[DeepLearning.AI] **Machine Learning Data Lifecycle in Production**

Notes by **José Hilario**

	Research ML	**Production ML**
Data	Static	Dynamic - shifting
Priority	Highest overall accuracy	Fast inference, good interpretability
Training	Optimal tuning and training	Continuously assess and retrain
Fairness	Very important	Crucial
Challenge	High acc. algorithm	Entire system

"Data is the hardest part of ML and the most important piece to get right... Broken data is the most common cause of problems in production ML systems." (Uber)

The Data Lifecycle

- Managing the entire lifecycle of data: labeling, feature space coverage, minimal dimensionality, maximum predictive data, fairness, rare conditions and cases.
- **Modern software development accounts for:** scalability, extensibility, configuration, consistency and reproducibility, safety and security, modularity, testability, monitoring.
- **Challenges in production-grade ML:** 1) To build integrated ML systems; 2) To continuously operate it in production; 3) To handle continuously changing data; 4) To optimize compute resource costs.
- **ML pipelines outline:** Directed acyclic graphs and pipeline orchestration frameworks; infrastructure to automate; monitor and maintain model training and deployment; CD foundation MLOps reference architecture: Data ⟶ Data cleansing ⟶ Data ingestion ⟶ Data analysis and transformation ⟶ Data validation ⟶ Feature engineering ⟶ Data splitting ⟶ Model building ⟶ Training optimization ⟶ Model validation ⟶ Training at scale ⟶ Final model.

Collecting Data

- Understand users; translate user needs into data problems.
- Ensure data quality, data coverage and high predictive signal.
- Source, store and monitor quality data responsibly.
- **Need meaningful data:** 1) Maximize predictive content; 2) Remove non-informative data; 3) Aim for feature space coverage.
- **Data availability and collection:** 1) What kind of data, and how much data, is available? 2) How often does the new data come in? 3) Is the new data annotated? If not, how hard would it be to annotate it?
- **Get to know your data:** 1) Identify data sources; 2) Check if they are refreshed; 3) Check consistency for values, units and data types; 4) Monitor outliers and errors.
- **Dataset issues:** 1) Inconsistent formatting; 2) Compounding errors from other ML models; 3) Monitor data sources for system issues and outages.
- **Measure data effectiveness:** 1) Intuition about data value can be misleading; 2) Feature engineering helps to maximize the predictive signals; 3) Feature selection helps to measure the predictive signals.
- **Responsible data:** 1) Inputs: open source dataset, web scrapping, synthetic dataset, own dataset, collect live data; 2) Security: policies and methods and means to secure personal data or personally identifiable information (PII); 3) Privacy: proper usage, collection, retention, deletion and storage of data; 4) Give user control of what data can be collected; 5) Compliance with regulations; 6) Protect PII: replace unique values with summary values (aggregation), remove some data to create a less complete picture (redaction); 6) Not to fail users: representational harm (stereotypes), opportunity denial, harm by disadvantage (infer bad associations in particular demographics).
- **Raters:** 1) Generalists (crowd-sourcing tools); 2) SMEs (specialized tools for subject-matter experts); 3) Users (derived labels).
- **Rater keypoints:** 1) Investigate rater context and incentives; 2) Ensure pool diversity; 3) Manage cost; 4) Freshness requirements; 5) Evaluate rater tools.

Labeling Data

- **Taking action:** 1) How to detect problems early on? 2) What are the possible causes? 3) What can be done to solve these issues?
- **Kinds of problems:** 1) Gradual problems (data changes such as trends and seasonality; distribution of features changes; relative importance of features changes); 2) World changes (styles change; score and processes change; competitors change; business expands to other places); 3) Sudden problems (data collection problems such as bad sensors or cameras, bad log data; system problems such as bad software update, loss of network connectivity, system down, bad credentials).
- **Why is it necessary to understand your model?** 1) Errors do not have uniform cost to your business; 2) The data you have is rarely the data you wish you had; 3) Model objective is nearly always a proxy for your business objectives; 4) Some percentage of your users may have a bad experience and knowing which is important to find ways to mitigate this problem.
- **Detecting problems with deployed models:** 1) Data and scope changes; 2) Monitor models and validate data to find problems early on; 3) Changing ground truth: label new training data.
  - **Easy problems:** Ground truth changes slowly (months/years). Model re-training is driven by: 1) Model improvements, better data; 2) Changes in software and/or systems. Labeling: 1) Curated datasets; 2) Crowd-based.
  - **Harder problems:** Ground truth changes faster (weeks). Model re-training is driven by: 1) Declining model performance; 2) Model improvements, better data; 3) Changes in software and/or systems. Labeling: 1) Direct feedback; 2) Crowd-based.
  - **Really hard problems:** Ground truth changes very fast (days/hours/minutes). Model re-training is driven by: 1) Declining model performance; 2) Model improvements, better data; 3) Changes in software and/or systems. Labeling: 1) Direct feedback; 2) Weak supervision.
- **Data labeling:** 1) Direct labeling (process feedback); 2) Human labeling; 3) Semi-supervised labeling; 4) Active learning; 5) Weak supervision.
  - **Direct labeling:** Continuous creation of training dataset. Similar to reinforcement learning rewards.
    - Features from inference requests ⟶ Labels from monitoring predictions ⟶ Join results with inference requests ⟶ Repeat.
    - Advantages: 1) Training dataset is continuously created; 2) Labels evolve quickly; 3) Captures strong label signals.
    - Disadvantages: 1) Hindered by inherent nature of the problem; 2) Failure to capture ground truth; 3) Largely customized design (usually not reusable off-the-shelf).
  - **Human labeling:** Raters examine data and assign labels manually. Go from raw data to raters who create the dataset.
    - Methodology: 1) Collect unlabeled data; 2) Human raters are recruited; 3) Instructions to guide raters are created; 4) Data is divided and assigned to raters; 5) Labels are collected and conflicts are resolved.
    - Advantages: 1) More labels; 2) Pure supervised learning.
    - Disadvantages: 1) Quality consistency; 2) Slow; 3) Expensive (especially when subject-matter experts have to be recruited as raters); 4) Small datasets.

Validating Data

- **Data drift:** Changes in data over time, such as data collected once a day.
- **Data skew:** Difference between two static versions, or different sources, such as training set and serving set.
- **Concept drift:** Change on the statistical properties of the labels over time.
- **Schema skew:** Training and serving data do not conform to the same schema.

- **Distribution skew:** Divergence of training and serving datasets. This can manifest by covariate drift, concept drift, and other kinds of drifts and shifts.
- Skew detection requires continuous evaluation/monitoring of the data.

Type of Shift	**Formalization**
Dataset Shift	$\mathbb{P}_{\text{TRAIN}}(x, y) \neq \mathbb{P}_{\text{SERVE}}(x, y)$
Covariate Shift	$\mathbb{P}_{\text{TRAIN}}(y \mid x) = \mathbb{P}_{\text{SERVE}}(y \mid x)$ $\mathbb{P}_{\text{TRAIN}}(x) \neq \mathbb{P}_{\text{SERVE}}(x)$
Concept Shift	$\mathbb{P}_{\text{TRAIN}}(y \mid x) \neq \mathbb{P}_{\text{SERVE}}(y \mid x)$ $\mathbb{P}_{\text{TRAIN}}(x) = \mathbb{P}_{\text{SERVE}}(x)$

- **Desirable capabilities of a data validation tool:** 1) Generates data statistics and browser visualizations; 2) Infers the data schema; 3) Performs validity checks against schema; 4) Detect training/serving skew; 5) Provides methods that can be used to revise the schema based on domain knowledge; 6) Allows you to scale the computation of statistics over datasets; 7) Analyze specific slices of the dataset; 8) Inspect discrepancies between training and evaluation datasets.

Feature Engineering, Transformation and Selection

- **Squeezing the most out of the data:** 1) Make data useful before training a model on it; 2) Representing data in forms that helps models to learn (e.g., normalized data); 3) Increase predictive quality of the data; 4) Dimensionality reduction is recommended whenever possible. This also helps reduce required computational resources.
- **Goal of feature engineering:** To improve the model's ability to learn while reducing the computational resources they require. It does this by transforming and projecting, eliminating and/or combining the features in the raw data to form a new version of the dataset.
- **Pre-processing operations:** 1) Data cleansing (eliminating or correcting erroneous data); 2) Feature tuning (transformations on data such as normalization and scaling); 3) Representation transformation; 4) Feature extraction; 5) Feature construction; 6) Use of empirical knowledge of the data (e.g. apply data transformations to augment the data).
- **Feature engineering techniques:** 1) Numerical range (scaling, normalizing, standardizing); 2) Grouping (bucketing, bag-of-words).
  - The chosen technique will be very dependent on the particular algorithm that is going to be used (e.g., certain normalization techniques work better for learning using neural networks under certain conditions).
  - Scaling (minmax normalization, z-score standardization) converts values from their natural range into a prescribed range. Some benefits include: 1) Helps NNsconverge faster; 2) Do away with NaNs during training; 3) For each feature, the model learns the right weights.
  - Bucketing and binning are useful for collecting data over time or events.
- Dimensionality reduction in embeddings (PCA, t-SNE, UMAP).
- Feature crosses combines multiple features into one new feature, encoding non-linearity in the feature space, or encoding the same information in fewer features.

Feature Transformation at Scale

- Inconsistencies in feature engineering: 1) Training and serving code paths are different; 2) Diverse deployment scenarios (mobile, server, web); 3) Risks of introducing training-serving skews; 4) Skews will lower the performance of the model.
- Transformations: 1) Instance-level (clipping, multiplying, expanding features, etc.); 2) Full-pass (minmax normalization, standard scaling, bucketizing, etc.).
- When to do transformations on data?
  - Pre-process training dataset: 1) Only needs to be ran once; 2) compute on entire dataset; 3) transformation needs to be reproduced at serving; 4) slower iterations.
  - Within the model: 1) Easy iterations; 2) Transformation guarantees; 3) Expensive transformations; 4) Long model latency; 5) Transformations per batch can produce skew.
  - Per batch: 1) Good when having lots of data; 2) Normalize by average within a batch; 3) Pre-compute average and reuse it during normalization.

Feature Selection

- **Feature space coverage:** Train/test datasets should be representative of the serving dataset. This must be ensured in the production system as well via continuous monitoring. Must consider seasonality, trends, drift and new values in the serving data.
- **Feature selection:** Identify the features that best represent the relationship, and remove the rest. Reduce the size of the feature space in order to reduce storage and I/O requirements, as well as minimizing training and inference costs.
  1. **Unsupervised feature selection:** 1) Doesn't consider the features-target variable relationship; 2) Removes redundant features (correlations).
  2. **Supervised feature selection:** 1) Such as filter methods, statistical tests, wrapper methods or embedded methods; 2) Uses feature-target variable relationship; 3) Selects those features contributing the most.

Data Journey and Data Storage

- Data Journey: Raw features and labels $\longrightarrow$ Input-output map $\longrightarrow$ ML model to learn mapping.
- The data transforms throughout the data journey. Interpreting model results requires an understanding of the data transformations.
- Artifacts are created throughout the execution of the ML pipeline components (data in the different stages of transformation, schema, images, metrics, etc.).
- **Data provenance and lineage:** 1) The chain of transformations that led to the creation of a particular artifact; 2) Important for debugging and reproducibility; 3) Helps with debugging and understanding the ML pipeline (inspection and interpretation of artifacts, comparisons, model evolution); 4) Data lineage: Data protection and regulation (track and organize personal data; regulatory compliance); 5) Data versioning (data pipeline management; code versioning; environment versioning; version control of datasets).
- **Data versioning:** 1) Data pipeline management is a major challenge; 2) ML requires reproducibility; 3) Code versioning; 4) Environment versioning (Docker, Terraform); 5) Version control of datasets (DVC, GitLFS).
- **Metadata.** In ML, the metadata is equivalent to logging in software. Every run of a production ML pipeline generates metadata containing information about the various pipeline components and their executions, training runs, and resulting artifacts (e.g., trained models). In the event of unexpected pipeline behaviors or errors, this metadata can be leveraged to analyze the lineage of pipeline components, and help debug issues.
- **Reliability during data evolution.** The platform needs to be resilient to disruptions from inconsistent data, software errors, scaling issues, problems with misconfigurations, and the orchestration of execution among different components in the pipeline. Data errors should be treated as first-class citizens. As data evolves during its lifecycle, ML pipelines need to: 1) Account for anomaly detection; 2) Account for scalable solutions; 3) Provide resilient mechanisms for disruptions.
- **Enterprise data storage.** Consider the usage of feature stores, databases, data warehouses, data lakes.

Advanced Labeling

- **Active learning.** Select the points to be labeled that would be the most informative for model training. Intelligently sample the data. The learning cycle is: Evaluate ML model on unlabeled data $\rightarrow$ Active learning sampling $\rightarrow$ Human annotator labeling $\rightarrow$ Train ML model $\rightarrow$ Repeat. Some sampling methods are: 1) Margin sampling; 2) Cluster-based sampling; 3) Query-by-committee sampling.
- **Semi-supervised learning.** Start with a small dataset labeled by humans; then get a large pool of unlabeled data; infer their labels based on clusters or structures relating them to the labeled data. We have explained a method based on transductive learning, where label propagation is done based on similarity or community structures. Advantages: 1) This combination boosts accuracy; 2) Getting unlabeled data is very cheap.
- **Weak supervision.** Start with unlabeled data, without ground truth labels; then, one or more weak supervision sources (e.g., heuristics to automate labeling); then, learn a generative model to determine the weights for these weak supervision sources (in this case, the heuristics).