ML Pipelines

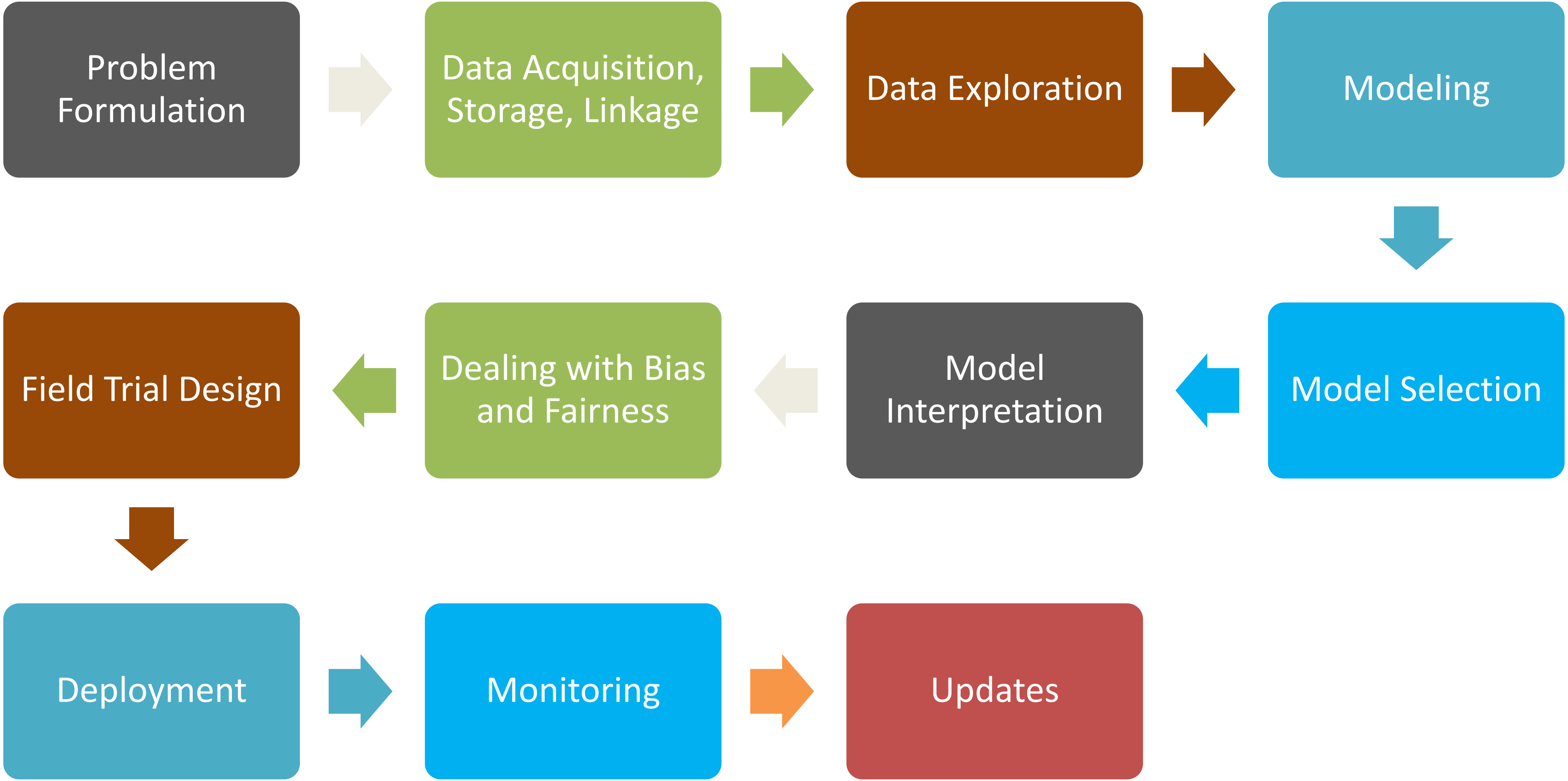
Rayid Ghani

Carnegie Mellon University

MACHINE LEARNING DEPARTMENT

Heinz College

INFORMATION SYSTEMS • PUBLIC POLICY • MANAGEMENT



Problem Formulation
Data Acquisition, Storage, Linkage
Data Exploration
Modeling
Model Selection
Model Interpretation
Dealing with Bias and Fairness
Field Trial Design
Deployment
Monitoring
Updates

Things we will cover

- What is an ML Pipeline?
- Why should we build ML pipelines?
- What components should it have?
- Best Practices
- Good Examples

What is an ML Pipeline?

- Supports end-to-end workflow for an ML project/system
- Modular
- Reconfigurable

Why build a pipeline?

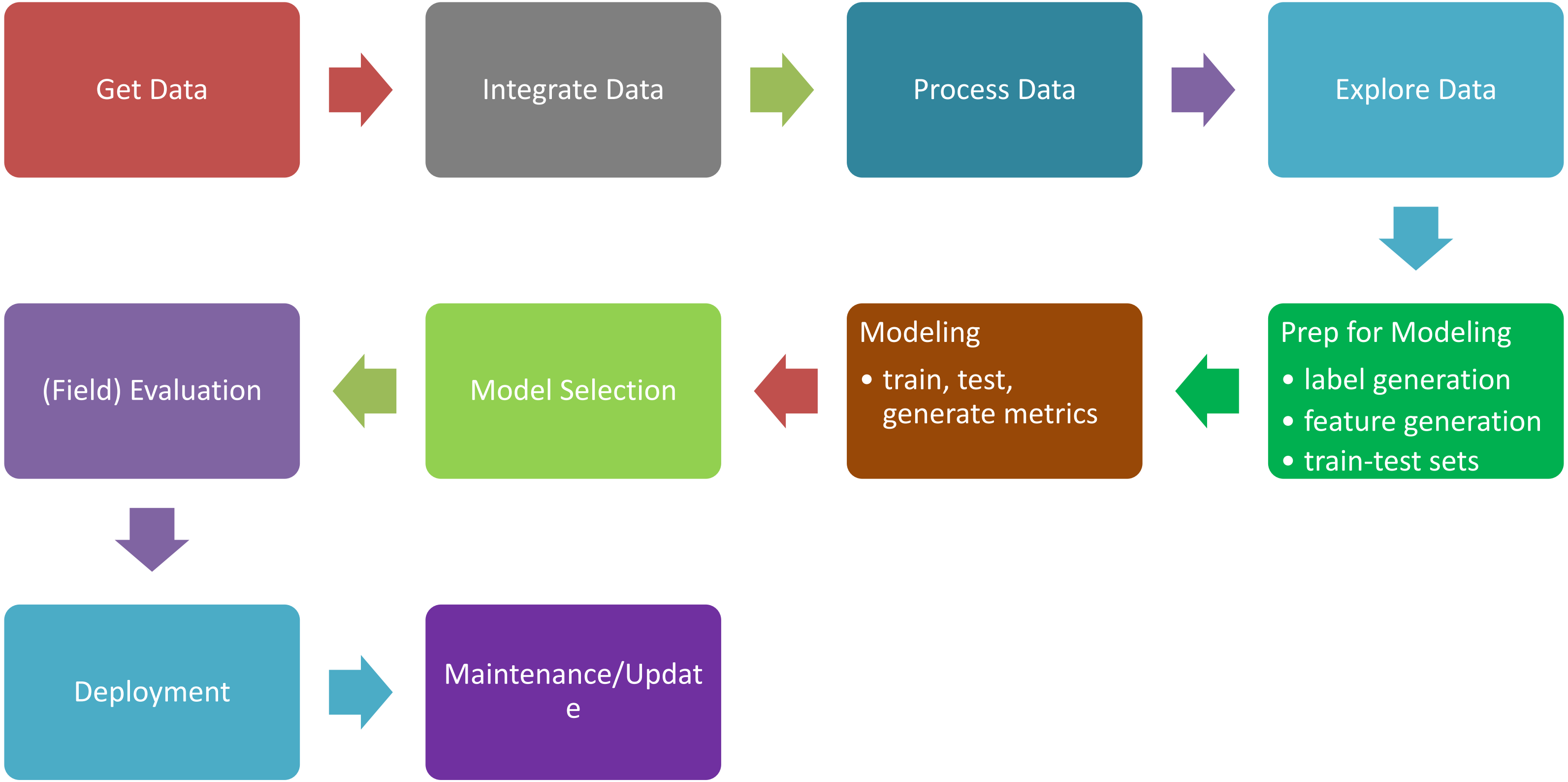
- Reusable across projects
- Test new ideas, components, hypothesis easily
- Reduce bugs/errors
- Allows reproducibility of analysis and results

What makes a pipeline?

- Inputs
- Components
- (Intermediate and final) outputs

Pipeline Flow & Components

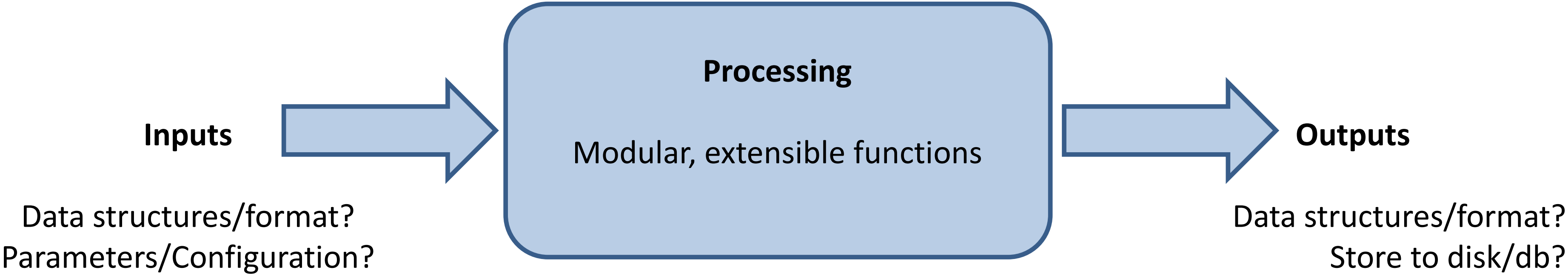
Pipeline Flow



What components does a pipeline have?

- Read/Load Data (from csv, db, api)
- Integrate Data (dedupe, link)
- Process Data (cleaning)
- Explore Data (descriptive stats, correlations, outliers, over time, clustering)
- Modeling Prep
    - Create training and validation (test) sets
    - Missing values (fill/impute, create dummy)
    - Transformations (scale/normalize, log, square, root)
    - Feature Generation
    - Label Generation
    - Define metric(s)
- Modeling
    - Build model(s) on training sets
    - Apply model(s) on test sets
    - Calculate metric(s)
- Model Selection
- Field Trial
- Deploy
- Maintain

Things to keep in mind about each component



Components: Data Acquisition & Integration

- Get Data
  - API, CSV, Database
- Store Data
  - Database
- Integrate Data
  - Record Linkage

Components: Explore and Prepare data

- Data Exploration
  - Distributions
  - Missing Values
  - Correlations
  - Other Patterns
- Pre-Processing
  - Leakage
  - Deal with Missing values
  - Scaling
  - Data errors

Components: Feature Creation

- Data comes with fields or columns (if it’s even structured), not features
- Common Features
  - Discretization
  - Transformations
  - Interactions/Conjunctions
  - Disaggregation
  - Aggregations
    - Temporal
    - Spatial
- How are you handling imputation of missing values?

Components: Model Selection

- Select pool of methods applicable for task: what model types will you use?
- Select space of hyperparameters to explore for each model type

Components: Validation

- Using historical data
  - Methodology
  - Metric

- Field Experiment
  - Methodology
  - Metric

Deployment

- Model monitoring
- Re-training
  - How often?
  - Re-select methods?
- Scoring

What types of variations do you want to test using your pipeline?

- Different models
- Model parameters
- Different Labels/Outcomes
- Different Deployment Settings
- Different Feature (Groups)
- Different Metrics

Best Practices

- Draw a diagram of the pipeline:
  - What function runs each step? What are the inputs? What are the outputs?
- Config files (yaml, json, py)
- Make each step modular and extensible so it can easily be re-used
- Build a **simple**, end-to-end version first, then add more functionality
- Think about how you'll store outputs:
  - Store models as pickles
  - Store predictions in databases
  - Store evaluation metrics in databases
  - Sample results schema

Useful Resources

- Data Science Project Scoping Guide
- Open Source Data Science Tools
  - Triage: ML Toolkit
  - Aequitas: Bias Audit Tool
  - Code for projects: www.github.com/dssg

Rayid Ghani

Carnegie Mellon University

Machine Learning Department

Heinz College
Information Systems • Public Policy • Management

rayid@cmu.edu