Internship experience at Salesforce

Rishab Srivastava
SID: 3031824316

August 6, 2019

Abstract

This summer, I interned at Salesforce.com, Inc. as a Software Engineer on their Security Einstein team. The team explores, builds and scales new ways to secure the Salesforce platform by applying big-data & machine learning techniques to security, specifically in anomaly detection, behavior-based analysis and correlation.

The team is part of the Einstein group at Salesforce, which uses sophisticated ML and AI technologies for prediction. We investigate application-layer attacks including credential stuffing, session hijacking, and phishing and invent detection & response techniques. The team works end-to-end on the model pipeline - right from proof-of-concept for models to data acquisition, data & feature engineering, pre-processing, production-level software engineering, and efficacy testing.

Figure 1: Salesforce.com, Inc, a customer relationship management solution.
www.salesforce.com

Contents

1 Introduction

As a software engineer intern with some background in data science, I was responsible for interfacing between Einstein engineers and Data Scientists to help scale and efficacy test some of the machine learning models that exist in the staging environment and production.

I wrote production-level code in Scala, Apache Spark and contributed to the Security Einstein codebase. I also worked with PySpark on the Amazon AWS to do some data engineering, feature extraction, and exploratory data analysis on EMR (Elastic-Map-Reduce) notebooks. I also ran EMR clusters and used S3 buckets for running some framework steps on big data.

2 ML model evaluation and efficacy

My first and main project involved building continuous efficacy testing routines for some of the machine learning models that the team already had in production, namely BadAppl, Session-Roaming-Detection and Cred-Stuffing. The main motivation behind efficacy testing is as follows:

- Ensuring model correctness and effectiveness in case of any improvements and engineering changes. Such changes may include, but are not limited to:
  - New features and transformations (BadAppl)
  - Improved detection logic or model behavior (SRD)

  - Transition to a new underlying technology (GraphLinks)
- Searching and selecting the best performing model by optimizing in the hyper-parameter space. Examples of hyper-parameters are:
  - User and org thresholds (Cred-Stuffing)
  - Size of training samples or training windows (BadAppl)
  - Number of discriminating columns, normalizationThreshold

2.1 Machine Learning in Security

A brief aside on ML model evaluation in the security domain. It is very difficult to obtain application-layer customer data due to trust and privacy concerns. When the user and organization data are acquired, it is difficult to obtain feedback for the data and categorize it as *anomalous* or normal. Since labels are not specified, this makes anomaly detection an *unsupervised learning* problem. As a result, the evaluation of model performance, especially efficacy, becomes problematic.

Ideally, we want a process that is similar to this:

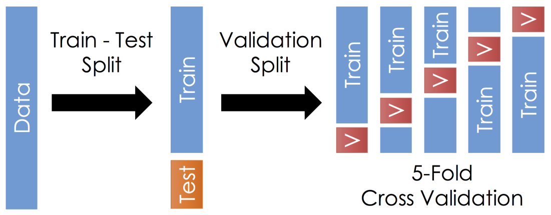


Figure 2: The train-test validation split

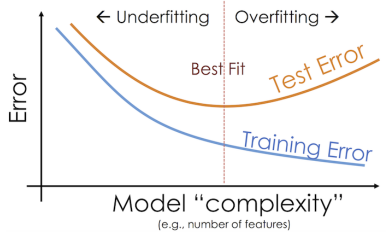


Figure 3: Select the best-fit model

Since we don't have labeled data, we follow the following routine to curate data, run the model, and test efficacy:

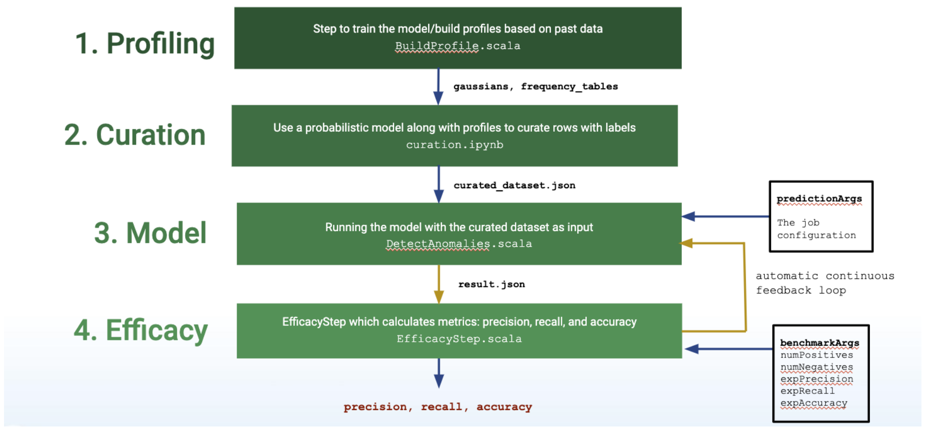
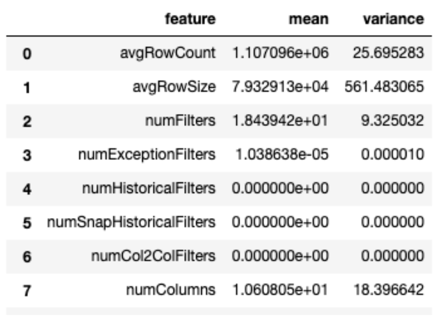

1. Profiling
Step to train the model/build profiles based on past data
BuildProfile.scala
gaussians, frequency_tables
2. Curation
Use a probabilistic model along with profiles to curate rows with labels
curation.ipynb
curated_dataset.json
3. Model
Running the model with the curated dataset as input
DetectAnomalies.scala
predictionArgs
The job configuration
result.json
automatic continuous feedback loop
4. Efficacy
EfficacyStep which calculates metrics: precision, recall, and accuracy
EfficacyStep.scala
benchmarkArgs
numPositives
numNegatives
expPrecision
expRecall
expAccuracy
precision, recall, accuracy

Figure 4: Dynamic validation routine for ML models

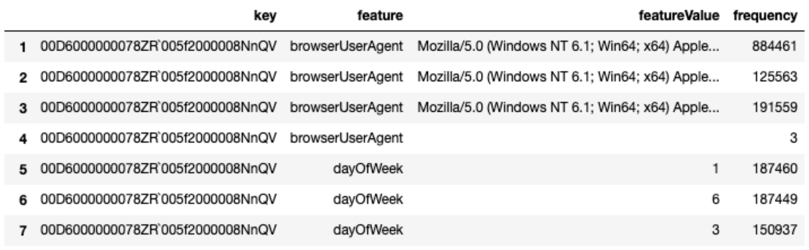
2.2 Profiling

Most machine learning models include a profiling stage that is used set certain baselines for user and organization behavior over time. This can also be called the training stage, depending on the kind of model that we are working with. Profiling is completed as a framework step before running the model, and the output from profiling is used by the model to make predictions and detect anomalous behavior.



	feature	mean	variance
0	avgRowCount	1.107096e+06	25.695283
1	avgRowSize	7.932913e+04	561.483065
2	numFilters	1.843942e+01	9.325032
3	numExceptionFilters	1.038638e-05	0.000010
4	numHistoricalFilters	0.000000e+00	0.000000
5	numSnapHistoricalFilters	0.000000e+00	0.000000
6	numCol2ColFilters	0.000000e+00	0.000000
7	numColumns	1.060805e+01	18.396642

Figure 5: Gaussians - mean and variances for numeric features



	key	feature	featureValue	frequency
1	00D600000078ZR`005f2000008NnQV	browserUserAgent	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...	884461
2	00D600000078ZR`005f2000008NnQV	browserUserAgent	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...	125563
3	00D600000078ZR`005f2000008NnQV	browserUserAgent	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...	191559
4	00D600000078ZR`005f2000008NnQV	browserUserAgent		3
5	00D600000078ZR`005f2000008NnQV	dayOfWeek	1	187460
6	00D600000078ZR`005f2000008NnQV	dayOfWeek	6	187449
7	00D600000078ZR`005f2000008NnQV	dayOfWeek	3	150937

Figure 6: Frequencies for categorical features

2.3 Data Curation

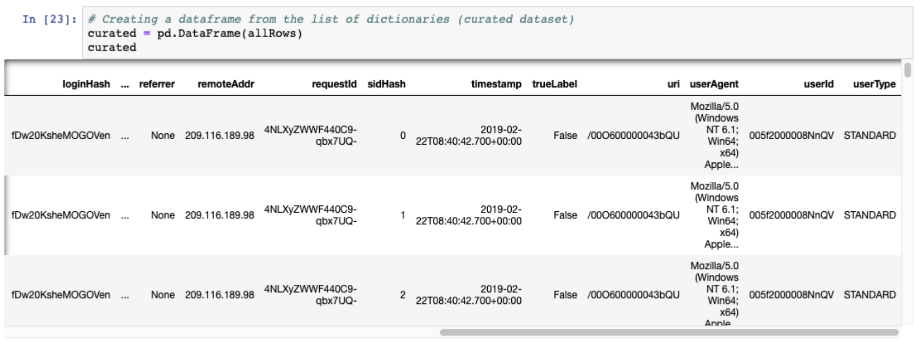
This step is essential to evaluating the efficacy of our model. Since our raw dataset does not have labels attached to it, we use this data curation step to create synthetic anomalies - which can be fed in to our model for detection. Data Curation is based on the following important assumptions:

- Most of the traffic is normal and a small percentage of incoming traffic is abnormal or *anomalous*
- Malicious traffic is statistically different from normal traffic, and thus appears less frequently
- We can then use a probabilistic model to curate data (with labels) as our validation dataset using the output from profiling

```
def createRows(n, gaussians, frequency_table):
    Generate a normal row using the gaussians, frequency_table
    If row is_anomaly with probability p:
      row[trueLabel] = True
      If row is_numeric anomaly with probability r:
        Set random numeric feature value = anomalous value
        (ex. rowCount = mean + 500 * SD)
      Else is_categorical:
            Set random categorical feature value = anomalous value
        (ex. browserType = torBrowser)
    Else:
        row[trueLabel] = False

Repeat for all n rows
```

Figure 7: Sample data curation process for BadAppl



```
In [23]: # Creating a dataframe from the list of dictionaries (curated dataset)
         curated = pd.DataFrame(allRows)
         curated
```

loginHash	...	referrer	remoteAddr	requestId	sidHash	timestamp	trueLabel	uri	userAgent	userId	userType
fDw20KsheMOGOVen	...	None	209.116.189.98	4NLXyZWWF440C9-qbx7UQ-	0	2019-02-22T08:40:42.700+00:00	False	/00O600000043bQU	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...	005f2000008NnQV	STANDARD
fDw20KsheMOGOVen	...	None	209.116.189.98	4NLXyZWWF440C9-qbx7UQ-	1	2019-02-22T08:40:42.700+00:00	False	/00O600000043bQU	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...	005f2000008NnQV	STANDARD
fDw20KsheMOGOVen	...	None	209.116.189.98	4NLXyZWWF440C9-qbx7UQ-	2	2019-02-22T08:40:42.700+00:00	False	/00O600000043bQU	Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple	005f2000008NnQV	STANDARD

Figure 8: Curated dataset for BadAppl - with **trueLabel** column

3 Efficacy Tests

3.1 BadAppl Efficacy Test

- Run the Principal Component Analysis based BadAppl model (with default configuration) on the curated dataset
- For each row, compare the item's `predictedLabel` with the `trueLabel` column
- Calculate truePositives, falsePositives, trueNegatives, falseNegatives
- Use these to calculate `precision`, `recall`, `accuracy` and test against expected benchmarks
- This test is run by Jenkins automatically every time that code is pushed to the master branch, i.e. every time any model is changed
  - This makes it straightforward to detect any efficacy/correctness failures for the model

3.2 SRD Efficacy Test

- SRD does not have an explicit profiling stage, so the EfficacyTest was more straightforward
- Test very similar to BadAppl model, except a few design decisions:
- Used default `zeroDayWeights` so that test is generalizable
- Positive (anomalous) and negative datasets obtained by Ankur
  - SRD Model run once with each dataset
- Metrics are calculated are returned as usual

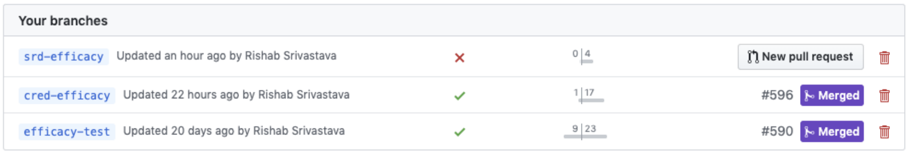
3.3 Cred-Stuffing Efficacy Tests

- Cred-stuffing does not have a profiling stage as well
  - Comes in the category of rule-based inference rather than machine learning
- Created extensive scenarios to test different cases in order to prove correctness of model
- Examples of cases include:
  - Case 1: A massive single endpoint attack
  - Case 2: An edge case with 499 endpoint events (`urlThreshold` = 500)
  - Case 3: A single domain with 3 different endpoints
  - Case 4: At-risk user login with 2 identical endpoint signatures
  - Case 5: User login with no detection endpoints
  - Case 6: At-risk user login after detection endpoint

4 Efficacy in Production

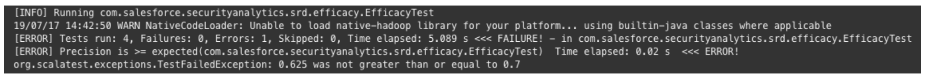
4.1 Jenkins Continuous Integration

Jenkins runs Efficacy Tests as part of the test suite for each model every time a commit is made to a branch. This makes it straightforward to detect any efficacy failures for changes that do not meet our expected benchmarks.



Your branches					
srd-efficacy	Updated an hour ago by Rishab Srivastava	✗	0 | 4	New pull request	
cred-efficacy	Updated 22 hours ago by Rishab Srivastava	✓	1 | 17	#596	Merged
efficacy-test	Updated 20 days ago by Rishab Srivastava	✓	9 | 23	#590	Merged

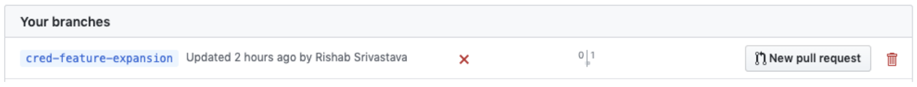
Figure 9: A code commit on the `srd-efficacy` branch



```
[INFO] Running com.salesforce.securityanalytics.srd.efficacy.EfficacyTest
19/07/17 14:42:50 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[ERROR] Tests run: 4, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 5.089 s <<< FAILURE! - in com.salesforce.securityanalytics.srd.efficacy.EfficacyTest
[ERROR] Precision is >= expected(com.salesforce.securityanalytics.srd.efficacy.EfficacyTest)  Time elapsed: 0.02 s  <<< ERROR!
org.scalatest.exceptions.TestFailedException: 0.625 was not greater than or equal to 0.7
```

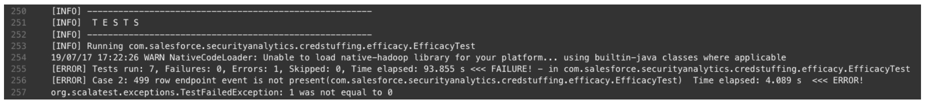
Figure 10: The commit fails our precision benchmark

In the example above, the branch `srd-efficacy` fails the **Precision is ≥ expected** test. This is because this branch changes some feature that causes the precision to drop to `0.625`. Our expected precision is 0.7 and so the test fails.


Your branches

cred-feature-expansion Updated 2 hours ago by Rishab Srivastava

Figure 11: A code commit on the `cred-feature-expansion` branch



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.salesforce.securityanalytics.credstuffing.efficacy.EfficacyTest
19/07/17 17:22:26 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
[ERROR] Tests run: 7, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 93.855 s <<< FAILURE! - in com.salesforce.securityanalytics.credstuffing.efficacy.EfficacyTest
[ERROR] Case 2: 499 row endpoint event is not present(com.salesforce.securityanalytics.credstuffing.efficacy.EfficacyTest)  Time elapsed: 4.089 s  <<< ERROR!
org.scalatest.exceptions.TestFailedException: 1 was not equal to 0
```

Figure 12: The commit fails **Case 2:** 499 row endpoint event is not present

Given above is a new branch we created to expand the features for the Cred-Stuffing model. However, due to this expansion, our model fails the Case 2 of our EfficacyTest. As a result, the code needs to be fixed before we can commit the changes to the `master` branch.

4.2 Efficacy Steps on AWS

For every model, there is a general version of `EfficacyTest` called `EfficacyStep` which can be run using an AWS EMR Cluster as a Framework Step as shown below.

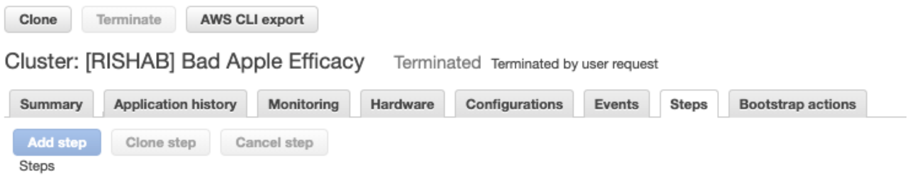

Cluster: [RISHAB] Bad Apple Efficacy Terminated Terminated by user request

Figure 13: An Elastic-Map-Reduce cluster to test efficacy for BadAppl

The Efficacy Step is run as a step after both Profiling and Model Step. The curated dataset is created manually (on Jupyter or EMR notebooks) and saved as JSON into an S3 bucket from where the model can access it. The entire process is as shown below:

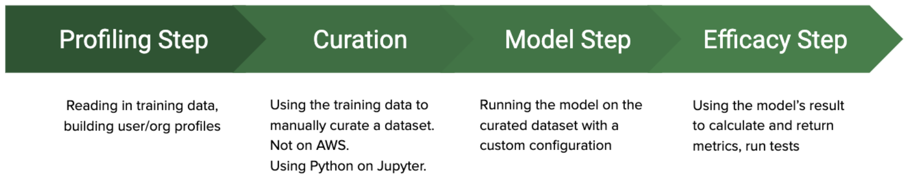

Profiling Step
Curation
Model Step
Efficacy Step
Reading in training data, building user/org profiles
Using the training data to manually curate a dataset. Not on AWS. Using Python on Jupyter.
Running the model on the curated dataset with a custom configuration
Using the model's result to calculate and return metrics, run tests

Figure 14: An Elastic-Map-Reduce cluster to test efficacy for BadAppl

5 Conclusion

In conclusion, I would like to thank my supervisor Mr. Manju Bhandary as well as my colleagues in Salesforce's Security Einstein team for making this internship an incredible learning experience for me. I would also like to thank my recruiter and the FutureForce team at Salesforce for all the fun events!

These two months have made me realize how my coursework at Berkeley is perfectly geared to achieve my future career goals in the field of Software Engineer and Data Science. I will continue taking courses in Computer Science, Statistics and Math to further strengthen my technical skills which can be utilized in industry.

Over the course of this internship, I have picked up a variety of technical and soft skills as well as learned about Software Engineering in industry. Some technical skills I've learned over the course of my internship are Scala, Spark, Kafka, and AWS. I want to continue working at the intersection of engineering and data science, in a team that uses cutting-edge machine learning and econometric techniques for prediction.