Federated Learning

Motivation

Federated learning (FL) is used by **Gboard**. Google plans to use it in more apps in the future.

FL has the potential to become a standard way to train models. **Systems are needed to make this possible**.

FL a good case study of how to build large systems for ML.

Agenda

Overview of Applied ML

Problem and Design

Techniques

Design Exercises

Q/A or Discussion

Check for Understanding: True/False

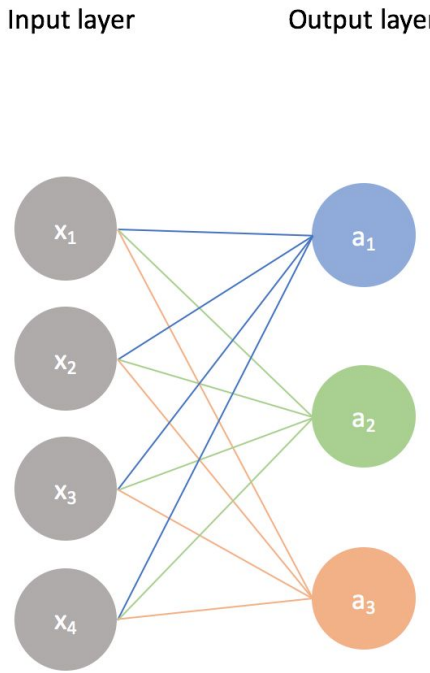
The 3 phases of a round are *selection*, *configuration*, and *reporting*.

A *device* will download the *global model* every *round* that it is *selected*.

The *server* must wait for the slowest *device* before starting the next training *round*.

Part 1: Overview of Applied ML

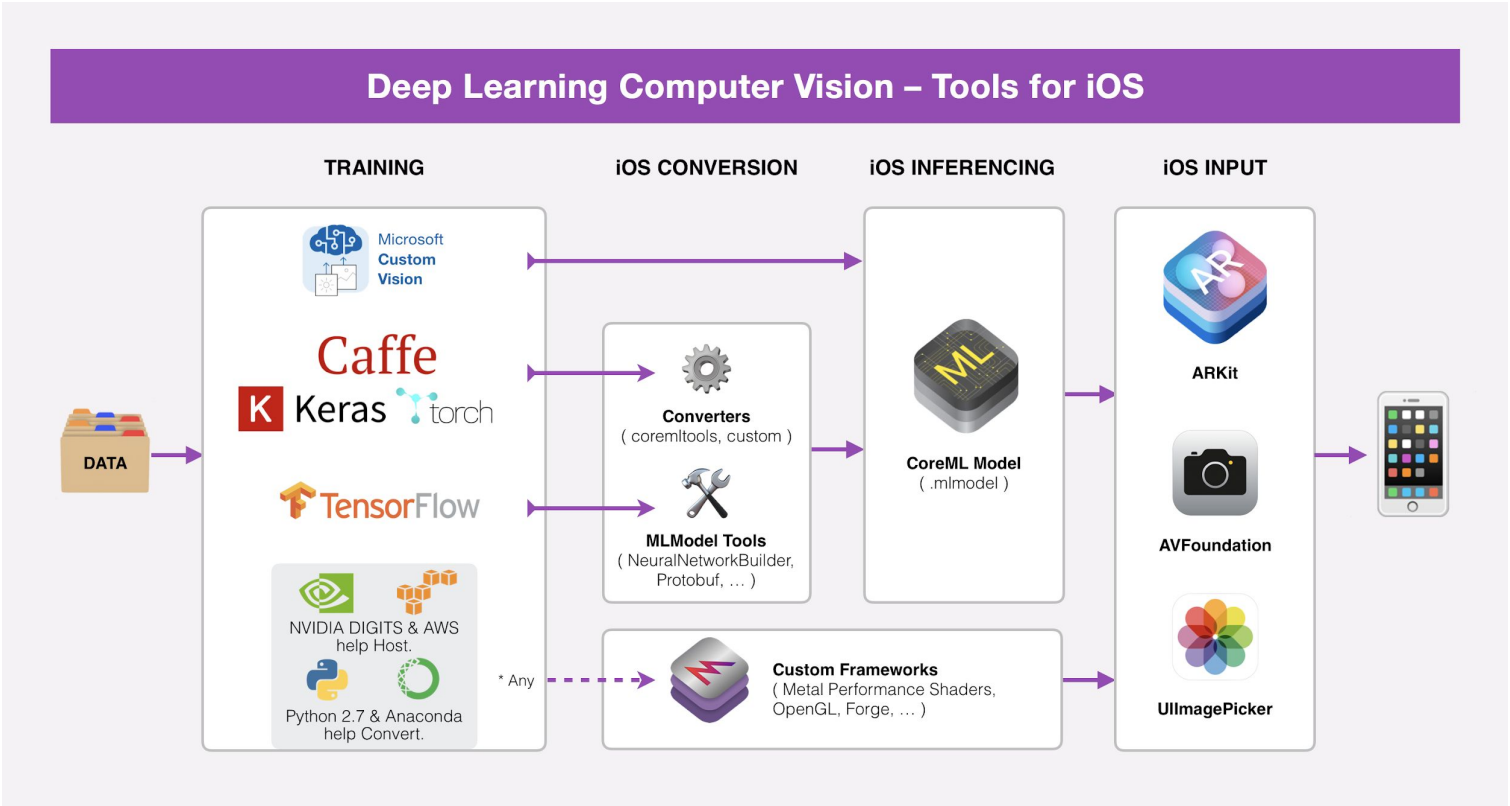
Question: What are the (model) parameters?



A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

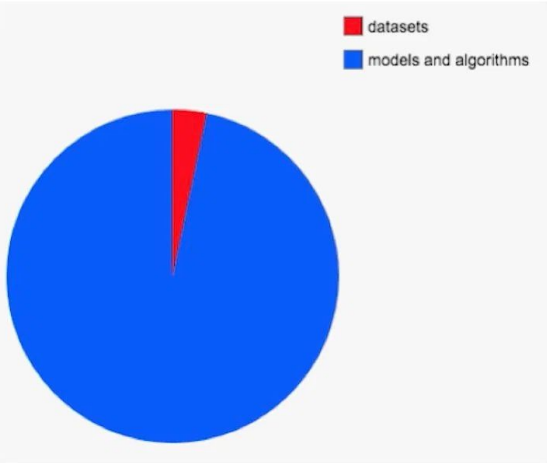
The Mobile ML Development Workflow



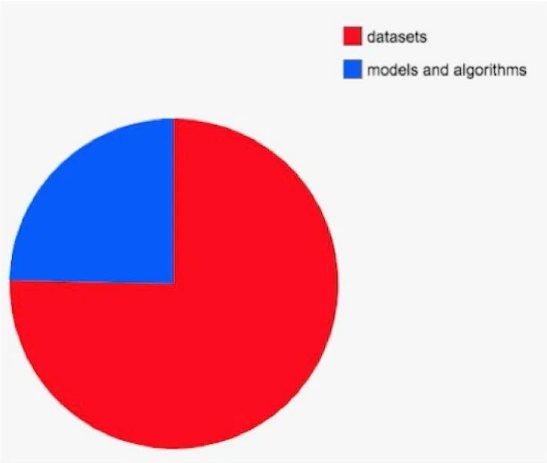
The Problems ML Engineers Face I

Amount of lost sleep over...

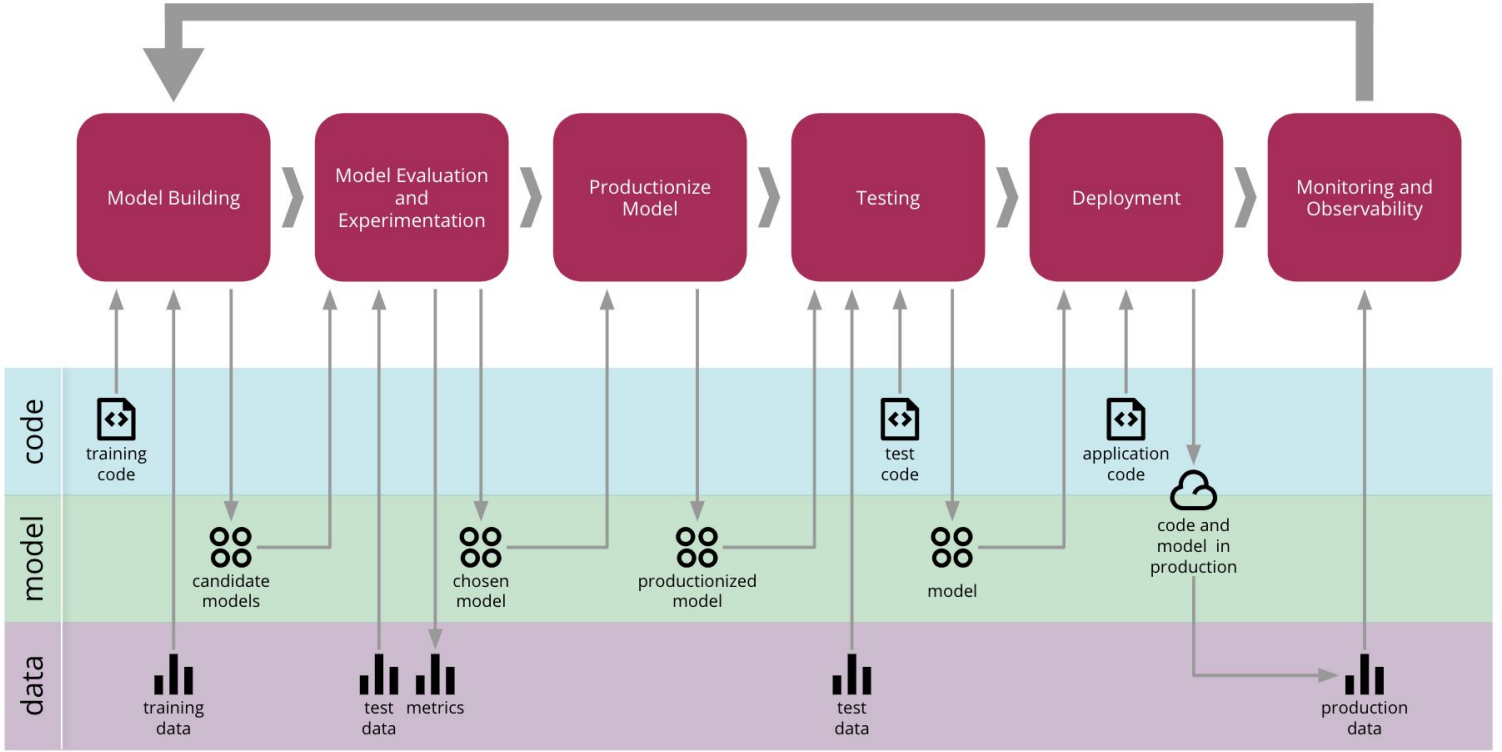
PhD



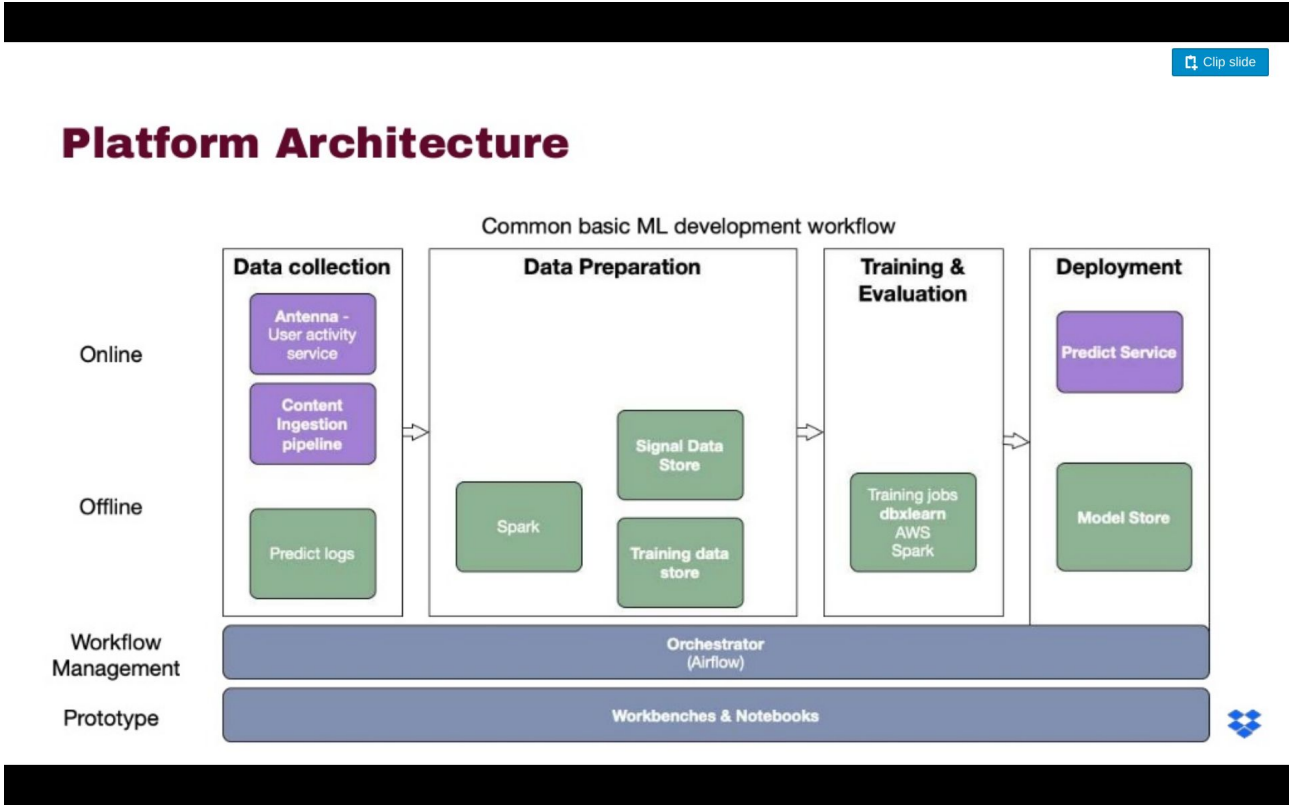
Tesla



The Problems ML Engineers Face II



Dropbox's ML Platform



Some Use Cases of ML

Snapchat Filters

Siri

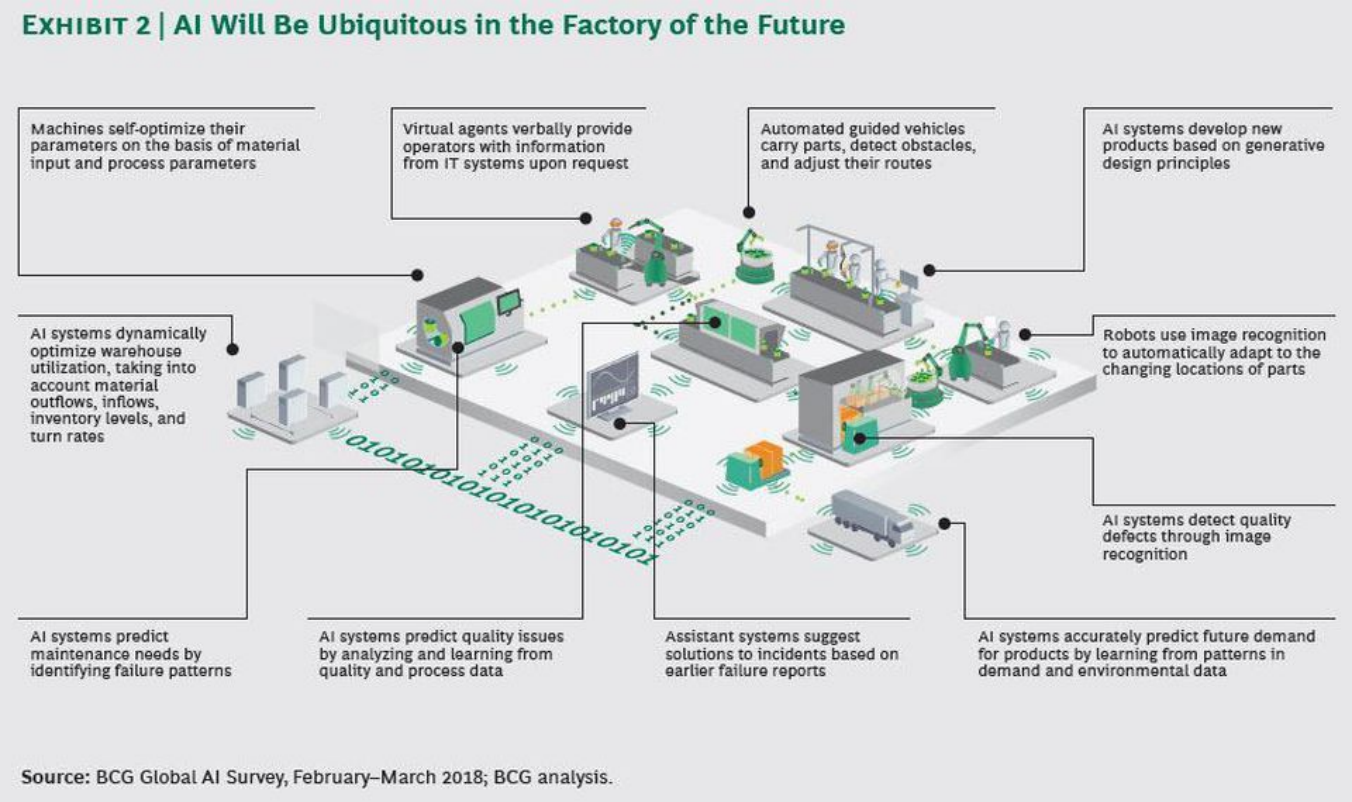
ATM Facial Recognition

Predictive Maintenance (in Manufacturing)

Database Tuning

Taxi Route Scheduling

Use Cases at the “Edge”



Part 2: Problem and Design

How is federated learning more ‘private’?

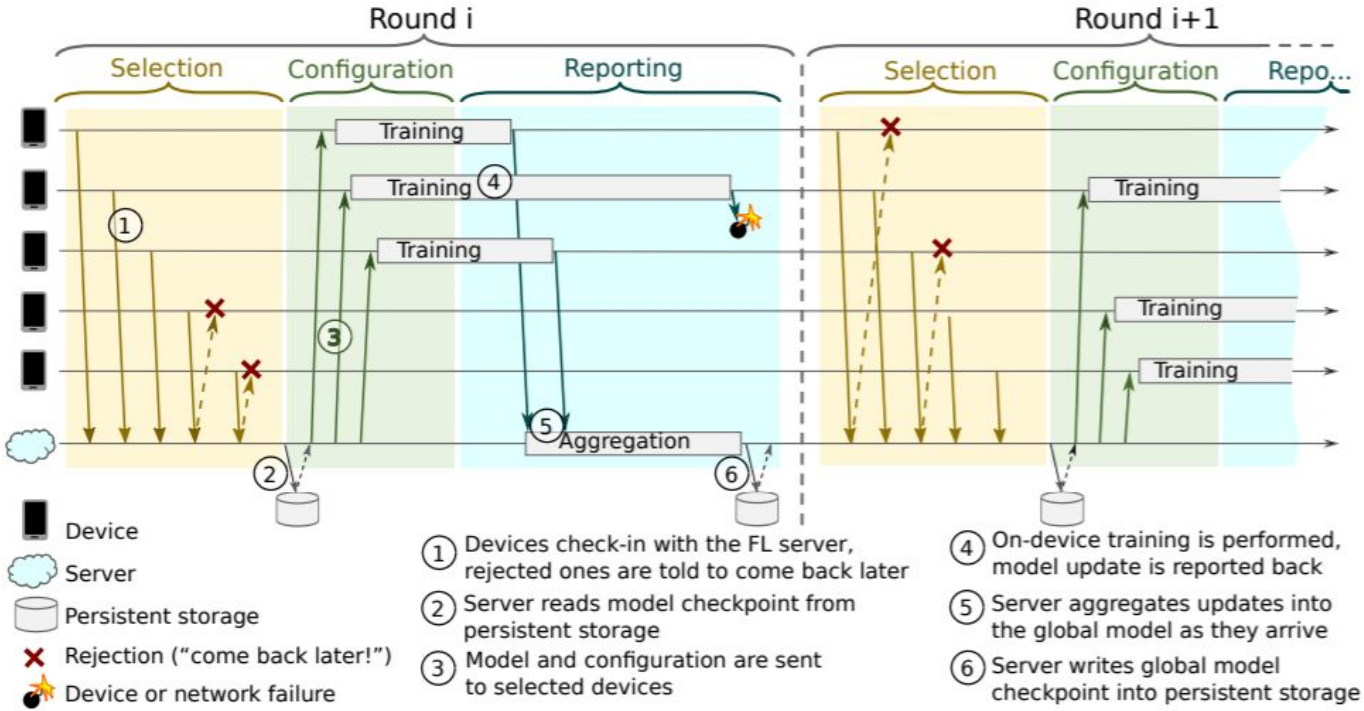


Figure 1: Federated Learning Protocol

In general, why is federated learning hard?

In general, why is federated learning hard?

Algorithm

Network performance

Fault tolerance

Differential privacy

Concurrency

Malicious clients*

Part 3: Techniques

The Many Different Papers Published

- Federated Optimization: Distributed Optimization Beyond the Datacenter (Oct 2015)
- Communication-Efficient Learning of Deep Networks from Decentralized Data (Feb 2016)
- Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data (April 2017)
- Practical Secure Aggregation for Privacy-Preserving Machine Learning (October 2017)
- Federated Learning for Mobile Keyboard Prediction (Feb 2019)
- Towards Federated Learning at Scale: System Design (March 2019)

Key Points of The Old Paper

Structured updates and *sketched updates* are two **lossy data reduction** methods to reduce network costs associated with federated learning by 10-100x (compared to sending back all 10^6+ parameters (4-5 MB*) from the chosen subset of clients every round)

Structured updates: lossy reduction of model updates using “structure”

- *Random Mask (works better):* Zero out (aka mask off) a large fraction of the parameter matrix. Generate a random mask every round. Lose <1% **compared to ‘lossless’ single-machine training** in accuracy while taking 16x less traffic. Takes <8 GB network traffic to fully train on 50k 32x32 images (CIFAR 10).

Sketched updates: lossy compression of model updates before sending to server

- Combine subsampling, quantization, and rotation. Lose **5-6% in accuracy** while costing **256x less** network traffic. Takes ~1 GB to converge on CIFAR 10, 30 GB on the Reddit dataset.

Core Federated ML Techniques

Federated Optimization (Distributed SVRG)

Federated Averaging

Secure Aggregation

(Opinionated) Takeaways about Federated Learning

Federated Learning performs well in the following conditions:

- **You want to build ML models on sensitive data like user messages**
- You don’t need to manually label the data
- Your model is not large (<10^8 parameters)
- Your dataset is not that large (data fits on the device)
- You can tolerate 1-10% losses in model accuracy
- You do not need to retrain models frequently, quickly, or concurrently*

Federated ML replaces single-node server-side training on sensitive data. Distributed training on parameter servers target an entirely different use case: training models quickly on up to billions of parameters and up to PBs of data.

Exercise 1: Life of a Training Example

Exercise 2: Design a Development Platform for FL

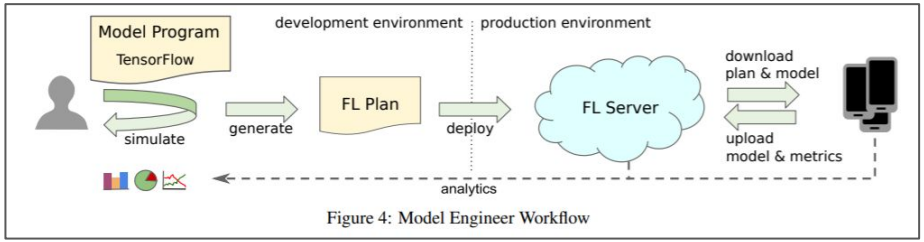


Figure 4: Model Engineer Workflow