Machine Learning Program Management Example
Dependency Risk Analysis with AI Library

Assessing potential risk across diverse tracking systems

Date: January 2019
Presenters:
  Mark Yarborough, Program Management Team, Red Hat
  Dr. Prasanth Anbalagan, AI Center of Excellence Team, Red Hat
Business Need, Problem Space, Mitigations

Business Need: Red Hat Program Management’s core mission is to guide multiple product releases through development, test and ship. This entails managing risks across many simultaneous workstreams.

Problem Space: We need to assess risk for multiple dependencies, which are tracked in diverse systems, with differing workflows, and subject to target dates that may be flux.

Possible Mitigations:
- Converge multiple systems to single system
- Assign “Dependency Management” guru or team
- Write rules-based code to detect risks
- Try Machine Learning (ML)?

Let’s investigate this...
Goal: Assess Machine Learning Potential

Is ML currently “accessible” enough for non-Data Scientists to leverage for this use case?

Questions to answer:

- Is this a good use case for ML, given the data available?
- What steps required to prepare data for ML training?
- What facilities are available for ML training?

Results:

- Add ML to risk analysis process?
Presentation Flow

- Introduction
- Problem Space
- Training Data Setups
  - First Pass
  - Iterations
- Training and Results
  - AI Library overview and workflow
  - Training results for these examples
  - Updating AI Library for new use case
- Conclusion
CROSS-PRODUCT DEPENDENCIES
(Generic Problem Setup)
Consider a generic tracking system...

The system may contain “Issues” that represent new capabilities under development.

Example Systems:

- Jira
- Bugzilla
- Github/Gitlab
- Trello
- Rally
- etc...
Consider a second tracking system...

Identical in function, and different in every detail...

**Examples:**

- **Bugzilla**
  - Future Feature, state, custom flags, etc...

- **Jira**
  - Feature Request, status and resolution (combo), custom workflows, etc...

- **Trello**
  - Unstructured… Lot’s of variation…
Consider a schedule with a milestone...

The milestone may depend on delivery of new capabilities tracked in separate systems.

The tracking systems and the schedule system may not be (probably are not) natively capable of communicating with each other.
Consider a logical “Dependency Table”...

Capture critical dependencies for Schedule X, one per line, scattered across tracking systems.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date Needed</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue</td>
<td></td>
<td>etc</td>
</tr>
</tbody>
</table>

This is what we want to know...
There might be a lot of Dependency Tables...
Assume one per schedule, for possibly hundreds of schedules.

One per schedule, Maybe hundreds...
Maybe more than two tracking systems...

History, acquisitions, etc...

Red Hat uses Bugzilla, several Jira, Github, Rally, Trello, and various others…
TRAINING DATA SETUPS
(First Pass)
Training Data Example: Nutshell

We need to find, or fabricate, a large data set to feed our ML training algorithms. How large may not be known beforehand (more is usually better). We’d like a human-readable collection to ease the task of cleaning and tagging the data. We’ll ultimately need this data transformed into formats appropriate for the chosen training model. In the example that follows, we’ll prep the data using little or no “programming” (just spreadsheet massage).
Dependency transformations for ML training data

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date Needed</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

*Transformation(s)*

Training Sample(s)

<table>
<thead>
<tr>
<th>One Hot Issue Fields</th>
<th>One Hot “Days Left”</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1,0,0,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,…</td>
<td>0,9</td>
<td></td>
</tr>
</tbody>
</table>

Prediction

| 0,1,0,0,0,1,0,1,0,0,1,1,0,0,1,0,0,0,0,1,1,…,?? |  ?? |

Risk is provided for training samples

Risk is the output for prediction process

Easily Human readable
Initial Training Flow

We'll need to select a data source, then build training data and verify results...

Instance: JBoss Jira (https://issues.jboss.org/issues/)

Issue Types:
- Feature Request
- Business Requirement
- Enhancement

Range: Opened in 2018, as of Nov 2018
Grab first “raw data” from selected instance

This is the data we’ll massage further to create training sets.

Choose Source

Grab Data

Prep Data

Train & Measure

Jira Query Language (executed Nov 2018):

issuetype in ("Feature Request", "Business Requirement", Enhancement)

AND created > startOfYear()

Fields of interest:

Status,
Resolution,
Priority

Export Jira query results to CSV file

Import CSV to Google Sheet ⇐ This is where we’ll massage the training data

Yield: **5600+** issues ⇐ Looks like enough data to get us started
Details of grabbing raw data from JBoss Jira
Capture important fields for the 5600+ records, and move into google sheet for cleanup...

JQL Query:
issuetype in ("Feature Request", "Business Requirement", Enhancement) AND created > startOfYear()
Prepare data for first training run...

Typically a subset of the big data grab, formatted to suit the training algorithm.

Choose Source

Grab Data

Prep Data

Train & Measure

Training Algorithm:
- Supervised Learning
- Linear Regression

Training Fields (from Jira):
- Status, \( \leftarrow \) Our original query grabbed 30 separate values for Status
- Resolution, \( \leftarrow \) Original query grabbed 15 values, including blank, for Resolution

Target Value (we supply):
- Risk (range: 0 - 1)

Format: “One Hot” encoding, \( \leftarrow \) We’ll massage in new google sheet

Output: CSV file, \( \leftarrow \) We’ll hand this off to Prasanth to feed training model
Details for one hot encoding of Status/Resolution

We broke Status and Resolution in to 30,15 one hot sets, then added Risk (the training “answer”)...
Perform first training run and measure result

How close are the verification predictions to the training samples provided?

Choose Source

Grab Data

Prep Data

Train & Measure

Prasanth Anbalagan will cover this more in his section
Iterate: Prep, train, measure, repeat...

Refine chosen training samples, expand fields to train on, etc.

- Choose Source
- Grab Data
- Prep Data
- Train & Measure

Iterate as needed…

- Add more fields from original issues to increase depth and accuracy of prediction
- Add fields not contained in the original issue, but relevant to prediction
- “Synthesize” additional data (e.g. BLOCKED)
- Combine fields into simpler representations
- Favor false positives over false negatives

Timecheck!
TRAINING DATA SETUPS
(Iterations)
Iterate: Add “Days Left” value...

In reality, the risk is time-subjective: How far away is the actual deadline?

- Training Fields: Status, Resolution, Days Left
  - Derived from our “Target Date” value...

- Target Value: Risk (range: 0 - 1)
- Format: “One Hot” encoding
- Output: CSV file
Iterate: Collapse Status/Resolution to “State”

We’re going to map the 45 combinations down to a manageable 6 virtual “states”

Choose Source

Grab Data

Prep Data

Train & Measure

Prep Data

Train & Measure

Prep Data

Train & Measure

Training Fields: **State**, **Status**, **Resolution**, **Days Left**

Note: There is no “State” field native in Jira

Target Value: Risk (range: 0 - 1)

Format: “One Hot” encoding

Output: CSV file
Details: With mapped State and Days Left

We've gone from 55+ columns to the manageable set below, with little loss of information...

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mapped &quot;State&quot; OneHot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Punt</td>
<td>Block</td>
<td>Open</td>
<td>Work</td>
<td>Test</td>
<td>Done</td>
<td>Days Left OneHot</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

As always, export from "massaging" google sheet into CSV file to feed training algorithm

myarboro$ head -8 third-training-oneshots.csv
0,1,0,0,0,0,0,1,0,0,0,0,0,0.8
0,0,0,0,0,0,1,0,0,0,0,0,0,0
0,1,0,0,0,0,0,1,0,0,0,0,0,0.1
1,0,0,0,0,0,0,0,0,0,0,1,0,0.9
0,0,0,0,0,0,0,0,0,0,1,0,0,0.2
0,0,0,0,1,0,0,0,0,0,0,1,0,0.1
0,0,0,0,1,0,0,0,0,0,1,0,0,0.3
0,0,0,1,0,0,0,0,0,1,0,0,0,0.6
Iterate: Ongoing...

To increase accuracy of predictions, or to incorporate changes as workflows change, or...

- Add “Priority” ???
- Synthesize minority entries ??? (BLOCKED)
- Favor false positives ???

Choose Source

Grab Data

Prep Data

Train & Measure

Prep Data

Train & Measure

Prep Data

Train & Measure

Iterate ongoing…
Iterate More Instances?

Our larger goal is to handle many diverse tracking systems...

Choose Source

Grab Data

Prep Data

Train & Measure

Prep Data

Train & Measure

Prep Data

Train & Measure

Prep Data

Train & Measure

Prep Data

Train & Measure

Bugzilla?

Next Source

Grab Data

Prep Data

Train & Measure

Github?

Next Source

Grab Data

Prep Data

Train & Measure

etc...
Business Use Case to Machine Learning Solution

IMPLEMENTATION
- Data Science expertise required in implementing the models

INFRASTRUCTURE
- Choice of Infrastructure vs Deployment vs Management

ACCESSIBILITY
- Ease of use in deploying, managing, and modifying the implementation

COLLABORATION
- Sharing expertise and implementations to build a common foundation and consistency
AI LIBRARY
Reusable AI Components

OPEN SOURCE COLLECTION OF AI COMPONENTS
Machine learning solutions to common use cases to allow rapid prototyping. Some build upon use cases at Red Hat.

PART OF OPEN DATA HUB
“Machine learning-as-a-service” platform, built on top of OpenShift and Kubernetes
Understanding the Workflow

Save Data

Run Model

Use Results

aws

OpenWhisk

ML models

Container Application Platform

python modules

Red Hat OpenShift

Container Platform
**Workflow**

- Save data
- Invoke action (training, prediction, poll etc)
- Read data or store results
- poll status
- submit jobs (ML models)

Container Application Platform

- Project1
- OpenWhisk
- Jobs
- Project2

DevConf.CZ 2019
AI Library

- Object Storage (S3 Compatible)
- OpenWhisk + OpenShift
- Ansible playbooks for deployment
- Measures of predictive accuracy

AI Components

- AI Library
- Accuracy_measures
  - Association Rule Learning
  - Correlation Analysis
  - Duplicate Bug Detection
  - Flake Analysis
  - Matrix Factorization
  - Sentiment Analysis

- Storage
- Actions
- Ansible
Linear Regression

X vs Y

Y

X
### Y - Risk Value

### X - Status

**Resolution**

- Cannot Reproduce
- Deferred
- Done
- Duplicate
- Duplicate Issue
- Explained
- Incomplete Description
- Migrated to another ITS
- Out of Date
- Partially Completed
- Rejected
- Resolved at Apache
- Won't Do
- Won't Fix
- WONT FIX

**Days left until deadline**

- 2
- 5
- 10
- 20
- 40
- Over 40

---

**Multivariate Regression**
### Model Parameters:

**Intercept:**

-206421380772.45294

**Coefficients:**

\[
\begin{bmatrix}
1.18322170e+11 & 1.18322170e+11 & -9.88894856e+08 & -2.15835101e+08 \\
1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 \\
1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 \\
1.18322170e+11 & 1.18322170e+11 & -7.64732379e+08 & 1.18322170e+11 \\
1.18322170e+11 & 3.21852933e+08 & 1.18322170e+11 & 1.18322170e+11 \\
1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 & 1.18322170e+11 \\
1.18322170e+11 & 1.18322170e+11 & 5.39047694e+10 & 5.39047694e+10 \\
5.39047694e+10 & 5.39047694e+10 & -1.28103444e+09 & 3.46866002e+08 \\
5.89050989e+08 & 5.39047694e+10 & 5.39047694e+10 & 5.39047694e+10 \\
0.0000000000e+00 & 5.39047694e+10 & 0.0000000000e+00 & 5.39047694e+10 \\
5.39047694e+10 & 3.41944417e+10 & 3.41944417e+10 & 3.41944417e+10 \\
3.41944417e+10 & 3.41944417e+10 & 3.41944417e+10 & 3.41944417e+10
\end{bmatrix}
\]
<table>
<thead>
<tr>
<th>Risk</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>0.70</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>0.70</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.01</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>0.90</td>
<td>0.79</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td>0.00</td>
<td>0.15</td>
<td></td>
</tr>
</tbody>
</table>

**Prediction**

**Validation**

Absolute Error = |Predicted - Actual value|

Median Absolute Error 0.09
AI Library

AI Components

- Object Storage (S3 Compatible)
- OpenWhisk + OpenShift
- Ansible playbooks for deployment
- Measures of predictive accuracy

Accuracy Measures
- Association Rule Learning
- Correlation Analysis
- Duplicate Bug Detection
- Flake Analysis
- Matrix Factorization
- Sentiment Analysis
- Linear regression

Storage
- Actions
- Ansible
What is Open Data Hub?

The Open Data Hub is a machine learning-as-a-service platform built on Red Hat’s Kubernetes-based OpenShift® Container Platform, Ceph object storage, and Kubernetes, integrating a collection of open source projects. It inherits from upstream efforts such as Kubeflow and is the base of Red Hat’s internal data science and AI service. Data scientists can create models using Jupyter notebooks, and select from popular tools such as TensorFlow®, scikit-learn, apache Spark™ and more for developing models. Teams can spend more time solving critical business needs and less on installing and maintaining infrastructure with the Open Data Hub.

Open Data Hub is a meta-project that integrates open source projects into a practical solution. It aims to foster collaboration between communities, vendors, user enterprises, and academics following open source best practices. The open source community can experiment and develop intelligent applications without incurring high costs and having to master the complexity of modern machine learning and artificial intelligence software stacks.

Getting Started

For additional information about the Open Data Hub, read our blog.

To set up the Open Data Hub, all you need is a running OpenShift® cluster. For storing data and models, we recommend using a Ceph object store such as Ceph.

Once your OpenShift and Ceph installations are running, deploy the Open Data Hub components using our Ansible playbooks and OpenShift® deployment templates.
https://gitlab.com/opendatahub/ai-library

SEEKING ADDITIONAL CONTRIBUTIONS

Community contributions are growing (PnT Ops) and we want more.
CONCLUSION
Conclusion

- Dependency Risk Analysis
  - Did Machine Learning Work? Yes, to get us started…
  - Next, iterate to refine and expand scope...
- Journey from Business use case to Machine Learning Solution
  - Best Practices
    - Data Preparation
    - Modeling
      - Approach
      - Validation
      - Improvements