Who are Data Scientists?
Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.
(((Josh Wills)))
@josh_wills

Data Scientist (n.): Person who is better at statistics than any software engineer and better at software engineering than any statistician.

9:55 AM - 3 May 2012

1,500 1,068
Means: skills vary wildly
But they’re in demand and expensive
“The Sexiest Job of the 21st Century”
- HBR

How many Data Scientists do you have?
At Stitch Fix we have ~80
~85% have not done formal CS
But what do they do?
What is Stitch Fix?
STITCH FIX
STITCH FIX
“PS! Thank you for such an awesome top”
Original Image  →  Encode  →  Compressed Data  →  Decode  →  Reconstructed Image
Two Data Scientist facts:

1. Has AWS console access*.

2. End to end, they’re responsible.
How do we enable this without
Make doing the *right* thing the *easy* thing.
Fellow Collaborators

Horizontal team focused on Data Scientist Enablement
1. Eng. Skills
2. Important
3. What they work on
Let’s Start
Will Only Cover

1. Source of truth: S3 & Hive Metastore  
2. Docker Enabled DS @ Stitch Fix  
3. Scaling DS doing ML in the Cloud
Source of truth:

S3 & Hive Metastore
Want Everyone to Have Same View

- Person A
- File
- Person B

Actions:
- Create
- Read
- Overwrite
- Delete
- Read

Error: Read attempt from Person B to File is blocked by Person A.
This is Usually Nothing to Worry About

- OS handles correct access
- DB has ACID properties
This is Usually Nothing to Worry About

- OS handles correct access
- DB has ACID properties
- But it’s easy to outgrow these options with a big data/team.
S3

- Amazon’s Simple Storage Service
- Infinite* storage
- Can write, read, delete, BUT NOT append.
- Looks like a file system*:
  - URIs: my.bucket/path/to/files/file.txt
- Scales well

* For all intents and purposes
Hive Metastore

- Hadoop service, that stores:
  - Schema
  - Partition information, e.g. date
  - Data location for a partition
Hive Metastore

- Hadoop service, that stores:
  - Schema
  - Partition information, e.g. date
  - Data location for a partition

S3

Subdirectories

../sold_items/

Files

- 20161001/
- 20161002/
- 20161003/
- ...
- 20161031/
  - file001.txt

Hive Metastore:

<table>
<thead>
<tr>
<th>Partition</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>20161001</td>
<td>s3://bucket/sold_items/20161001</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>20161031</td>
<td>s3://bucket/sold_items/20161031</td>
</tr>
</tbody>
</table>
But if we’re not careful

- Replacing data in a partition

S3

../sold_items/
   • 20161001/
   • 20161002/
   • 20161003/
   • 20161031/
   • file001.txt
But if we’re not careful

- Replacing data in a partition
But if we’re not careful
But if we’re not careful

- S3 is eventually consistent
- These bugs are hard to track down
Hive Metastore to the Rescue

- Use Hive Metastore to control partition source of truth

- Principles:
  - Never delete
  - Always write to a new place each time a partition changes

- Stitch Fix solution:
  - Use an inner directory called Batch ID
Batch ID Pattern

S3 Subdirectories
../sold_items/
  20161001/
  20161002/
  20161003/
  ...
  20161031/
T_STAMP/
  file001.txt
Batch ID Pattern

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>20161001</td>
<td>s3://bucket/sold_items/20161001/20161002002334/</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20161031</td>
<td>s3://bucket/sold_items/20161031/20161101002256/</td>
</tr>
</tbody>
</table>
Batch ID Pattern

- Overwriting a partition is just a matter of updating the location

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>20161001</td>
<td>s3://bucket/sold_items/20161001/20161002002334/</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>20161031</td>
<td>s3://bucket/sold_items/20161031/20161101002256/</td>
</tr>
<tr>
<td></td>
<td>s3://bucket/sold_items/20161031/20161102234252</td>
</tr>
</tbody>
</table>
Batch ID Pattern

- Overwriting a partition is just a matter of updating the location
- To the user this is a hidden inner directory

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>20161001</td>
<td>s3://bucket/sold_items/20161001/20161002002334/</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>20161031</td>
<td>s3://bucket/sold_items/20161031/20161101002256/</td>
</tr>
<tr>
<td></td>
<td>s3://bucket/sold_items/20161031/20161102234252</td>
</tr>
</tbody>
</table>
Enforce via API

Person A: Save data to `prod.sold_items` for 2016

API: Write to `prod/sold_items/2016/CUR_TS/data.txt`

S3: Update partition for 2016 to point at `prod/sold_items/2016/CUR_TS/data.txt`
Enforce via API

Person A: Read data from prod.sold_items for 2016

API: What is the location for prod.sold_items for 2016?

MetaStore: prod/sold_items/2016/TS_1

S3: Read data from prod/sold_items/2016/TS_1/*
API for Data Scientists

Python:

```python
store_dataframe(df, dest_db, dest_table, partitions=['2016'])
df = load_dataframe(src_db, src_table, partitions=['2016'])
```

R:

```r
sf_writer(data = result, 
          namespace = dest_db, 
          resource  = dest_table, 
          partitions = c(as.integer(opt$ETL_DATE)))

sf_reader(namespace = src_db, 
           resource  = src_table, 
           partitions = c(as.integer(opt$ETL_DATE)))
```
Batch ID Pattern Benefits

- Full partition history
  - Can rollback
    - Data Scientists are less afraid of mistakes
  - Can create audit trails more easily
    - What data changed and when
  - Can anchor downstream consumers to a particular batch ID
Docker Enabled
DS @ Stitch Fix
Ad hoc Infra: In the Beginning...

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Env. Mgmt.</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Workstation" /></td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>
# Ad hoc Infra: Evolution I

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Env. Mgmt.</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Workstation Icon" /></td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td><img src="image2.png" alt="Environment Management Icon" /></td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Ad hoc Infra: Evolution I

1. **Workstation**: Various types of workstations are available with different features.
2. **Environment Management (Env. Mgmt.)**: Indicates the level of management required for each workstation.
3. **Scalability**: Specifies the scalability of each workstation, helping in understanding the adaptability to future demands.
## Ad hoc Infra: Evolution II

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Env. Mgmt.</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Workstation Image" /></td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td><img src="image2.png" alt="Workstation Image" /></td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td><img src="image3.png" alt="Workstation Image" /></td>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>
## Ad hoc Infra: Evolution III

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Env. Mgmt.</th>
<th>Scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Image of IP]</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>[Image of cloud]</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>[Image of docker]</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
Why Does Docker Lower Overhead?

- Control of environment
  - Data Scientists don’t need to worry about env.

- Isolation
  - can host many docker containers on a single machine.

- Better host management
  - allowing central control of machine types.
Flotilla UI

1 Active Containers

<table>
<thead>
<tr>
<th>name / alias</th>
<th>version</th>
<th>jupyter</th>
<th>rstudio</th>
<th>status</th>
<th>memory</th>
<th>uptime</th>
<th>$ so far</th>
<th>actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>stefan_qcon_sf</td>
<td>/flotilla:1.2</td>
<td>jupyter</td>
<td>rstudio</td>
<td>RUNNING</td>
<td>4 GB</td>
<td>a few seconds</td>
<td>$0</td>
<td>❌</td>
</tr>
</tbody>
</table>
Our Docker Image

- Has:
  - Our internal API libraries
  - Jupyter Notebook:
    - PySpark
    - IPython
  - Python libs:
    - scikit, numpy, scipy, pandas, etc.
  - RStudio
  - R libs:
    - Dplyr, magrittr, ggplot2, lme4, BOOT, etc.

- Mounts User NFS

- User has terminal access to file system via Jupyter for git, pip, etc.
Docker Deployment
Docker Deployment
Our Docker Problems So Far

- Docker tightly integrates with the Linux Kernel.
  - Hypothesis:
    - Anything that makes uninterruptable calls to the kernel can:
      - Break the ECS agent because the container doesn’t respond.
      - Break isolation between containers.
    - E.g. Mounting NFS

- Docker Hub:
  - Switched to artifactory
Scaling DS doing ML in the Cloud
1. Data Latency
2. To Batch or Not To Batch
3. What’s in a Model?
Data Latency

How much time do you spend waiting for data?
This could be a laptop, a shared system, a batch process, etc.
Use Compression

*This could be a laptop, a shared system, a batch process, etc.*
Use Compression - The Components

**Model**

```
[ 1.3234543 0.23443434 ... ]
[1 0 0 1 0 0 ... 0 1 0 0]
[0 1 0 1 ...]
[ ... 1 0 1 1]
```

**Features**

```
[1 0 0 1 0 0 ... 0 1 0 0]
```

**Predicted Results**

```
{100: 0.56, ..., 110: 0.65, ...
 ... , ..., 999: 0.43}
```
### Use Compression - Python Comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1.3234543 0.23443434 ... ]</td>
<td>[1 0 0 1 0 0 ... 0 1 0 0 0 1 0 1 ... ... 1 0 1 1]</td>
<td>[1 0 0 1 0 0 ... 0 1 0 0]</td>
<td>[1.3234543 0.23443434 ... ]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Features</th>
<th>Pickle: 3.1KB</th>
<th>Zlib+Pickle: 921B</th>
<th>JSON: 2.8KB</th>
<th>Zlib+JSON: 681B</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1 0 0 1 0 0 ... 0 1 0 0 ]</td>
<td>[1 0 0 1 0 0 ... 0 1 0 0 0 1 0 1 ... ... 1 0 1 1]</td>
<td>[1.3234543 0.23443434 ... ]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted Results</th>
<th>Pickle: 2.6MB</th>
<th>Zlib+Pickle: 600KB</th>
<th>JSON: 769KB</th>
<th>Zlib+JSON: 139KB</th>
</tr>
</thead>
<tbody>
<tr>
<td>{100: 0.56, ... ,110: 0.65, ... , ... , 999: 0.43}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Observations

- Naïve scheme of JSON + Zlib works well:

```python
import json
import zlib

...  
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```
Observations

- Naïve scheme of JSON + Zlib works well:

```python
import json
import zlib
...
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```

- Double vs Float: do you really need to store that much precision?
Observations

- Naïve scheme of JSON + Zlib works well:

```python
import json
import zlib
...
# compress
compressed = zlib.compress(json.dumps(value))
# decompress
original = json.loads(zlib.decompress(compressed))
```

- Double vs Float: do you really need to store that much precision?

- For more inspiration look to columnar DBs and how they compress columns
To Batch or Not To Batch:

When is batch inefficient?
Online & Streamed Computation

- Online:
  - Computation occurs synchronously when needed.

- Streamed:
  - Computation is triggered by an event(s).
Online & Streamed Computation

Very likely you start with a batch system
Online & Streamed Computation

Do you need to recompute:
- features for all users?
- predicted results for all users?

Very likely you start with a batch system.
Online & Streamed Computation

- Do you need to recompute:
  - features for all users?
  - predicted results for all users?

- Are you heavily dependent on your ETL running every night?

Model

Features

Predicted Results

Batch System

Very likely you start with a batch system
Online & Streamed Computation

- Do you need to recompute:
  - features for all users?
  - predicted results for all users?

- Are you heavily dependent on your ETL running every night?

- Online vs Streamed depends on in house factors:
  - Number of models
  - How often they change
  - Cadence of output required
  - In house eng. expertise
  - etc.

Very likely you start with a batch system
Online & Streamed Computation

- Do you need to recompute:
  - features for all users?
  - predicted results for all users?

- Are you heavily dependent on your ETL running every night?

- Online vs Streamed depends on in house factors:
  - Number of models
  - How often they change
  - Cadence of output required
  - In house eng. expertise
  - etc.

Very likely you start with a batch system

We use online system for recommendations
Streamed Example
Streamed Example
Streamed Example

Diagram:
- Trigger
- Kinesis
- Lambda
- Cache
- S3
- Online Prediction
Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
  - Hopefully $$$ savings
  - Hopefully less stressed Data Scientists
Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
  - Hopefully $$$ savings
  - Hopefully less stressed Data Scientists

- Requires better software engineering practices
  - Code portability/reuse
  - Designing APIs/Tools Data Scientists will use
Online/Streaming Thoughts

- Dedicated infrastructure → More room on batch infrastructure
  - Hopefully $$$ savings
  - Hopefully less stressed Data Scientists

- Requires better software engineering practices
  - Code portability/reuse
  - Designing APIs/Tools Data Scientists will use

- Prototyping on AWS Lambda & Kinesis was surprisingly quick
  - Need to compile C libs on an amazon linux instance
What’s in a Model?

Scaling model knowledge
Ever:
- Had someone leave and then nobody understands how they trained their models?
Ever:
- Had someone leave and then nobody understands how they trained their models?
  - Or you didn’t remember yourself?
Ever:

- Had someone leave and then nobody understands how they trained their models?
  - Or you didn’t remember yourself?

- Had performance dip in models and you have trouble figuring out why?
Ever:

- Had someone leave and then nobody understands how they trained their models?
  - Or you didn’t remember yourself?

- Had performance dip in models and you have trouble figuring out why?
  - Or not known what’s changed between model deployments?
Ever:
  ● Had someone leave and then nobody understands how they trained their models?
    ○ Or you didn’t remember yourself?
  ● Had performance dip in models and you have trouble figuring out why?
    ○ Or not known what’s changed between model deployments?
  ● Wanted to compare model performance over time?
Ever:
- Had someone leave and then nobody understands how they trained their models?
  - Or you didn’t remember yourself?
- Had performance dip in models and you have trouble figuring out why?
  - Or not known what’s changed between model deployments?
- Wanted to compare model performance over time?
- Wanted to train a model in R/Python/Spark and then deploy it a webserver?
Produce Model Artifacts

- Isn’t that just saving the coefficients/model values?
Isn’t that just saving the coefficients/model values?
  ○ NO!
Produce Model Artifacts

- Isn’t that just saving the coefficients/model values?
  - NO!
- Why?
Isn't that just saving the coefficients/model values?
○ NO!
Why?

Hyperparameters
Training Data
Who
When
Features
Performance
Library Versions
Final Coeff. Values

Model
Produce Model Artifacts

- Isn’t that just saving the coefficients/model values?
  - NO!
- Why?

How do you deal with organizational drift?

- Hyperparameters
- Training Data
- Who
- When
- Features
- Performance
- Library Versions
- Final Coeff. Values
Isn’t that just saving the coefficients/model values?
  ○ NO!

Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time

Produce Model Artifacts

- Hyperparameters
- Training Data
- Who
- When
- Features
- Performance
- Library Versions
- Final Coeff. Values
Produce Model Artifacts

- Isn’t that just saving the coefficients/model values?
  - NO!
- Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time

- Hyperparameters
- Training Data
- Who
- When
- Features
- Performance
- Library Versions
- Final Coeff. Values

Helps a lot with model debugging & diagnosis!
Produce Model Artifacts

- Isn’t that just saving the coefficients/model values?
  - NO!
- Why?

How do you deal with organizational drift?

Makes it easy to keep an archive and track changes over time

Hyperparameters
Training Data
Who
When
Features
Performance
Library Versions
Final Coeff. Values

Helps a lot with model debugging & diagnosis!

Model

Can more easily use in downstream processes
Produce Model Artifacts

- Analogous to software libraries
- Packaging:
  - Zip/Jar file
But all the above seems complex?
We’re building APIs.
Fin; Questions?

@stefkrawczyk