DevOps and Machine Learning

Jasjeet Thind
VP, Data Science & Engineering, Zillow Group
@JasjeetThind
Agenda

Overview of Zillow Group (ZG)

Machine Learning (ML) at ZG

Architecture

DevOps for ML
Zillow Group Composed of 9 Brands

Build the world's largest, most trusted and vibrant home-related marketplace.
Machine Learning at ZG

- Personalization
- Ad Targeting
- Zestimate (AVMs)
- Premier Agent (B2B)
- Mortgages
- Deep Learning
- Demographics & Community
- Business Analytics
- Forecasting home price trends

@JasjeetThind
High Level Architecture

- ML WEB SERVICES
- ANALYTICS
  - DATA SCIENCE
  - DATA LAKE
  - MACHINE LEARNING
  - DATA COLLECTION
  - DATA QUALITY

@JasjeetThind
DevOps & ML at ZG

Tight collaboration, fast iterations, and lots of automation.

CONTINUOUS INTEGRATION (CI)  CONTINUOUS DELIVERY (CD)

Research  Dev  Test  Ship  ML Workflow  Real-time ML

@JasjeetThind
Research

Data scientists prototype features and models to solve business problems

Models

- Random Forest
- Gradient Boosted Machines
- CNN / Deep Learning
- NLP / TF-IDF / Word2vec / Bag of Words
- Linear Regression
- K-means clustering
- Collaborative Filtering

@JasjeetThind
DevOps Challenges in Research

(1) Data scientists iterate through lots of prototype code, models, features and datasets that never get into production

(2) How do data scientists validate experiments on production system?
How to apply DevOps to experiments?

Check-in prototypes

- Organize around research area
- Document heavily. Experiments often revisited
- Check-in subsets of training & scoring datasets

For large training datasets, store in the data lake

- Pointers to data in README.md
DevOps enables experiments in production

Build tools for scientists to package and deploy ML models and features into production A/B tests

- Once submitted via the tool the model goes through the same release / validation pipeline as ML production models go through.

Real world prototyping can be done before production quality ML code.

@JasjeetThind
ML engineers operationalize features and models for scale.

Automatically build and upload artifacts
DevOps ML Challenges in Dev

1. Unit tests non-trivial as model outputs not deterministic
   
   \[
   \text{output} = f(\text{inputs})
   \]

2. CI requires fast builds. But accurate predictions require large training data sets, and training models is time consuming.
**Unit Tests w/ ML**

Don’t test “real” model outputs.

- Leave that to integration / regression tests
- Focus on code coverage, build and unit tests are fast.

Train models w/ mock training data

- Test schema, type, # records, expected values

Predictions on mock scoring data

```python
// load mock small training dataset
training_data = read("training_small.csv")

// model trained on mock training data
model = train(training_data)

// schema test
assertHasColumn(model.trainingData, "zipcode")

// type test
assertIsInstance(model.trainingData, Vector)

// # of records test
assertEqual(model.trainingData.size, 32)

// expected value test
assertEqual(model.trainingData[0].key, 207)

// load mock small scoring dataset
predictions = score(model, scoring_data)
```
Mock training data facilitates fast builds

Model prediction accuracy tested in the Test phase
After artifact uploaded by build process, deploy and auto-trigger its regression tests

- DevOps API
- Jenkins Deployment
- Aiida
- AWS EMR Regression Cluster

Upload new job

Kick off regression tests after deployment
DevOps ML Challenges in Test

(1) Training and scoring models can take many hours or more
(2) Models have very large memory footprint
(3) Regression tests non-trivial as model outputs not deterministic
(4) Real time model outputs vs that of backend systems
How to train and score in Test?

For regression testing, create “golden” datasets, smaller subsets of training and scoring data, for training and scoring models.

Makes training and scoring time much faster
Reducing Model Memory Footprint in Test

Model dynamically generated using golden datasets

Model itself not in GIT, treated as binary
Regression Tests and ML

Adjust with fuzz factor

- May need to run multiple times and average results

Model validation tests to ensure model metrics meet minimum bar

- More on this later

```python
// load "real" subset training data from data lake
training_data = read("training_golden_dataset.csv")

// model trained on golden training data
model = train(training_data)

// generate model output
scoring_data = read("scoring_golden_dataset.csv")
predictions = score(model, scoring_data)

// fuzz factor
id = 5
zestimate = predictions[id]
assertAlmostEqual(zestimate, 238000, delta=7140)
```
Verify Real-time Matches Backend

Real-time machine learning services thousands / millions of concurrent requests

// get model output from web service API for scoring row
id = 5
prediction1 = getFromMLApi(id)

// get model output from batch pipeline for same scoring row
prediction2 = getFromBatchSystem(id)

// get model output from near real time pipeline for same scoring row
prediction3 = getFromNRTSystem(id)

// verify the values are equal
assertAlmostEqual(prediction1, prediction2, prediction3, delta=7140)
SHIP

Automated deployment of ML code to production
DevOps ML Challenges in Ship

(1) Deploy starts during model training and scoring

(2) Backward compatibility issues with code and models for real-time ML API
4 Ways to Handle Deploy during Training

Terminate ML workflow and start the deploy

Wait for ML workflow to finish and start the deploy

Deploy at certain times (e.g. once per day at midnight)

Deploy to another cluster, shutdown the previous cluster
Shipping real-time ML services

Complete successful run of batch pipeline.

- Then ship code and models to web services hosts.
- More on deployment to web services hosts later
ML Workflow

Production ML backends generate models and predictions.

- Raw Data Sources
- Data Collection
- Generate datasets and features
- Model Learning
- Scoring
- Evaluate Model metrics
- Deployment
DevOps Challenges w/ML Workflow

(1) Data quality given high velocity petabyte scale data continually being pushed into the system

(2) Ensuring models aren’t stale as data continues to evolve

(3) Partial failures due to intermediate model training pipelines failing

(4) Model deployments don’t cause bad user experience
Build Analytical Systems for Data Quality

Monitor

- what is expected data
- outlier detection
- missing data
- expected # of records
- latency

Reports / alerts that drive action

Data Quality Architecture

@JasjeetThind
Automate ML to Keep Models Fresh

Run continuously or frequently

Workflow engine

- Oozie, Airflow, Azkaban, SWF, etc

Logging, monitoring, and alerts throughout the pipeline

@JasjeetThind
Persist Intermediate Train Data for Failures

Breakup ML pipelines into appropriate abstraction layers

Leverage Spark which can recover from intermediate failures by replaying the DAG
Model validation tests to verify quality

Don’t replace existing model if one or more metrics fail. Trigger alert instead.
Replace if all tests pass.

```python
if (mean_squared_error(y_actual, y_pred) > 3) // mean squared error
    alert()
else if (median_error(y_actual, y_pred) > 6) // median error
    alert()
else if (auc(y_actual, y_pred)) // area under curve
    alert()
else
    deploy_models() // all model metrics passed so replace old models
```
Real-time ML

Models need to be deployed into production

Models need to be executed per request with concurrent users under strict SLA
DevOps ML Challenges w/ Real-time ML

1. Deployment of models can take many seconds or minutes

2. Latency to execute ML model for each request and return responses must meet strict SLA (< 100 milliseconds)

3. Need to ensure features at serving time same as backend systems
Automate Deployment of Models

Replicate hosts

During deploy, take host offline, update models, put back into rotation, repeat for replicated hosts.

For small models, do in-memory swap
CPU is Bottleneck for Real-time ML

Real-time feature generation requires data serialization

- Minimize as much as possible
- If C++ code needs to prep data for ML in Python, consider re-writing Python code into C++

Hyperparameters - balance latency and accuracy

- 1000 trees for RF that meets SLA might be a better business decision than 2000 trees with “slight” accuracy gain

@JasjeetThind
Features at serving should equal backend

Save feature set used at serving time and pipe into a log or queue

- Use features directly in training
- Or at least verify they match training sets
Free Zillow Data

Zillow Home Value Index (ZHVI)
- Top / Middle / Bottom Thirds
- Single Family / Condo / Co-op
- Median Home Value Per Sq Ft

Zillow Rent Index (ZRI)
- Multi-family / SFR / Condo / Co-op
- Median ZRI Per Sq ft
- Median Rent List Price

Other Metrics
- Median List Price
- Price-to-Rent ratio
- Homes Foreclosed
- For-sale Inventory / Age Inventory
- Negative Equity
- And many more…

Time Series: national, state, metro, county, city and ZIP code levels

ZTRAX: Zillow Transaction and Assessment Dataset

Previously inaccessible or prohibitively expensive housing data for academic and institutional researchers FOR FREE.

- More than 100 gigabytes
- 374 million detailed public records across more than 2,750 U.S. counties
- 20+ years of deed transfers, mortgages, foreclosures, auctions, property tax delinquencies and more for residential and commercial properties.
- Assessor data including property characteristics, geographic information, and prior valuations on approximately 200 million parcels in more than 3,100 counties.

Email ZTRAX@zillow.com for more information
We’re Hiring!!

Roles

● Principal DevOps Engineer
● Machine Learning Engineer
● Data Scientist
● Product Manager
● Data Engineer

Related Blogs

● Zillow.com/data-science
● Trulia.com/blog/tech/
Zillow Prize - $1 million dollars

The brightest scientific minds have a unique opportunity to work on the algorithm that changed the world of real estate - the Zestimate® home valuation algorithm.

www.zillow.com/promo/zillow-prize