**Machine Learning System Architecture and Why it Matters**

**Specific Challenges of ML Systems**

* Reproducibility may be used for:
  + Research
  + Model improvement
  + Audits
  + Regulatory reasons.
* Entanglement: “The changing anything changes everything principle”
  + Example: If we have an input feature which we change then the importance weights or use of the remaining features may all change as well.
* Data dependence: The biggest difference between machine learning pipeline and traditional web apps. We input both code and data, not only code. Data changes over time
* Configuration issues: There’s the need for incrementing models and experimenting. Not only that, but the configuration has to be flexible to make it easier to compare two models with different configurations.
* Data and feature engineering (preparation): There’s a lot of supporting code to get data into and out of the expected formats.
* **Separation of expertise:**
  + DevOps: Doing the deployments
  + Business: Determines what the requirements are
  + Software engineering: Putting the models into applications
  + Data Science: Develops the model
* **Reproducibility: We need to have the ability to:**
  + Access the model version
  + The exact data that was used
  + The features
  + The configuration
  + Third party libraries
  + Infrastructure
  + …

**Machine Learning System Approaches**

**General ML Architectures**

* Train by batch, predict on the fly, serve via REST API
  + Example: A model trained and persisted offline then loaded into a web application that can give real time predictions about the price of a house
* Train by batch, predict by batch, serve through a shared database
  + Example: A model trained and persisted offline then loaded into a web application where the user uploads a CSV file of houses and then wait 30 minutes for an email telling them to check the website for results
* Train, predict by streaming
  + Example: A model that receives a queue of houses and updates the model easily
* Train by batch, predict on mobile (or other client)
  + Example: An IOS app using the core ML framework that would not need to call a backend service to make a prediction. Instead the predictions could be made on the device

**Architecture Comparison**

* Train by batch, predict on the fly, serve via REST API
  + Pros:
    - Predictions on the fly
    - Easy for A/B test
  + Cons:
    - We are not able to use a slow algorithm
    - It’s complex to scale
* Train by batch, predict by batch, serve through a shared database
  + Pros:
    - Easy to use different system for the front end and batch system (even different languages)
    - Easier to manage model version and prediction results
    - Can use slow and complex algorithm
  + Cons:
    - Predictions are not on the fly
* Train, predict by streaming
  + Pros:
    - Predict on the fly
    - We can update the model interactively
  + Cons:
    - Requires a complex infrastructure
* Train by batch, predict on mobile (or other client)
  + Pros:
    - Predict on the fly
  + Cons:
    - Limited in the number of algorithms we have available to use

**Architecture Component Breakdown**

* Breakdown of the offline training phase
  + Training Data: Applications to load the data
  + Feature Extractor: Applications to create features
    - Example: Generating sentiment analysis features based on input text data
  + Model Builder: Applications that serialize and persists trained models, versioning and ensuring that the model is a suitable format for deployment
* Breakdown of the prediction phase
  + Clients send requests to our REST API, we perform feature engineering and give a prediction
* System Diagram (System Architecture)
  + Left: A representation of our application code
  + Right:
    - Our application code can be converted into docker images
    - We can persist our trained model to file services such as Amazon S3
    - We ensure that our code is managed effectively by using versioning software like GitHub
  + Middle:
    - (CI/CD = Continuous Integration/Continuous Deployment)
    - Includes test and deployment
  + Bottom:
    - (PaaS) We deploy our applications to either cloud platforms such as Heroku
    - (IaaS) Or to our own configured cloud infrastructure such as the AWS elastic container service
    - Then we are able to serve our predictions by our REST API as requests come in from clients

**Building a Reproducible Machine Learning Pipeline**

* Importance:
  + Financial costs
  + Lost time
  + Lost reputation
  + Difficult to determine if a new model is truly better than the previous one
* We need to make sure that all steps circled in red are reproducible both in our research environment and in our deployment models
* Ensuring reproducibility during data gathering:
  + Use some order in SQL to load the data (like using order by)
  + Snapshot:
    - Cons:
      * New regulations on the use of data may not allow you to store data somewhere else other than in the source
      * Data may be too big to store
  + “Not in house already” means that it is not implemented in your company