Deploying and Evaluating Data Products

https://db.tt/JIYOoiPu

Josh Levy, PhD
About Vast

Data and analytics for considered purchases (vehicles, homes, ...)

Full – Stack Analytics

Insight Layer
(Domain specific applications, Actionable Insights)

Analytics Layer
(Statistical models, Algorithms, Machine Learning)

Data Layer
(Hadoop, NoSQL, Analytics Databases)

White Label Marketplaces, Market Reports, Sales Apps

Pricing, Supply, Demand, Recommendations, Behavior

Inventory (rapid churn), Consumer Behavior

Graphic: Chip Hazard (Flybridge Capital Partners)
http://www.kdnuggets.com/2014/05/stacking-deck-next-wave-opportunity-big-data.html
Maturity levels for turning "models" into "products"

1. Can we deploy one model into a production environment?
2. Given two models that perform similar functions can we evaluate which is better?
3. Can we operationalize model training?
Goals for Mature Data Products

New version of a model automatically
● trained
● deployed
● evaluated

Traffic automatically routed to top performer
Outline for this talk

1. Can we deploy one model into a production environment?
2. Given two models that perform similar functions can we evaluate which is better?
Deploying models is hard

Conway's Law helps to explain why

*Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations*
Vast has put together a Data Science team that thinks about training and validating models, running experiments.
Models don't exist in isolation.

Add value when exposed in a product.
Models don't exist in isolation.

Add value when exposed as a product.
Communication barriers between these teams
Simplified Problem

Expose models as services internal apps and external customers can access.
Data Science and Engineering collaborate to build Analytics Layer.
POST /price/1/auto/listing-price-drivers

Retrieve price drivers for the auto listing in the POST body.

Query Parameters

- **modelName** (optional) the name of the analytical model to be used [default]
- **includeModelContext** (optional) whether or not to include model context in the response [true, false] 
- **callerId** (optional) a user-friendly identifier for the originating application 
- **partnerId** (optional) a user-friendly identifier for the originating partner

Samples

Request JSON
Response JSON

analytics-service-server-5.8.0-86652

```json
{
  "listing": {
    "sample-record-id": "",
    "city": "Austin",
    "state": "TX",
    "county": "Travis",
    "zipCode": "78703",
    "region": "US",
    "latitude": 30.2713,
    "longitude": -97.7436,
    "advertiser": "rover.ebay.com",
    "price": 36000,
    "year": 2010,
    "make": "Honda",
    "model": "Accord",
    "trim": "EX",
    "mileage": 49999,
    "condition": "Used",
    "bodyStyle": "Sedan",
    "transmission": "Automatic",
    "fuel": "Gasoline",
    "driveType": "FWD",
    "carfax": "false",
    "exteriorColor": "Black",
    "cpp": "false",
    "engine": "4 Cyl",
    "oneOwner": "false"
  }
}
```
Exposing Models as Services

Communications challenges between scientists and engineers

- Human language & concepts
- Technology platforms
Communication between Humans

Looking for common ground between

Data Scientists thinking about Experiments, Training and Validation

Engineers thinking about Scalability, Deployment, Reliability, and Monitoring
Technology Platforms

Back-end engineering

- Typically JVM (Scala)
- Apps and other platforms can call JSON over HTTP services
Technology Platforms

Data Science: "Use the most comfortable tool for the job."

- Typically Python (sklearn) or R for models trained on inventory (millions of rows)
- Hadoop (scalding or streaming) when working with user behavior, systems exhaust
Most comfortable tool for the job

DS team started CS PhD-heavy, DIY mindset

Growth from MS in Business Analytics program

- Strong stats background, productive in R
- Different hiring standards than back end engineering
Expectations for MS in Business Analytics

We're learning to push "Data Janitor" work onto engineering team.
Assuming MSBAs take clean data in, and produce models.
Those models need to become products, but MSBAs don't need to write production scala.
Answers from Vast in chronological order
1. Rewrite scoring function for JVM
2. Export as PMML
3. Expose as JSON over HTTP or WebSockets from Python or R
1. Rewriting Scoring Function

**Pro**
- DevOps is a freebie
- Additional implementations of an interface can be cheap

**Con**
- First version of a model is expensive
  - Back to Conway's law - difficult communication
- Worry about transcription errors
1. Rewriting Scoring Function

Best Practice
Scientist writes code to generate some model representation.

Engineer writes code to read model and score live data; Exposes that code as a service.
2. Export as PMML

**Pro**
- Cheaper, easier than rewrite
  - Scientist generates PMML
  - Engineer plugs in off the shelf runner
- DevOps still free

**Con**
- Limited to doing things that can be expressed in PMML
- Jumped through hoops for feature transformations
PMML Example: Auto Pricing Model

Training: split according to domain knowledge
- (make, model)
- (age, location)
- (completeness of data)

LASSO on each group
- Predicts total price
- Decomposes price by features
Auto Pricing Model

One PMML Decision Tree per make, model

- Internal nodes from manual splits
- ~500 leaf nodes. Each is a regression model.
Market Insights

- **Heated Seats**
  - **about $200 more**
  - Added value of about $200 above vehicles without this feature.

- **Premium Sound System**
  - **about $150 more**
  - Added value of about $150 above vehicles without this feature.

- **One Owner**
  - **about $50 more**
  - Added value of about $50 above vehicles without this feature.
3. **Expose as JSON over HTTP**

**Pro**
- Everyone speaks in their native tongue
- More natural data xform than PMML
- automate this: yhat, Wolfram, Azure

**Con**
- Dev/Ops needs to harden something new
4 lines of boilerplate to wrap python code in YhatModel

2 more lines to deploy model
## WeightedPercentileModel

<table>
<thead>
<tr>
<th>Stage</th>
<th>Version</th>
<th>Language</th>
<th>Last Updated On</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staging</td>
<td>1e89e2a</td>
<td>python</td>
<td>Fri, Oct 03 2014 18:51:24 UTC</td>
<td>online</td>
</tr>
<tr>
<td>Production</td>
<td>1e89e2a</td>
<td>python</td>
<td>Fri, Oct 03 2014 18:51:24 UTC</td>
<td>online</td>
</tr>
</tbody>
</table>

### REST

http://yhat.oak.vast.com/levy/models/WeightedPercentileModel/

$ curl -X POST -H "Content-Type: application/json" \
   --user levy:2eafe8aad9974afe8aad59ff99b9528 \
   --data '{"your data": "goes here"}' \
   http://yhat.oak.vast.com/levy/models/WeightedPercentileModel/

### Websockets
# WeightedPercentileModel

<table>
<thead>
<tr>
<th>Stage</th>
<th>Version</th>
<th>Language</th>
<th>Last Updated On</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Staging</td>
<td>1e89e2a</td>
<td>python</td>
<td>Fri, Oct 03 2014 18:51:24 UTC</td>
<td>online</td>
</tr>
<tr>
<td>Production</td>
<td>1e89e2a</td>
<td>python</td>
<td>Fri, Oct 03 2014 18:51:24 UTC</td>
<td>online</td>
</tr>
</tbody>
</table>

## Example Input

```json
{
  "data": [
    {
      "weight": 1,
      "value": 100
    },
    {
      "weight": 10,
      "value": 50
    }
  ]
}
```

## Example Output

```json
{
  "yhat_model": "WeightedPercentileModel",
  "yhat_id": "c14b5b73-ffa1-4b1e-b6cd-bab2e0cac84",
  "result": {
    "25": 50,
    "50": 50,
    "75": 50,
    "100": 100,
    "00": 0
  }
}
```
Deployed model exists as a service, could be integrated into apps. Our engineers wrap it in another service layer so they can control authentication, logging, ...
RARE

Sunroom
Few houses like this one have this amenity.

Playground
Few houses like this one have this amenity.

Comparisons

3 BEDROOMS
Average
This residential has a typical number of bedrooms as compared to similar properties.

9% SMALLER
Below Average Sqft
This residential's square footage is 9% smaller than similar properties.

18% HIGHER
Above Average Price/Sqft
This residential's price/sqft is 18% higher than similar properties.

10% MORE EXPENSIVE
Above Average Price
This residential is 10% more expensive than similar properties.

AVERAGE
Average Lot Size
This residential's lot has a typical size.

3 BATHROOMS
Average
This residential has a typical number of bathrooms as compared to similar properties.
Deployment Recommendations

New Projects: yhat from the start
Existing Projects: Tempting to continue using PMML or Rewrite

- Want to deploy from python / R as soon as you want to do something new
- But then someone has to support both

Better to deploy directly from training code
Deployment Recommendations

Always name your models. Allow multiple models that perform the same function to coexist.

- Sometimes want different customers on different instances of the model
  - trained on public data only
  - trained on public data + proprietary from customer X

- Allows competition
Outline for this talk

1. Can we deploy one model into a production environment?
2. Given two models that perform similar functions can we evaluate which is better?
Three types of evaluation

Depending on the nature of the model use

1. Direct evaluation against ground truth
2. Indirect evaluation against business metrics
3. Human judgment
Evaluating price predictions against ground truth

Vehicle listings come off the feed with a price.

Given a choice between two models, choose the one that minimizes mean absolute error.
Direct Evaluation Environment

Summary of global measures

Drill-down graphs

Residual graphs
  ● CDF
  ● Drill-down graphs
<table>
<thead>
<tr>
<th>Data/VIN</th>
<th>Prediction Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>JNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>TVSB</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>XNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>YSBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>ZNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>ANBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>BNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>CNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>DNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>ENBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>FNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>GNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>HNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>INBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>JNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>KNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>LNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>MNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>NNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>ONBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>PNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>QNBX</td>
<td>3EE92813212345678</td>
</tr>
<tr>
<td>RNBX</td>
<td>3EE92813212345678</td>
</tr>
</tbody>
</table>

**Links to upstream data for outliers**

**Allows manual investigation**

- Is the model to blame for the outlier?
- Is the data garbage?
Indirect Evaluation

Recommendations on details page

- Can't measure relevance or quality
- Conversion rate is important to business
Human Judgment

Sort in early versions of HomeStory

- Can't measure "relevance"
- Cold start: Not enough traffic to optimize conversion rate yet
- Best we can do is "not embarrassing"
- Be good enough to attract traffic, set up future experiments
We use tools for bulk side-by-side comparisons.
Evaluation Recommendations

Do as much as you can

- Direct evaluation against truth
- Indirect evaluation against business metrics
- Human judgment

Goals

- Data driven decisions
- As much automation as possible
Plugs

Back end engineering team is hiring

Looking for Strong Junior / Senior Java/Scala person

Resumes to Olivier: omodica@vast.com