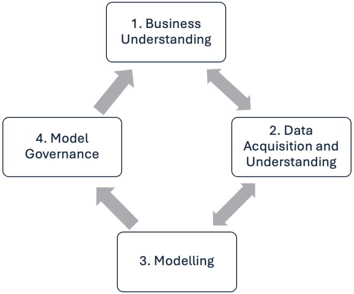
# Applied Machine Learning

# The Data Science Lifecycle

# Main phases



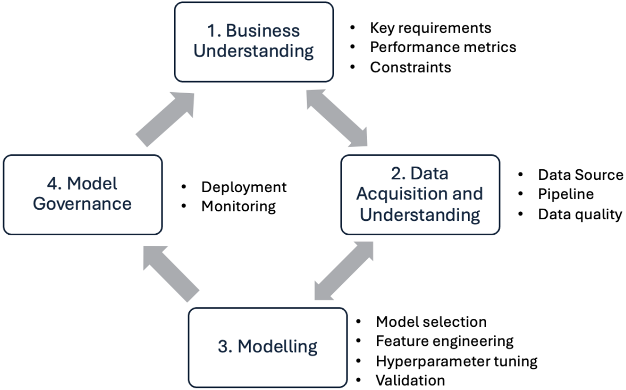

**Learning goals**

- Know phases of data science lifecycle
- Identify essential concepts, tools, and methods for each phase

# THE DATA SCIENCE LIFECYCLE

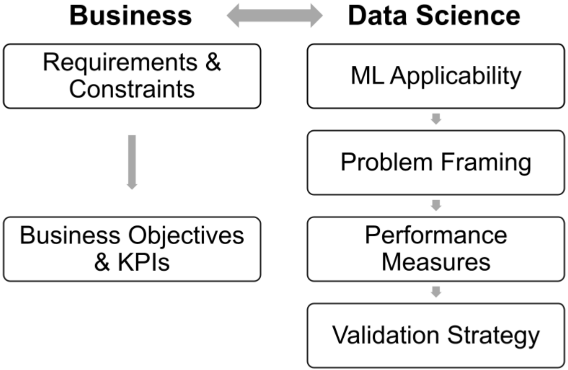
▸ "CRISP-DM" n.d.

**Lifecycle Management**: Process of managing a (product) lifecycle from inception, through engineering, design and manufacturing to deployment and eventual disposal.
**CRISP-DM (Cross-Industry Standard Process for Data Mining)**: A widely used framework that formalizes the iterative process of turning data into knowledge.



# 1. Business Understanding

# OVERVIEW BUSINESS UNDERSTANDING



# REQUIREMENTS & CONSTRAINTS

**Key question**: What is the end goal of the project?

- Define the intended **output**:
  - Report, dashboard, or interactive tool?
  - Scalable pipeline for production use?

# REQUIREMENTS & CONSTRAINTS

**Key question**: What is the end goal of the project?

- Define the intended **output**:
  - Report, dashboard, or interactive tool?
  - Scalable pipeline for production use?
- Identify constraints
  - Ethical: Avoid sensitive data use (e.g., address may proxy ethnicity)
  - Legal: Exclude protected attributes (e.g., gender in automated hiring)
  - Reproducibility: Ensure the analysis is fully reproducible in a single step
  - Explainability: All model decisions must be explainable to the user
  - Prediction Latency: Real-time predictions required
  - Model Size: Stay within specified memory constraints

# REQUIREMENTS & CONSTRAINTS

**1. ML Applicability: Is ML justified?**

- Explore non-ML baselines first (rules, heuristics, simple analytics)
- Use ML only if it adds clear value over simpler/existing alternatives

# REQUIREMENTS & CONSTRAINTS

1. **ML Applicability: Is ML justified?**
   - Explore non-ML baselines first (rules, heuristics, simple analytics)
   - Use ML only if it adds clear value over simpler/existing alternatives
2. **Problem Framing: How to frame the task?**
   - *Imbalanced data:* classification vs. anomaly detection?
   - *Pred. maintenance:* failure probability (classif.) vs. remaining lifetime (regr.)?
   - Select the ML task that aligns with your KPI and data characteristics

Classification

Regression

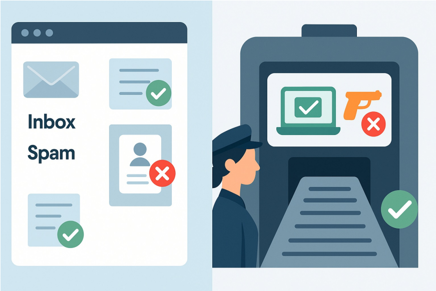
Clustering

Anomaly Detection

Dimensionality Reduction

# BUSINESS OBJECTIVES: PERFORMANCE MEASURES

- **How to translate key performance indicators (KPIs) to metrics?**
  - **Spam detection**: Job application flagged as spam = unacceptable
    $\Rightarrow$ Occasional inbox spam is fine $\Rightarrow$ minimize FP $\Rightarrow$ maximize **precision**
  - **Airport security**: Weapon flagged as laptop = unacceptable
    $\Rightarrow$ Occasional false alarm is fine $\Rightarrow$ minimize FN $\Rightarrow$ maximize **recall**



# BUSINESS OBJECTIVES: PERFORMANCE MEASURES

- **How to translate key performance indicators (KPIs) to metrics?**
  - **Spam detection**: Job application flagged as spam = unacceptable
    $\Rightarrow$ Occasional inbox spam is fine $\Rightarrow$ minimize FP $\Rightarrow$ maximize **precision**
  - **Airport security**: Weapon flagged as laptop = unacceptable
    $\Rightarrow$ Occasional false alarm is fine $\Rightarrow$ minimize FN $\Rightarrow$ maximize **recall**
- **Multiple objectives:** What if improving one metric worsens another?
  - *Scalarization*: Combine into one score (e.g., $F_1$)
  - *Pareto/frontier*: Analyze trade-off without collapsing (e.g., ROC curve)

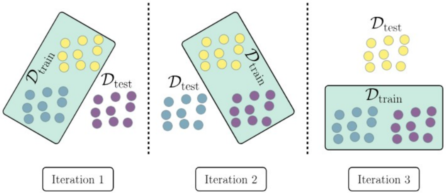
| **True Values** | **Predicted Values** | | |
|---|---|---|---|
| | **Pos** | **Neg** | |
| **Pos** | TP | FN | **Sensitivity/Recall/TPR** $= \frac{TP}{TP+FN}$ |
| **Neg** | FP | TN | **Specificity** $= \frac{TN}{TN+FP}$ |
| | **Precision** $= \frac{TP}{TP+FP}$ | **NPV** $= \frac{TN}{TN+FN}$ | $\mathbf{F}_1 = \frac{2\,\text{Precision Recall}}{\text{Precision} + \text{Recall}}$ |

# BUSINESS OBJECTIVES: VALIDATION STRATEGY

**Key question**: How to validate the model reliably and realistically?

**1. Strategic Considerations**

- **What should your model generalize to?**
  - *Blocking factors* (e.g., customers, hospitals, product types)
  - *Temporal effects* (e.g., seasonality, trends)
  - *Spatial effects* (e.g., regions, branches)
- **Simulate realistic deployment:**
  - **Time series:** Train on data from 2016–2020, deploy in 2021
  - **Validation:** Mimic deployment – train on 2016–2019, validate on 2020



# BUSINESS OBJECTIVES: VALIDATION STRATEGY

**Key question**: How to validate the model reliably and realistically?

## 1. Strategic Considerations

- **What should your model generalize to?**
  - *Blocking factors* (e.g., customers, hospitals, product types)
  - *Temporal effects* (e.g., seasonality, trends)
  - *Spatial effects* (e.g., regions, branches)
- **Simulate realistic deployment:**
  - **Time series:** Train on data from 2016–2020, deploy in 2021
  - **Validation:** Mimic deployment – train on 2016–2019, validate on 2020

## 2. Beware of Data Leakage

- **Target Leakage:** Features leak future info not available at prediction time
  - Example: "Total Sales in Next Month" used to predict this month
- **Train-Test Leakage:** Information from the test set is used during training
  - Example: Scaling the full dataset before splitting

# 2. Data Acquisition and Understanding

# DATA SOURCES AND DATA VERSIONING

**Key question:** What data is available and where does it come from?

- **Internal vs. External:** External feeds may change format or be discontinued
- **Structured vs. Unstructured:** Tables vs. free-text, images, audio, etc.
- **Databases vs. Files:** DBs offer querying, consistency and security
- **On-Premise vs. Cloud:** Governed by regulations, cost and team expertise

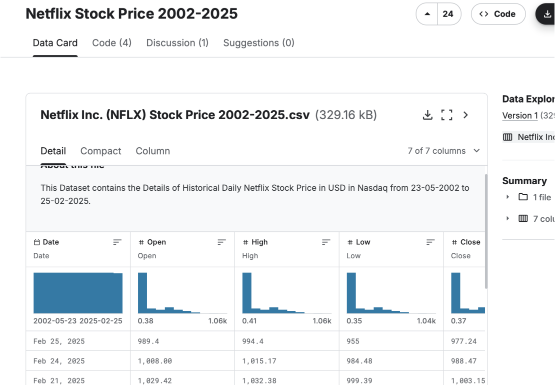
**Data Versioning:**

- Data can evolve - use version control for data, like `git` for code
- Tools: `DVC` (Data Version Control), `Pachyderm`

# DATASHEETS ▶ "Gebru et al" 2018

*Datasheets* summarize key metadata about a dataset's content, structure, provenance, and known limitations to support informed and responsible use.

- Why and how was the data collected?
- What are its limitations and ethical concerns?
- For which tasks is it suitable or unsuitable?

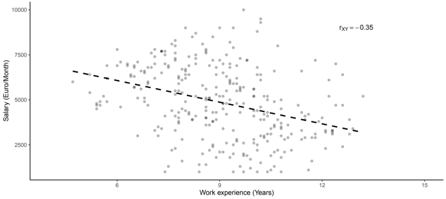


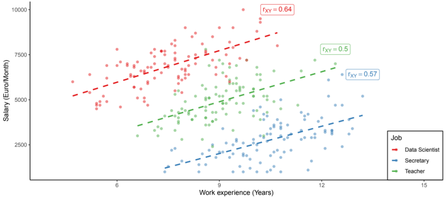

Source: ▶ "Kaggle Datasets" 2025

# DATA EXPLORATION

**Key question**: What features are in the data, and what do they reveal?

- **Build a data dictionary**:
  - Feature meaning, units, valid ranges/category-levels
  - Missing value encoding and special codes
- **Examine structure and quality to gain understanding**:
  - Compute (stratified) summary statistics and inspect missingness
  - Plot univariate and bivariate distributions
  - Identify outliers, anomalies, imbalances, redundancy, or high correlation
- **Watch for confounders:** unmeasured variables may distort associations



# DATA QUALITY

**Key question:** Is the data fit for modeling? *Garbage in, garbage out.*

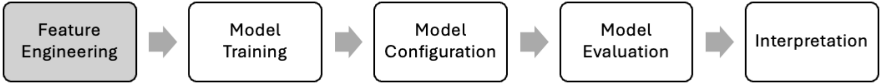
- Assess missingness: How many values are missing, and where?
- Detect implausible or inconsistent values (requires domain expertise)
- Standardize formats (e.g., date/time parsing)
- Drop near-constant or non-informative features
- Remove redundancy and obsolete variables

**Watch for data bias:**

- *Sampling bias:* E.g., a survey at 10am in the city center misses working adults
- *Response bias:* Online reviews often reflect extreme opinions, not the median

# 3. Modeling

# FEATURE ENGINEERING




**Feature Engineering** transforms raw data into informative input variables for modeling.

- **Transformation**: Scaling, log-transform, one-hot encoding
- **Selection**: Wrapper and filter methods, model-based importance
- **Dimensionality Reduction**: PCA, t-SNE
- **Creation**: Aggregation, binning, interaction terms

**Best Practices:**

- Avoid overengineering: irrelevant features introduce noise
- Fit transformations **only on training data**; reuse parameters for test data
- Prevent leakage: never use test data in feature construction

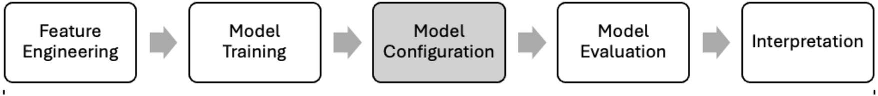
# MODEL TRAINING

Feature Engineering → **Model Training** → Model Configuration → Model Evaluation → Interpretation

Treat model training like a scientific experiment:

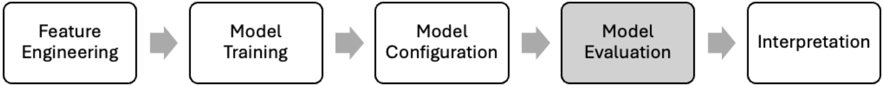
- **Track all inputs precisely:**
  - Dataset version
  - Algorithm and hyperparameters
  - Code version (e.g., Git commit/tag)
- **Proceed iteratively:**
  - *Start simple* – build a fast, interpretable baseline
  - Gradually increase model complexity and validate each improvement

# MODEL CONFIGURATION




- How do you track the exact configuration of a model experiment?
- How do you record small changes between experiments?
- Store in plain text (e.g., `.yaml`, `.json`) for readability and version control
- Recommended tool: structured configuration managers
  - **Python**: ▸ "Hydra" n.d. - structured, hierarchical configuration management
  - **R**: ▸ "R config package" n.d. - environment-based YAML setup

# MODEL EVALUATION




- Select models based on predefined **metrics** (e.g., accuracy, AUC, RMSE)
- Use the **validation or test set** in line with your validation strategy
- Assess **robustness**: How does the model respond to noise, perturbations, or adversarial inputs?
  $\Rightarrow$ Add synthetic noise and measure changes in model parameters/performance

# INTERPRETATION AND FAIRNESS

Feature Engineering → Model Training → Model Configuration → Model Evaluation → **Interpretation**

To understand and trust your model, apply methods from **Interpretable Machine Learning (IML)**:

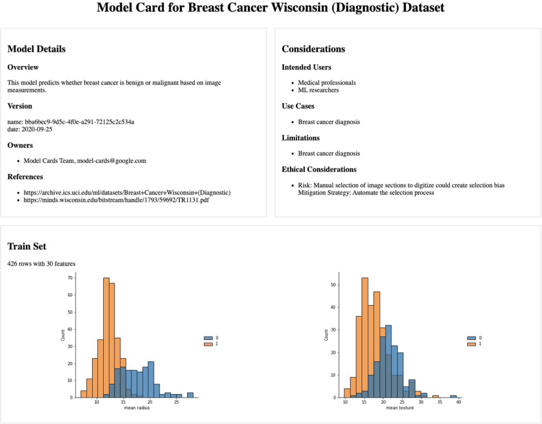
- **Feature Importance**: Which features drive predictions globally?
- **Effects & Interactions**: Visualize marginal effects and feature interactions
- **Local Explanations**: Explain individual predictions
- **Bias Detection**: Reveal fairness issues and unintended discrimination
  - Example: Amazon’s 2018 hiring tool penalized resumes with “women’s” due to biased training data ▸ “The Guardian” 2018

# MODEL CARDS FOR MODEL REPORTING

▸ "Mitchell et al." 2019

**Model Card**: Documentation of the model similar to a Datasheet for data

- **Model Details**: Algorithm, parameters, model version
- **Intended Use**: Intended use and out-of-scope use cases
- **Performance**: Key metrics, validation approaches
- **Analyses**: Feature/target distributions in train/test data
- **Ethical Considerations**: Failure modes, fairness concerns, inappropriate use



**Model Card for Breast Cancer Wisconsin (Diagnostic) Dataset**

**Model Details**

**Overview**

This model predicts whether breast cancer is benign or malignant based on image measurements.

**Version**

name: bba6bec9-9d5c-4f0e-a291-72125c2c534a
date: 2020-09-25

**Owners**

- Model Cards Team, model-cards@google.com

**References**

- https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
- https://minds.wisconsin.edu/bitstream/handle/1793/59692/TR1131.pdf

**Considerations**

**Intended Users**

- Medical professionals
- ML researchers

**Use Cases**

- Breast cancer diagnosis

**Limitations**

- Breast cancer diagnosis

**Ethical Considerations**

- Risk: Manual selection of image sections to digitize could create selection bias
  Mitigation Strategy: Automate the selection process

**Train Set**

426 rows with 30 features

# 4. Model Governance

# MODEL GOVERNANCE

- Model Governance includes **people, processes, and technologies** needed to manage and protect your ML models.
- Ensures long-term maintainability and adaptation as requirements evolve.
- Two extremes must be avoided:
  - **Repression**: too many rules, rigid standards, excessive control
  - **Chaos**: too much speed, freedom, creativity, and unstructured change

Model governance steps are:


*"Developing and deploying ML systems is relatively fast and cheap, but maintaining them over time is difficult and expensive."*

▸ *“Hidden Technical Debt in ML Systems” n.d.*

# DEPLOYMENT

**Deployment** means making a model accessible to users (humans or other systems).

- Common deployment strategies:
  - **One-off**: Manually trained and deployed once; no regular updates
  - **Batch**: Periodically retrained on recent data (e.g., weekly updates)
  - **Real-time**: Continuously updated on streaming data; often latency-critical
- For frequent updates, establish model **versioning and lifecycle tracking**
  - Tools like ▸ "MLflow" n.d. provide version control and a model registry

# MONITORING

**Monitoring** ensures that deployed models work reliably and flags issues in production:

- **Pipeline Failures**: Bugs in data ingestion or transformation logic
- **Data Drift**: Changes in input distribution
  - Example: An image classifier trained on summer data fails when images contain winter snow
  - Also affects evolving targets (e.g., new categories or value ranges)
  - **Remedy**: Label sufficient new data and retrain the model
- **Concept Drift**: Changes in the meaning of labels
  - Example: Redefining “high blood pressure” from 140mmHg to 135mmHg
  - **Remedy**: Relabel historical data and retrain

# USER FEEDBACK, REINFORCEMENT

**Handover:** At project completion, engage users and transfer responsibilities:

- **System Validation**: Confirm with stakeholders that the solution meet their needs
- **Project Handover**: Transfer system ownership to the production team

**Feedback loops** occur when model outputs influence future inputs:

- **Direct Loops**: A model affects the data it later learns from
  - Example: Netflix recommends certain movies, then only learns from user responses to those
- **Hidden Loops**: Multiple models indirectly affect each other via the environment
  - Example: Flash crashes triggered by interacting trading algorithms amplifying market reactions