# MLOps

Justin Post

• Big picture of dealing with big data: it's complicated!



#### NC STATE UNIVERSITY

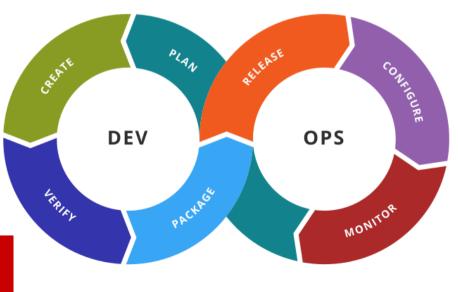
• Big picture of dealing with big data: it's complicated!



- Data pipeline
  - Is the data valid?
  - Garbage in garbage out!
- ML pipeline
  - Model performing well?
  - Still valid with new data or refit needed?
- How do others use our model???

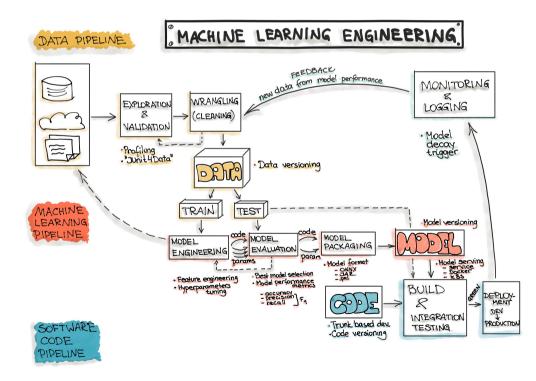
#### NC STATE UNIVERSITY

- Good software is hard to build quickly
- DevOps is a framework for software development and deployment
  - Automation of the software development lifecycle
  - Collaboration and communication
  - Continuous improvement and minimization of waste
  - Hyperfocus on user needs with short feedback loops



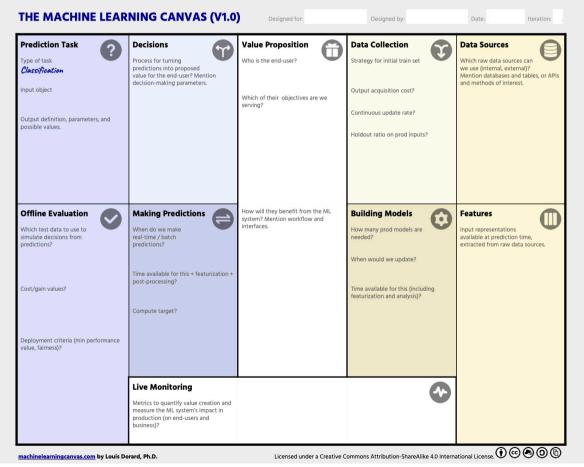
#### NC STATE UNIVERSITY

- Similar ideas have arisen when implementing ML models (especially on big data)
- ML-Ops is a framework for the entire ML development/deployment process
  - These notes are almost entirely distilled from their material!



# MLOps to Solve a Problem

#### Good read from "Value Proposition" on!



# MLOps Concepts

- Models really useful when they make reasonable predictions and are available to the 'core software system'
- Models should be 'first-class citizens'
- Must continually monitor and update models (three levels of change)
- Testing of models should be automated

<text><section-header><section-header>

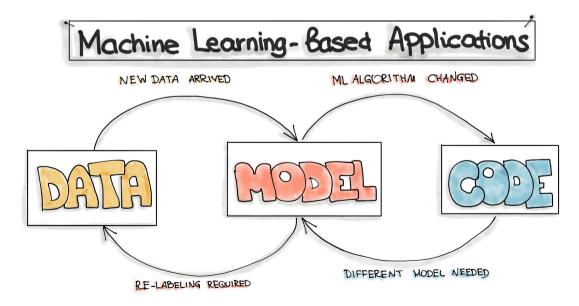
# **MLOps Evolution**

The Evolution of #MLOps			
Proprietary Inference Servers	The Rise of Open Source Data Science Tools	Containerization to-the-rescue	"MLOps Platforms"
using proprietary tools to perform modeling and inference • SAS • SPSS • FICO	<ul> <li>attempt to wrap the data science stack in a lightweight web service framework, and put it into production</li> <li>Python: <ul> <li>SciPy stack</li> <li>scitkit-learn</li> <li>TensorFlow etc.</li> </ul> </li> <li>R: <ul> <li>dplyr</li> <li>ggplot2</li> <li>etc.</li> <li>Spark, H2O, others</li> </ul> </li> </ul>	Containerization of the "Stone Age" approach, making it easy to scale, robust, etc.	<ul> <li>Dockerized open- source ML stacks</li> <li>Deployed them on- premise or in the cloud via Kubernetes</li> <li>and providing some manageability ("ML Ops").</li> </ul>
Pre-History Age	2000 20 Stone Age	)15 2 Bronze Age	2018 MLOps Gold Rush Age
Sketch: @visenger		Source	ce: bit.ly/mlops-evolution

# Three Main Processes of ML Deployment

Build model on data you collect to make predictions, classifications, recommendations, etc.

- Three main phases, each must be monitored (again taken from ml-ops.org!)
  - 1. Data Engineering: data acquisition & data preparation
  - 2. ML Model Engineering: ML model training & serving
  - 3. Code Engineering :integrating ML model into the final product



# Data Engineering

Usually create a *Data Engineering Pipeline*:

- Must integrate data from many source
- Data cleaning, imputation, and validation must be done
- Data splitting

Generally takes the longest time and most resources to do this part!

# ML Model Engineering

**Model Engineering Pipeline** generally has a few steps:

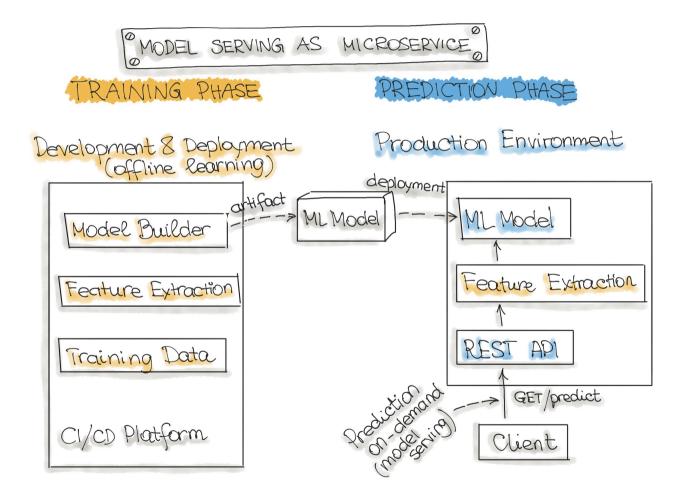
- Model Training
  - Including feature engineering and the hyperparameter tuning
- Model Evaluation
  - Ensure it meets predetermined standards
- Model Testing
  - On the holdout dataset
- Model Packaging
  - Exporting the final ML model to be used by a business application
  - See the "Model serialization formats" section

# Code Engineering

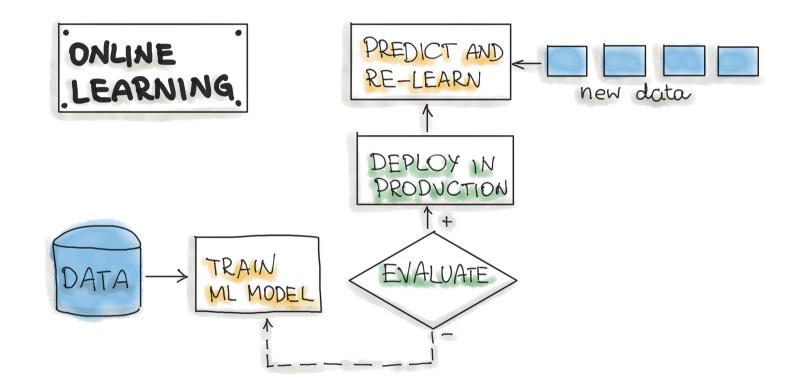
**Deployment pipeline** involves things like:

- Model Serving
  - Using the model in some software
- Model Performance Monitoring
  - Making sure the model is still performing ok on new data
- Model Performance Logging
  - Every time the model is used you log it

### Models Built on Batch Data



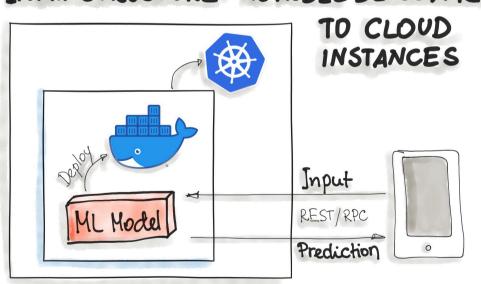
# Models Based on Streaming Data



# **Deployment Strategies**

Two common ways for deploying models:

- As Docker Containers to Cloud Instances
- As Serverless Functions



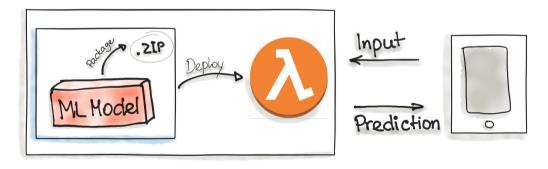
# INFRASTRUCTURE: HI HODEL DEPLOYMENT

# **Deployment Strategies**

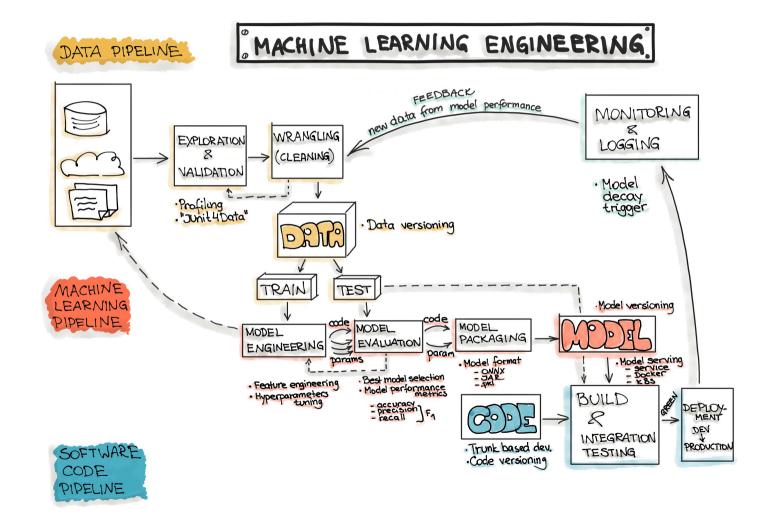
Two common ways for deploying models:

- As Docker Containers to Cloud Instances
- As Serverless Functions

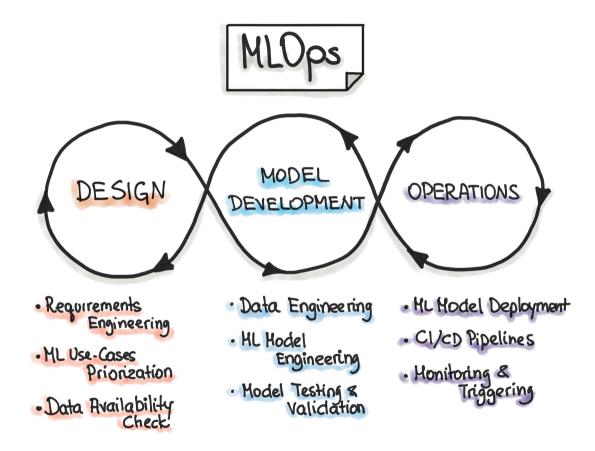
#### INFRASTRUCTURE: HL HODEL DE PLOYHENT AS SERVERLES FUNCTION



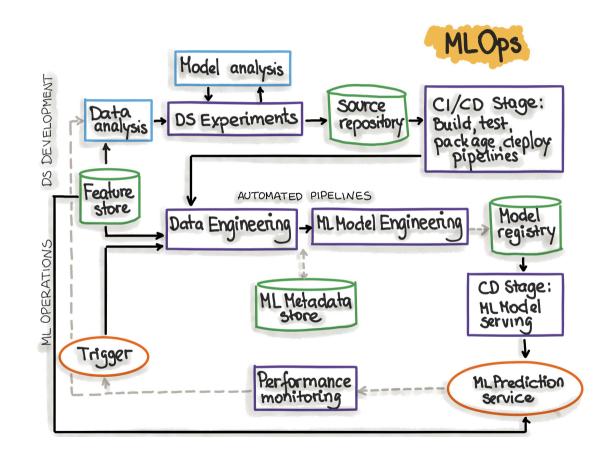
# **Big Picture Workflow**



#### **Iterative Process That Must Be Monitored**



### **Automated MLOps Pipeline**



# Important Components to Consider

#### Entire Section Worth Reading

- Source Control: Versioning the Code, Data, and ML Model artifacts
- Test & Build Services: Unit tests and building of model to be deployed
- Model Registry: Registry for storing already trained ML models
- Feature Store: Preprocessing input data as features to be consumed in the model training pipeline and during the model serving
- ML Metadata Store: Tracking metadata of model training, for example model name, parameters, training data, test data, and metric results.
- Reproducibility

# Recap

MLOps provides a framework to efficiently include ML models within a business application

- Three main phases, each must be monitored (again taken from ml-ops.org!)
  - 1. Data Engineering: data acquisition & data preparation
  - 2. ML Model Engineering: ML model training & serving
  - 3. Code Engineering :integrating ML model into the final product