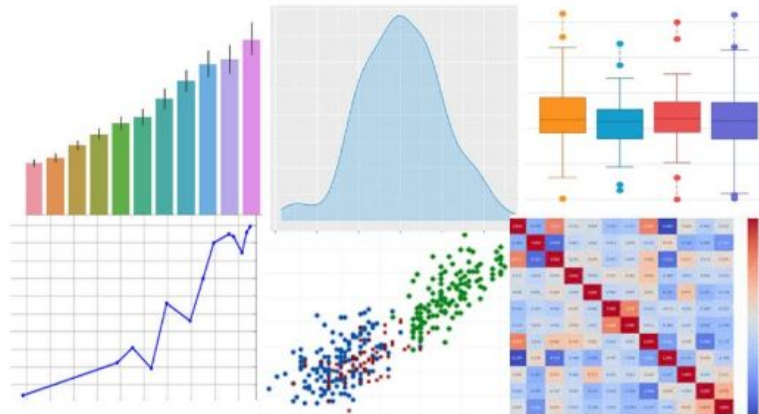
[Reference: Redman, T.C. Data Quality, 1996]



# EDA (Exploratory Data Analysis)

- Summarize dataset characteristics
- Detect anomalies, patterns, relationships
- Visual + statistical methods

20% of  
**EDA**  
Plots Data  
Scientists  
use 80% of  
the time



# EDA Techniques

- Descriptive statistics
- Correlation analysis
- Outlier detection
- Data visualization

## Tools for EDA

- Python libraries: pandas, matplotlib, seaborn, plotly
- Automated profiling:
  - pandas-profiling
  - Sweetviz

# Example EDA Workflow

- Load dataset
- Check missing values
- Visualize distributions
- Detect correlations
- Generate first hypotheses

# Summary & Key Takeaways

- Python dominates ML due to ecosystem & simplicity
- Pipelines & KDD structure workflows
- Data quality