Machine Learning in Production

Data Quality

More Quality Assurance...

Fundamentals of Engineering AI-Enabled Systems

Holistic system view: AI and non-AI components, pipelines, stakeholders, environment interactions, feedback loops

Requirements:
System and model goals
User requirements
Environment assumptions
Quality beyond accuracy
Measurement
Risk analysis
Planning for mistakes

Architecture + design:
Modeling tradeoffs
Deployment architecture
Data science pipelines
Telemetry, monitoring
Anticipating evolution
Big data processing
Human-AI design

Quality assurance:
Model testing
Data quality
QA automation
Testing in production
Infrastructure quality
Debugging

Operations:
Continuous deployment
Contin. experimentation
Configuration mgmt.
Monitoring
Versioning
Big data
DevOps, MLOps

Teams and process: Data science vs software eng. workflows, interdisciplinary teams, collaboration points, technical debt

Responsible AI Engineering

Provenance, versioning, reproducibility

Safety

Security and privacy

Fairness

Interpretability and explainability

Transparency and trust

Ethics, governance, regulation, compliance, organizational culture

Readings

Required reading:

- Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021, May). "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In Proc. Conference on Human Factors in Computing Systems (pp. 1-15).

Recommended reading:

- Schelter, S., et al. Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), pp.1781-1794.

Learning Goals

- Distinguish precision and accuracy; understanding the better models vs more data tradeoffs
- Use schema languages to enforce data schemas
- Design and implement automated quality assurance steps that check data schema conformance and distributions
- Devise infrastructure for detecting data drift and schema violations
- Consider data quality as part of a system; design an organization that values data quality

Data-Quality Challenges

Data cleaning and repairing account for about 60% of the work of data scientists.

Own experience?

Case Study: Inventory Management

Data Comes from Many Sources

Manually entered

Actions from IT systems

Logging information, traces of user interactions

Sensor data

Crowdsourced

Many Data Sources

sources of different reliability and quality

Inventory Database

Product Database:

ID	Name	Weight	Description	Size	Vendor
...	...	...	...	...	...

Stock:

ProductID	Location	Quantity
...	...	...

Sales history:

UserID	ProductId	DateTime	Quantity	Price
...	...	...	...	...

Raw Data is an Oxymoron

Shipment Received Note

Goods Received Receipt

Supplier: Mr. Adam Smith		**Destination: Marlon De compo**			
Shipment Date	June 17, 2015	Receipt No.	09XC10	Genre: Shipment Delivery Receipt	
Order Number	67-NY25	Delivery Duration	5 days		
Goods Details	**Packing Weight**	**Packing Size**	**Quantity**	**Delivered Quantity**	**Comments**
Received by: Mr. Boucle Norbed Shakar					

Recommended Reading: Gitelman, Lisa, Virginia Jackson, Daniel Rosenberg, Travis D. Williams, Kevin R. Brine, Mary Poovey, Matthew Stanley et al. "Data bite man: The work of sustaining a long-term study." In "Raw Data" Is an Oxymoron, (2013), MIT Press: 147-166.

What makes good quality data?

Accuracy: The data was recorded correctly.

Completeness: All relevant data was recorded.

Uniqueness: The entries are recorded once.

Consistency: The data agrees with itself.

Timeliness: The data is kept up to date.

Data is noisy

Unreliable sensors or data entry

Wrong results and computations, crashes

Duplicate data, near-duplicate data

Out of order data

Data format invalid

Examples in inventory system?

Data changes

System objective changes over time

Software components are upgraded or replaced

Prediction models change

Quality of supplied data changes

User behavior changes

Assumptions about the environment no longer hold

Examples in inventory system?

Users may deliberately change data

Users react to model output

Users try to game/deceive the model

Examples in inventory system?

Accuracy vs Precision

Accuracy: Reported values (on average) represent real value

Precision: Repeated measurements yield the same result

Accurate, but imprecise: Average over multiple measurements

Inaccurate, but precise: ?

Accuracy and Precision in Warehouse Data?

Data Quality and Machine Learning

More data -> better models (up to a point, diminishing effects)

Noisy data (imprecise) -> less confident models, more data needed

- some ML techniques are more or less robust to noise (more on robustness in a later lecture)

Inaccurate data -> misleading models, biased models

-> Need the "right" data

-> Invest in data quality, not just quantity

Poor Data Quality has Consequences

(often delayed consequences)

Example: Systematic bias in labeling

Poor data quality leads to poor models

Not detectable in offline evaluation

Problem in production -- now difficult to correct

Delayed Fixes increase Repair Cost

Copyright 1998 Steven C. McConnell. Reprinted with permission from *Software Project Survival Guide* (Microsoft Press, 1998).

Data Cascades

Detection almost always delayed! Expensive rework. Difficult to detect in offline evaluation.

Sambasivan, N., et al. (2021, May). "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In Proc. CHI (pp. 1-15).

Data Schema

Ensuring basic consistency about shape and types

Dirty Data: Example

TABLE: CUSTOMER

ID	Name	Birthday	Age	Sex	Phone	ZIP
3456	Ford, Harrison	18.2.76	43	M	9999999999	15232
3456	Mark Hamil	33.8.81	43	M	6173128718	17121
3457	Kim Kardashian	11.10.56	63	M	4159102371	94016

TABLE: ADDRESS

ZIP	City	State
15232	Pittsburgh	PA
94016	Sam Francisco	CA
73301	Austin	Texas

Problems with this data?

Schema Problems: Uniqueness, data format, integrity, ...

- Illegal attribute values: `bdate=30.13.70`
- Violated attribute dependencies: `age=22, bdate=12.02.70`
- Uniqueness violation: `(name="John Smith", SSN="123456"), (name="Peter Miller", SSN="123456")`
- Referential integrity violation: `emp=(name="John Smith", deptno=127)` if department 127 not defined

Data Schema

Define expected format of data

- expected fields and their types
- expected ranges for values
- constraints among values (within and across sources)

Data can be automatically checked against schema

Protects against change; explicit interface between components

Schema in Relational Databases

```
CREATE TABLE employees (
    emp_no       INT              NOT NULL,
    birth_date   DATE             NOT NULL,
    name         VARCHAR(30)      NOT NULL,
    PRIMARY KEY (emp_no));
CREATE TABLE departments (
    dept_no      CHAR(4)          NOT NULL,
    dept_name    VARCHAR(40)      NOT NULL,
    PRIMARY KEY (dept_no), UNIQUE  KEY (dept_name));
CREATE TABLE dept_manager (
   dept_no       CHAR(4)          NOT NULL,
```

Which Problems are Schema Problems?

TABLE: CUSTOMER

ID	Name	Birthday	Age	Sex	Phone	ZIP
3456	Ford, Harrison	18.2.76	43	M	9999999999	15232
3456	Mark Hamil	33.8.81	43	M	6173128718	17121
3457	Kim Kardashian	11.10.56	63	M	4159102371	94016

TABLE: ADDRESS

ZIP	City	State
15232	Pittsburgh	PA
94016	Sam Francisco	CA
73301	Austin	Texas

What Happens When new Data Violates Schema?

Schema-Less Data Exchange

- CSV files
- Key-value stores (JSon, XML, Nosql databases)
- Message brokers
- REST API calls
- R/Pandas Dataframes

```
2022-10-06T01:31:18,230550,GET /rate/narc+2002=4
2022-10-06T01:31:19,332644,GET /rate/i+am+love+2009=4
```

```
{"user_id":5,"age":26,"occupation":"scientist","gender":"M"}
```

Schema Library: Apache Avro

```
{   "type": "record",
    "namespace": "com.example",
    "name": "Customer",
    "fields": [{
            "name": "first_name",
            "type": "string",
            "doc": "First Name of Customer"
        },
        {
            "name": "age",
            "type": "int",
```

Schema Library: Apache Avro

Schema specification in JSON format

Serialization and deserialization with automated checking

Native support in Kafka

Benefits

- Serialization in space efficient format
- APIs for most languages (ORM-like)
- Versioning constraints on schemas

Drawbacks

- Reading/writing overhead
- Binary data format, extra tools needed for reading
- Requires external schema and maintenance
- Learning overhead

Speaker notes

Further readings eg https://medium.com/@stephane.maarek/introduction-to-schemas-in-apache-kafka-with-the-confluent-schema-registry-3bf55e401321, https://www.confluent.io/blog/avro-kafka-data/, https://avro.apache.org/docs/current/

Many Schema Libraries/Formats

Examples

- Avro
- XML Schema
- Protobuf
- Thrift
- Parquet
- ORC

Discussion: Data Schema Constraints for Inventory System?

Product Database:

ID	Name	Weight	Description	Size	Vendor
…	…	…	…	…	…

Stock:

ProductID	Location	Quantity
…	…	…

Sales history:

UserID	ProductId	DateTime	Quantity	Price
…	…	…	…	…

Summary: Schema

Basic structure and type definition of data

Well supported in databases and many tools

Very low bar of data quality

Instance-Level Problems

Inconsistencies, wrong values

Dirty Data: Example

TABLE: CUSTOMER

ID	Name	Birthday	Age	Sex	Phone	ZIP
3456	Ford, Harrison	18.2.76	43	M	9999999999	15232
3456	Mark Hamil	33.8.81	43	M	6173128718	17121
3457	Kim Kardashian	11.10.56	63	M	4159102371	94016

TABLE: ADDRESS

ZIP	City	State
15232	Pittsburgh	PA
94016	Sam Francisco	CA
73301	Austin	Texas

Problems with the data beyond schema problems?

Instance-Level Problems

- Missing values: `phone=9999-999999`
- Misspellings: `city=Pittsburg`
- Misfielded values: `city=USA`
- Duplicate records: `name=John Smith, name=J. Smith`
- Wrong reference: `emp=(name="John Smith", deptno=127)` if department 127 defined but wrong

Can we detect these?

Further readings: Rahm, Erhard, and Hong Hai Do. Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. 23.4 (2000): 3-13.

Discussion: Instance-Level Problems?

Data Cleaning Overview

Data analysis / Error detection

- Usually focused on specific kind of problems, e.g., duplication, typos, missing values, distribution shift
- Detection in input data vs detection in later stages (more context)

Error repair

- Repair data vs repair rules, one at a time or holistic
- Data transformation or mapping
- Automated vs human guided

Error Detection Examples

Illegal values: min, max, variance, deviations, cardinality

Misspelling: sorting + manual inspection, dictionary lookup

Missing values: null values, default values

Duplication: sorting, edit distance, normalization

Error Detection: Example

TABLE: CUSTOMER

ID	Name	Birthday	Age	Sex	Phone	ZIP
3456	Ford, Harrison	18.2.76	43	M	9999999999	15232
3456	Mark Hamil	33.8.81	43	M	6173128718	17121
3457	Kim Kardashian	11.10.56	63	M	4159102371	94016

TABLE: ADDRESS

ZIP	City	State
15232	Pittsburgh	PA
94016	Sam Francisco	CA
73301	Austin	Texas

Can we (automatically) detect instance-level problems? Which problems are domain-specific?

Example Tool: Great Expectations

```
expect_column_values_to_be_between(
    column="passenger_count",
    min_value=1,
    max_value=6
)
```

Supports schema validation and custom instance-level checks.

https://greatexpectations.io/

Example Tool: Great Expectations

great_expectations Home / taxi.demo / 20200819T024609.241003Z / 2020-08-19T02:46:09.241003+00:00 / cbb8bd044ccaa28d4db5e3d59c0be748

Actions

Validation Filter:

Show All | Failed Only

How to Edit This Suite

Show Walkthrough

Table of Contents

- Overview
- Table-Level Expectations
- passenger_count

	dropoff_location_id, payment_type, fare_amount, extra, mta_tax, tip_amount, tolls_amount, improvement_surcharge, total_amount, congestion_surcharge	'payment_type', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount', 'improvement_surcharge', 'total_amount', 'congestion_surcharge']

passenger_count

Search

Status	Expectation	Observed Value
✓	values must never be null.	100% not null
✓	distinct values must belong to this set: 1.0 2.0 3.0 4.0 5.0 6.0.	[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]
✓	Kullback-Leibler (KL) divergence with respect to the following distribution must be lower than 0.6.	KL Divergence: None (-infinity, infinity, or NaN)

https://greatexpectations.io/

Data Quality Rules

Invariants on data that must hold

Typically about relationships of multiple attributes or data sources, eg.

- ZIP code and city name should correspond
- User ID should refer to existing user
- SSN should be unique
- For two people in the same state, the person with the lower income should not have the higher tax rate

Classic integrity constraints in databases or conditional constraints

ML for Detecting Inconsistencies

	DBAName	AKAName	Address	City	State	Zip
t1	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	***Chicago***	IL	***60608***
t2	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t3	John Veliotis Sr.	Johnnyo's	3465 S Morgan ST	Chicago	IL	***60609***
t4	***Johnnyo's***	Johnnyo's	3465 S Morgan ST	***Cicago***	IL	60608

Conflicts

Does not obey data distribution

Conflict

Image source: Theo Rekatsinas, Ihab Ilyas, and Chris Ré, "HoloClean - Weakly Supervised Data Repairing." Blog, 2017.

Example: HoloClean

- User provides rules as integrity constraints (e.g., "two entries with the same name can't have different city")
- Detect violations of the rules in the data; also detect statistical outliers
- Automatically generate repair candidates (with probabilities)

Image source: Theo Rekatsinas, Ihab Ilyas, and Chris Ré, "HoloClean - Weakly Supervised Data Repairing." Blog, 2017.

Discovery of Data Quality Rules

Rules directly taken from external databases

- e.g. zip code directory

Given clean data,

- several algorithms that find functional relationships ($X \Rightarrow Y$) among columns
- algorithms that find conditional relationships (if Z then $X \Rightarrow Y$)
- algorithms that find denial constraints (X and Y cannot cooccur in a row)

Given mostly clean data (probabilistic view),

- algorithms to find likely rules (e.g., association rule mining)
- outlier and anomaly detection

Given labeled dirty data or user feedback,

- supervised and active learning to learn and revise rules
- supervised learning to learn repairs (e.g., spell checking)

Further reading: Ilyas, Ihab F., and Xu Chu. Data cleaning. Morgan & Claypool, 2019.

Excursion: Association rule mining

- Sale 1: Bread, Milk
- Sale 2: Bread, Diaper, Beer, Eggs
- Sale 3: Milk, Diaper, Beer, Coke
- Sale 4: Bread, Milk, Diaper, Beer
- Sale 5: Bread, Milk, Diaper, Coke

Rules

- {Diaper, Beer} -> Milk (40% support, 66% confidence)
- Milk -> {Diaper, Beer} (40% support, 50% confidence)
- {Diaper, Beer} -> Bread (40% support, 66% confidence)

(also useful tool for exploratory data analysis)

Further readings: Standard algorithms and many variations, see Wikipedia

Discussion: Data Quality Rules

Drift

Monitoring for Changes

https://www.anomalo.com/

Drift & Model Decay

Concept drift (or concept shift)

- properties to predict change over time (e.g., what is credit card fraud)
- model has not learned the relevant concepts
- over time: different expected outputs for same inputs

Data drift (or covariate shift, distribution shift, or population drift)

- characteristics of input data changes (e.g., customers with face masks)
- input data differs from training data
- over time: predictions less confident, further from training data

Upstream data changes

- external changes in data pipeline (e.g., format changes in weather service)
- model interprets input data incorrectly
- over time: abrupt changes due to faulty inputs

≡ **How do we fix these drifts?**

Speaker notes

- fix1: retrain with new training data or relabeled old training data
  - fix2: retrain with new data
  - fix3: fix pipeline, retrain entirely

On Terminology

Concept and data drift are separate concepts

In practice and literature not always clearly distinguished

Colloquially encompasses all forms of model degradations and environment changes

Define term for target audience

Breakout: Drift in the Inventory System

What kind of drift might be expected?

As a group, tagging members, write plausible examples in `#lecture`:

- *Concept Drift:*
- *Data Drift:*
- *Upstream data changes:*

Watch for Degradation in Prediction Accuracy

Indicators of Concept Drift

How to detect concept drift in production?

Indicators of Concept Drift

Model degradations observed with telemetry

Telemetry indicates different outputs over time for similar inputs

Relabeling training data changes labels

Interpretable ML models indicate rules that no longer fit

(many papers on this topic, typically on statistical detection)

Indicators of Data Drift

How to detect data drift in production?

Indicators of Data Drift

Model degradations observed with telemetry

Distance between input distribution and training distribution increases

Average confidence of model predictions declines

Relabeling of training data retains stable labels

Detecting Data Drift

- Compare distributions over time (e.g., t-test)
- Detect both sudden jumps and gradual changes
- Distributions can be manually specified or learned (see invariant detection)

Data Distribution Analysis

Plot distributions of features (histograms, density plots, kernel density estimation)

- Identify which features drift

Define distance function between inputs and identify distance to closest training data (e.g., energy distance, see also kNN)

Anomaly detection and "out of distribution" detection

Compare distribution of output labels

Data Distribution Analysis Example

https://rpubs.com/ablythe/520912

Microsoft Azure Data Drift Dashboard

Image source and further readings: Detect data drift (preview) on models deployed to Azure Kubernetes Service (AKS)

Dealing with Drift

Regularly retrain model on recent data

- Use evaluation in production to detect decaying model performance

Involve humans when increasing inconsistencies detected

- Monitoring thresholds, automation

Monitoring, monitoring, monitoring!

Breakout: Drift in the Inventory System

What kind of monitoring for previously listed drift in Inventory scenario?

Data Quality is a System-Wide Concern

"Everyone wants to do the model work, not the data work"

Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021, May). "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (pp. 1-15).

Data flows across components

Data Quality is a System-Wide Concern

Data flows across components, e.g., from user interface into database to crowd-sourced labeling team into ML pipeline

Documentation at the interfaces is important

Humans interacting with the system

- Entering data, labeling data
- Observed with sensors/telemetry
- Incentives, power structures, recognition

Organizational practices

- Value, attention, and resources given to data quality

Data Quality Documentation

Teams rarely document expectations of data quantity or quality

Data quality tests are rare, but some teams adopt defensive monitoring

- Local tests about assumed structure and distribution of data
- Identify drift early and reach out to producing teams

Several ideas for documenting distributions, including Datasheets and Dataset Nutrition Label

- Mostly focused on static datasets, describing origin, consideration, labeling procedure, and distributions; Example

Gebru, Timnit, et al. "Datasheets for datasets." Communications of the ACM 64, no. 12 (2021).
Nahar, Nadia, et al. “Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process.” In Pro. ICSE, 2022.

Common Data Cascades

Physical world brittleness

- Idealized data, ignoring realities and change of real-world data
- Static data, one time learning mindset, no planning for evolution

Inadequate domain expertise

- Not understand. data and its context
- Involving experts only late for trouble shooting

Conflicting reward systems

- Missing incentives for data quality
- Not recognizing data quality importance, discard as technicality
- Missing data literacy with partners

Poor (cross-org.) documentation

- Conflicts at team/organization boundary
- Undetected drift

Sambasivan, N., et al. (2021). "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In Proc. Conference on Human Factors in Computing Systems.

Discussion: Possible Data Cascades?

- Interacting with physical world brittleness
- Inadequate domain expertise
- Conflicting reward systems
- Poor (cross-organizational) documentation

Ethics and Politics of Data

Raw data is an oxymoron

Incentives for Data Quality? Valuing Data Work?

Bonus: Data Linter

Further readings: Nick Hynes, D. Sculley, Michael Terry. "The Data Linter: Lightweight Automated Sanity Checking for ML Data Sets." NIPS Workshop on ML Systems (2017)

Excursion: Static Analysis and Code Linters

Automate routine inspection tasks

```
if (user.jobTitle = "manager") {
    ...
}
```

```
function fn() {
    x = 1;
    return x;
    x = 3; // dead code
}
```

Static Analysis

- Analyzes possible executions of the code, without running it
- Different levels of sophistication
  - Simple heuristic and code patterns (linters)
  - Sound reasoning about all possible program executions
- Tradeoff between false positives and false negatives
- Often supporting annotations needed (e.g., `@Nullable`)
- Tools widely available, open source and commercial

Static Analysis Example

index.js .eslintrc

```
1
2
3   for (i = 0; i <= 10; i++)
4   {
5       if (i == 2)
6
7
8       }
9       var out = "The value is now " + i;
10      document.write(out);
11      document.write( text: "<br/>");
12  }
```

ESLint: Opening curly brace does not appear on the same line as controlling statement. (brace-style)

Static Analysis for Data Science Code

- Lots of research
- Style issues in Python
- Shape analysis of tensors in deep learning
- Analysis of flow of datasets to detect data leakage
- ...

Examples:

- Yang, Chenyang, et al.. "Data Leakage in Notebooks: Static Detection and Better Processes." Proc. ASE (2022).
- Lagouvardos, S. et al. (2020). Static analysis of shape in TensorFlow programs. In Proc. ECOOP.
- Wang, Jiawei, et al. "Better code, better sharing: on the need of analyzing jupyter notebooks." In Proc. ICSE-NIER. 2020.

A Linter for Data?

Data Linter at Google

Miscoding

- Number, date, time as string
- Enum as real
- Tokenizable string (long strings, all unique)
- Zip code as number

Outliers and scaling

- Unnormalized feature (varies widely)
- Tailed distributions
- Uncommon sign

Packaging

- Duplicate rows
- Empty/missing data

Further readings: Hynes, Nick, D. Sculley, and Michael Terry. The data linter: Lightweight, automated sanity checking for ML data sets. NIPS MLSys Workshop. 2017.

Summary

- Data from many sources, often inaccurate, imprecise, inconsistent, incomplete, ... -- many different forms of data quality problems
- Many mechanisms for enforcing consistency and cleaning
  - Data schema ensures format consistency
  - Data quality rules ensure invariants across data points
- Concept and data drift are key challenges -- monitor
- Data quality is a system-level concern
  - Data quality at the interface between components
  - Documentation and monitoring often poor
  - Involves organizational structures, incentives, ethics, ...

Further Readings

- Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F. and Grafberger, A., 2018. Automating large-scale data quality verification. Proceedings of the VLDB Endowment, 11(12), pp.1781-1794.
- Polyzotis, Neoklis, Martin Zinkevich, Sudip Roy, Eric Breck, and Steven Whang. "Data validation for machine learning." Proceedings of Machine Learning and Systems 1 (2019): 334-347.
- Polyzotis, Neoklis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2017. “Data Management Challenges in Production Machine Learning.” In Proceedings of the 2017 ACM International Conference on Management of Data, 1723–26. ACM.
- Theo Rekatsinas, Ihab Ilyas, and Chris Ré, “HoloClean - Weakly Supervised Data Repairing.” Blog, 2017.
- Ilyas, Ihab F., and Xu Chu. Data cleaning. Morgan & Claypool, 2019.
- Moreno-Torres, Jose G., Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. "A unifying view on dataset shift in classification." Pattern recognition 45, no. 1 (2012): 521-530.
- Vogelsang, Andreas, and Markus Borg. "Requirements Engineering for Machine Learning: Perspectives from Data Scientists." In Proc. of the 6th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2019.
- Humbatova, Nargiz, Gunel Jahangirova, Gabriele Bavota, Vincenzo Riccio, Andrea Stocco, and Paolo Tonella. "Taxonomy of real faults in deep learning systems." In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, pp. 1110-1121. 2020.