Model Deployment, Model Versioning, Working Environments (Dev -> Staging -> Prod)

Marco Morales
marco.morales@columbia.edu

Nana Yaw Essuman
nanayawce@gmail.com

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2021
Columbia University

model deployment

what happens after you've built your model?

Often, data scientists end their model development process after they have achieved an acceptable accuracy and are able to produce a graph for a presentation, and "ta-da! Mission Accomplished."

sometimes, this happens...

Or, this happens...

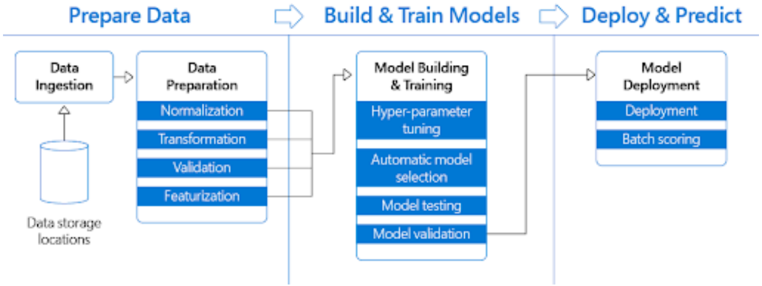
In [8]:
```
ml_model.say_something()
```

```
I am all alone
I was never deployed :(
Was my F1 score not that great?
```

what is model deployment?

Model deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the machine learning life cycle and can be one of the most cumbersome.

ML Lifecycle



types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

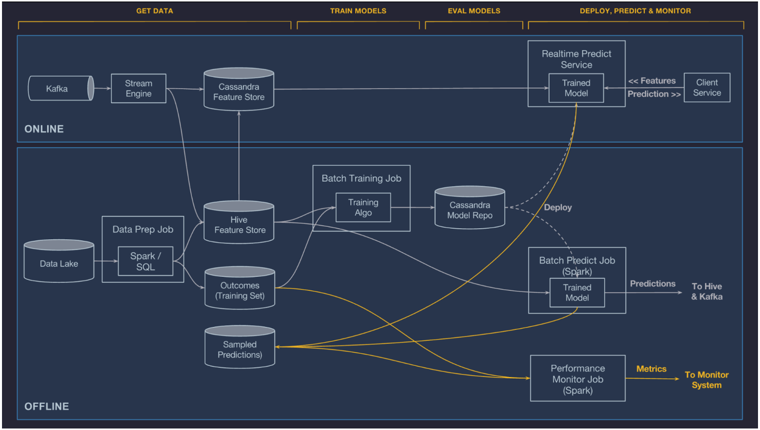
types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

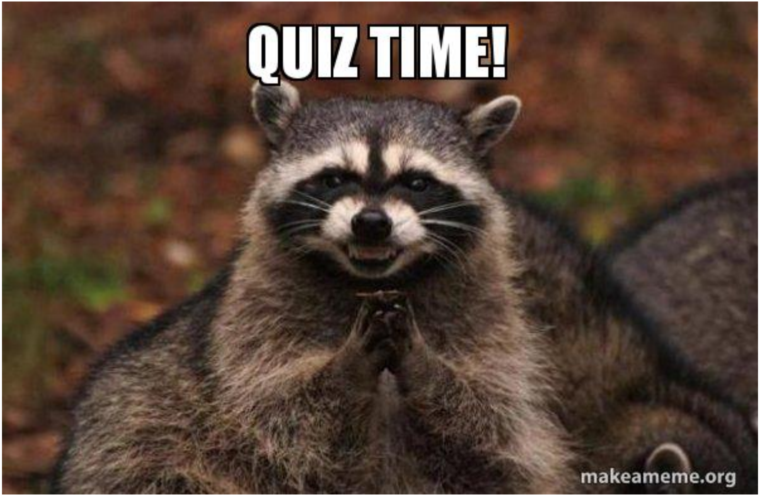
types of model deployment

- batch (offline inference)
  - looser time constraint but makes many more predictions
  - saves predictions to data storage (relational database or NoSQL database)
  - automated and usually triggered by a batch data ingestion or typically generated on some recurring schedule (e.g. hourly, daily)
- real-time (online inference)
  - return single predictions in less than a second
  - allows us to make predictions for any new data e.g. generating recommendations for new users upon signing up
  - model object often containerized using Docker to preserve ML environment
  - exposed through a REST API

ML Pipeline


GET DATA
TRAIN MODELS
EVAL MODELS
DEPLOY, PREDICT & MONITOR
Kafka
Stream Engine
Cassandra Feature Store
Realtime Predict Service
Trained Model
<< Features
Prediction >>
Client Service
ONLINE
Batch Training Job
Training Algo
Cassandra Model Repo
Deploy
Data Prep Job
Spark / SQL
Data Lake
Hive Feature Store
Outcomes (Training Set)
Batch Predict Job (Spark)
Trained Model
Predictions
To Hive & Kafka
Sampled Predictions)
Performance Monitor Job (Spark)
Metrics
To Monitor System
OFFLINE

Pop Quiz!!!



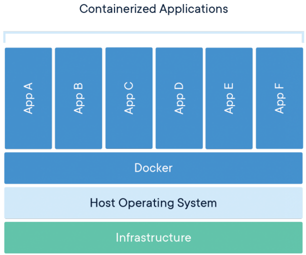
what is a container?

what is a container in cloud computing?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a:

- lightweight
- standalone
- executable package of software

that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

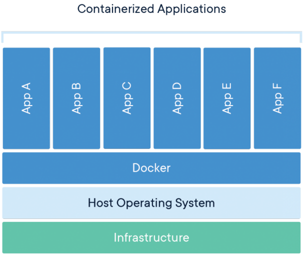


what is a container in cloud computing?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a:

- lightweight
- standalone
- executable package of software

that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

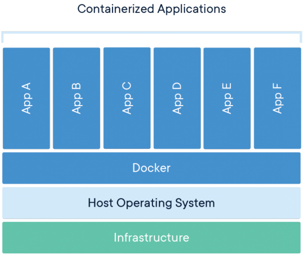


what is a container in cloud computing?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a:

- lightweight
- standalone
- executable package of software

that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.



model versioning

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: A Data Science Project

- Three **aims** of a data science project

  a) **reproducibility**
     - anyone should be able to arrive to your **same results**

  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine

- crucial tenets for **collaborative work**

  c) **scalability**
     - your project should also work for larger data sets and/or be on the path of automation

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

RECAP: A Data Science Project

- Three **aims** of a data science project
  a) **reproducibility**
     - anyone should be able to arrive to your **same results**
  b) **portability**
     - anyone should be able to **pick up where you left off** on any machine
- crucial tenets for **collaborative work**
  c) **scalability**
     - your project should also work for **larger data sets** and/or be on the path of **automation**

what is model versioning?

Machine learning is about rapid experimentation and iteration, and without keeping track of your modeling history you won't be able to learn much. Versioning lets you keep track of all of your models, how well they've done, and what hyperparameters you used to get there.

many paths to the top

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

what should be versioned?

- environment and infrastructure variables
- code
- libraries/packages version
- data used for training
- data used for validation
- other parameters used in model development

Model Tracker Example

		VERSION						RESULTS	
Model		Code (architecture)	Dataset	Data (training)	Data (validation)	Other Parameters	Data (testing)	Accuracy	% False Positives
Model	1	Alpha	pets	1.0	1.0	1.0	1.0	80%	10%
Model	1.1	Alpha	pets	1.1	1.0	1.0	1.0	75%	12%
Model	1.2	Alpha	pets	1.2	1.0	1.0	1.0	85%	8%
Model	1.3	Alpha	pets	1.2	1.1	1.0	1.0	85%	9%
Model	1.4	Alpha	pets	1.2	1.1	1.1	1.0	86%	6%
Model	1.5	Beta	pets	1.2	1.1	1.1	1.0	90%	6%
Model	1.6	Inception	pets	1.2	1.1	1.1	1.0	84%	10%
Model	1.7	Beta	pets	1.2	1.1	1.2	1.0	91%	5%

ML Model Tracking and Versioning DEMO

why is it important?

- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

why is it important?

- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

why is it important?

- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

why is it important?

- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

why is it important?

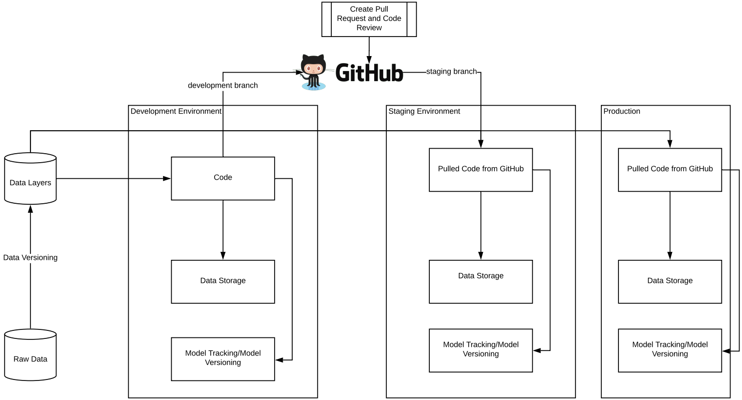
- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

why is it important?

- finding the best model
- failure tolerance
- movement between various working environments
- debug effectively
- provides transparency about what produced the model
- allows for faster experimentation

working environments

Working Environments



Model Deployment, Model Versioning, Working Environments (Dev -> Staging -> Prod)

Marco Morales
marco.morales@columbia.edu

Nana Yaw Essuman
nanayawce@gmail.com

GR5069
Topics in Applied Data Science
for Social Scientists

Spring 2021
Columbia University