Advanced Machine Learning in SPL with the Machine Learning Toolkit

Jacob Leverich
Software Engineer, Splunk
Disclaimer

During the course of this presentation, we may make forward looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC. The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not, be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.
Who am I?

- Splunker for 2 years, based in San Francisco

- Engineering lead for...
  - ML Toolkit and Showcase App
  - ITSI Anomaly Detection and Adaptive Thresholding features
  - Splunk custom search command interface

- Initial author of fit/apply commands in ML Toolkit

- Die-hard Longhorns fan
Agenda

- Machine Learning + Splunk
- ML-SPL: Machine Learning in SPL
  - What it is
  - How it works
- Overview of Algorithms and Analytics available in ML-SPL
- Tips for Feature Engineering in SPL
- Wrap up
Machine Learning + Splunk
Machine Learning is Not Magic

- ... it’s a process.

- The process starts with a question:
  - How many requests do I expect in the next hour?
  - How likely is this hard drive to fail in the near future?
  - Am I being hacked?
    ‣ Is it unexpected for Joe to login to the bastion host at 2am?
Machine Learning is Not Magic

... it’s a process.
Data preparation accounts for about 80% of the work of data scientists.

What data scientists spend the most time doing:
- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%
Splunk for Data Preparation

Collect Data

Clean/Transform

Explore/Visualize

Model

ML Toolkit

Publish/Deploy

Evaluate

Alerts, Dashboards, Reports

props.conf, transforms.conf, Datamodels Add-ons from Splunkbase, etc.

Pivot, Table UI, SPL

Add-ons from Splunkbase, etc.
ML-SPL: Machine Learning in SPL
ML-SPL: What is it?

- A suite of SPL search commands specifically for Machine Learning:
  - fit
  - apply
  - summary
  - listmodels
  - deletemodel
  - sample

- Implemented using modules from the Python for Scientific Computing add-on for Splunk:
  - scikit-learn, numpy, pandas, statsmodels, scipy
ML-SPL Commands: A “grammar” for ML

- Fit (i.e. train) a model from search results
  
  $\ldots \mid \text{fit } <\text{ALGORITHM}> <\text{TARGET}> \text{ from } <\text{VARIABLES} \ldots> <\text{PARAMETERS}> \text{ into } <\text{MODEL}>$

- Apply a model to obtain predictions from (new) search results
  
  $\ldots \mid \text{apply } <\text{MODEL}>$

- Inspect the model built by $<\text{ALGORITHM}>$ (e.g. display coefficients)
  
  $\mid \text{summary } <\text{MODEL}>$
ML-SPL Commands: \texttt{fit}

```
... | fit \texttt{<ALGORITHM>} \texttt{<TARGET>} from \texttt{<VARIABLES>} ...
  \texttt{<PARAMETERS>} into \texttt{<MODEL>}
```

Examples:

```
... | fit \texttt{LinearRegression}
       system_temp from cpu_load fan_rpm
       into temp_model

... | fit \texttt{KMeans} \texttt{k=10}
       downloads purchases posts days_active visits_per_day
       into user_behavior_clusters

... | fit \texttt{LinearRegression}
       petal_length from species
```
**fit: How It Works**

1. Discard fields that are null for all search results.
2. Discard non-numeric fields with >100 distinct values.
3. Discard search results with any null fields.
4. Convert non-numeric fields to binary indicator variables (i.e. “dummy coding”).
5. Convert to a numeric matrix and hand over to `<ALGORITHM>`.
6. Compute predictions for all search results.
7. Save the learned model.
**fit: How It Works**

... fit LogisticRegression field_A from field_*

1. Discard fields that are null for all search results.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_A</td>
<td>field_B</td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
</tr>
<tr>
<td>ok</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
fit: How It Works

... | fit LogisticRegression field_A from field_*

2. Discard non-numeric fields with >100 distinct values.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_A</td>
<td>field_B</td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
</tr>
<tr>
<td>ok</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
### fit: How It Works

... | fit LogisticRegression field_A from field_*

3. Discard search results with any null fields.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_A</td>
<td></td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
</tr>
<tr>
<td>ok</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
fit: How It Works

... | fit LogisticRegression field_A from field_*

4. Convert non-numeric fields to binary indicator variables.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>field_A</td>
</tr>
<tr>
<td></td>
<td>field_B</td>
</tr>
<tr>
<td></td>
<td>field_D=red</td>
</tr>
<tr>
<td></td>
<td>...=green</td>
</tr>
<tr>
<td></td>
<td>...=blue</td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
</tr>
</tbody>
</table>
**fit: How It Works**

... | fit LogisticRegression field_A from field_*

5. Convert to a numeric matrix and hand over to `<ALGORITHM>`.

\[
y = [1, 1, 0] \quad \quad \quad \quad x = [[41, 1, 0, 0], \\
\quad [32, 0, 1, 0], \\
\quad [1, 0, 0, 1]]
\]

e.g. for Logistic Regression:

\[
\hat{y} = \frac{1}{1 + e^{-(\theta^T x)}}
\]

Find \( \theta \) using maximum likelihood estimation.

*Model inference generally delegated to scikit-learn and statsmodels.*
(e.g. `sklearn.linear_model.LogisticRegression`)
**fit**: How It Works

... | fit LogisticRegression field_A from field_*

6. Compute predictions for all search results.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>field_A</td>
<td>field_B, field_C, field_D, field_E</td>
<td>predicted(field_A)</td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
<td>red, 172.24.16.5, ok</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
<td>green, 192.168.0.2, ok</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
<td>blue, 10.6.6.6, FRAUD</td>
</tr>
<tr>
<td>ok</td>
<td>43</td>
<td>blue, 171.64.72.1, ok</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>blue, 192.168.0.2, FRAUD</td>
</tr>
<tr>
<td>coefficient</td>
<td>feature</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>1.464</td>
<td>species=iris Setosa</td>
<td></td>
</tr>
<tr>
<td>4.25</td>
<td>species=iris Versicolor</td>
<td></td>
</tr>
<tr>
<td>5.352</td>
<td>species=iris Virginica</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>_intercept</td>
<td></td>
</tr>
</tbody>
</table>
**fit: How It Works**

... | fit LogisticRegression field_A from field_* into logreg_model

7. Save the learned model.

Serialize model settings, coefficients, etc. into a Splunk lookup table.
- Replicated amongst members of Search Head Cluster.
- Automatically distributed to Indexers with search bundle.
<table>
<thead>
<tr>
<th>Path</th>
<th>Owner</th>
<th>App</th>
<th>Sharing</th>
<th>Status</th>
<th>Actions</th>
</tr>
</thead>
</table>
fit: Properties

- Each event is an “example” for the learning algorithm.

- Resilient to missing values. *(but be careful!)*

- Automatically handles categorical (e.g. non-numeric) fields.

**SAVES ITS WORK:**
- Learned model can be applied to *new, unseen* data with the *apply* command.
Some algorithms are inherently **not scalable**.
- e.g. Kernel-based Support Vector Machines is $O(N^3)$

Input is sampled using **reservoir sampling**.
- Per-algorithm sample reservoir size, typically 100,000 events
- Configurable in `mlsp1.conf`

Some algorithms support **incremental fitting**, e.g.: SGDRegressor, SGDClassifier, NaiveBayes
- Use "partial_fit=t" option with `fit` command.
- No sampling, no event limit!

For the most part, you don’t need to care.
ML-SPL Commands: apply

... | apply <MODEL>

Examples:

... | apply temp_model
... | apply user_behavior_clusters
... | apply petal_length_from_species
### New Search

```
Inputlookup iris.csv
| apply petal_length_from_species
| table species petal_length predicted(petal_length)
```

150 results (before 7/27/16 5:06:58.000 PM)  No Event Sampling  

**Line Chart**

![Line Chart](chart.png)

<table>
<thead>
<tr>
<th>species</th>
<th>petal_length</th>
<th>predicted(petal_length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris Setosa</td>
<td>1.4</td>
<td>1.466</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.4</td>
<td>1.464</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.3</td>
<td>1.464</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.5</td>
<td>1.464</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.4</td>
<td>1.464</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.7</td>
<td>1.465</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.4</td>
<td>1.464</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>1.8</td>
<td>1.464</td>
</tr>
</tbody>
</table>
apply: How It Works

1. Load the learned model.
2. Discard fields that are null for all search results.
3. Discard non-numeric fields with >100 distinct values.
4. Convert non-numeric fields to binary indicator variables (i.e. “dummy coding”).
5. Discard variables not in the learned model.
6. Fill missing fields with 0’s.
7. Convert to a numeric matrix and hand over to `<ALGORITHM>`.
8. Compute predictions for all search results.
apply: How It Works

... | apply fraud_model

4. Convert non-numeric fields to binary indicator variables.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>field_A</td>
</tr>
<tr>
<td>ok</td>
<td>41</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>41</td>
</tr>
</tbody>
</table>
**apply**: How It Works

... | apply fraud_model

5. Discard variables not in the learned model.

<table>
<thead>
<tr>
<th>Target</th>
<th>field_A</th>
<th>field_B</th>
<th>field_D=red</th>
<th>...=green</th>
<th>...=blue</th>
<th>...=yellow</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>41</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
apply: How It Works

... | apply fraud_model

5. Convert to a numeric matrix and hand over to <ALGORITHM>.

\[ y = [1, 1, 0, 1, ?] \quad \text{X} = \begin{bmatrix} [41, 1, 0, 0], \\
[32, 0, 1, 0], \\
[1, 0, 0, 1], \\
[41, 0, 0, 0] \end{bmatrix} \]

e.g. for Logistic Regression:

\[ \hat{y} = \frac{1}{1 + e^{-(\theta^T x)}} \]

Compute \( \hat{y} \) using \( \theta \) found by \texttt{fit} command.
apply: How It Works

... | apply fraud_model

7. Compute predictions for all search results.

<table>
<thead>
<tr>
<th>Target</th>
<th>Explanatory Variables...</th>
<th>Prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>41</td>
<td>red 172.24.16.5 ok</td>
</tr>
<tr>
<td>ok</td>
<td>32</td>
<td>green 192.168.0.2 ok</td>
</tr>
<tr>
<td>FRAUD</td>
<td>1</td>
<td>blue 10.6.6.6 FRAUD</td>
</tr>
<tr>
<td>ok</td>
<td>43</td>
<td>171.64.72.1 ok</td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>yellow 192.168.0.2 ok</td>
</tr>
</tbody>
</table>
**apply**: Properties

- Learned models can be applied to *new, unseen* data.

  ```plaintext
  | fit    is to    | apply
  as
  | outputlookup  is to | lookup
  ```

- Resilient to missing values. *(but, again, be careful!)*

- Automatically handles categorical (e.g. non-numeric) fields.
**apply**: Scalability

- No limits.

- When possible, executes at the Indexing tier.
  - Fully parallelized; harness the CPU power of your Indexing Cluster.
  - Must set “`streaming_apply = true`” in `mlspl.conf`. 
ML-SPL Commands: summary

... | summary <MODEL>

Examples:

... | summary temp_model
... | summary user_behavior_clusters
... | summary petal_length_from_species
```
inputlookup iris.csv
| fit logisticRegression species from petal_length petal_width sepal_length sepal_width into species_model
| sample 15
```

<table>
<thead>
<tr>
<th>petal_length</th>
<th>petal_width</th>
<th>predicted(species)</th>
<th>sepal_length</th>
<th>sepal_width</th>
<th>species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>0.2</td>
<td>Iris Setosa</td>
<td>4.7</td>
<td>3.2</td>
<td>Iris Setosa</td>
</tr>
<tr>
<td>1.5</td>
<td>0.3</td>
<td>Iris Setosa</td>
<td>5.1</td>
<td>3.8</td>
<td>Iris Setosa</td>
</tr>
<tr>
<td>1.2</td>
<td>0.2</td>
<td>Iris Setosa</td>
<td>5.0</td>
<td>3.2</td>
<td>Iris Setosa</td>
</tr>
<tr>
<td>1.4</td>
<td>0.3</td>
<td>Iris Setosa</td>
<td>4.8</td>
<td>3.0</td>
<td>Iris Setosa</td>
</tr>
<tr>
<td>1.5</td>
<td>0.2</td>
<td>Iris Setosa</td>
<td>5.3</td>
<td>3.7</td>
<td>Iris Setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>1.4</td>
<td>Iris Versicolor</td>
<td>6.1</td>
<td>2.9</td>
<td>Iris Versicolor</td>
</tr>
<tr>
<td>3.6</td>
<td>1.3</td>
<td>Iris Versicolor</td>
<td>5.6</td>
<td>2.9</td>
<td>Iris Versicolor</td>
</tr>
<tr>
<td>3.7</td>
<td>1.0</td>
<td>Iris Versicolor</td>
<td>5.5</td>
<td>2.4</td>
<td>Iris Versicolor</td>
</tr>
<tr>
<td>5.5</td>
<td>1.8</td>
<td>Iris Virginica</td>
<td>6.5</td>
<td>3.0</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.0</td>
<td>1.5</td>
<td>Iris Virginica</td>
<td>6.0</td>
<td>2.2</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.7</td>
<td>2.1</td>
<td>Iris Virginica</td>
<td>6.7</td>
<td>3.3</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.6</td>
<td>1.4</td>
<td>Iris Virginica</td>
<td>6.1</td>
<td>2.6</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.4</td>
<td>2.1</td>
<td>Iris Virginica</td>
<td>6.9</td>
<td>3.1</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.9</td>
<td>2.3</td>
<td>Iris Virginica</td>
<td>6.8</td>
<td>3.2</td>
<td>Iris Virginica</td>
</tr>
<tr>
<td>5.2</td>
<td>2.0</td>
<td>Iris Virginica</td>
<td>5.5</td>
<td>3.0</td>
<td>Iris Virginica</td>
</tr>
</tbody>
</table>
\[ \hat{y} = \frac{1}{1 + e^{-(\theta^T x)}} \]
Algorithms and Analytics in ML-SPL
Regression Algorithms
(e.g. predict numeric fields)

- LinearRegression
  - ... including Lasso, Ridge, ElasticNet
- KernelRidge
- DecisionTreeRegressor
- RandomForestRegressor
- SGDRegressor

- All implemented with sklearn models.
Classification Algorithms (e.g. predict categorical fields)

- LogisticRegression
- DecisionTreeClassifier
- RandomForestClassifier
- SGDClassifier
- SVM
- Naïve Bayes
  - Including BernoulliNB and GuassianNB
Clustering Algorithms (e.g. group like with like)

- KMeans
- DBSCAN
- Birch
- SpectralClustering
Feature Engineering Algorithms (e.g. data pre-processing)

- TFIDF (term-frequency x inverse document-frequency)
  - Transform free-form text into numeric fields
- StandardScaler (i.e. normalization)
- FieldSelector (i.e. choose K best features for regression/classification)
- PCA and KernelPCA
“Pipeline” Multiple Algorithms

Example: Text Analytics
- TFIDF to transform free-form messages into numeric fields, followed by...
  - KMeans to group similar messages
  - BernoulliNB to classify messages (e.g. according to sentiment)
  - PCA to visualize distribution of messages
- ... | fit TFIDF message | fit Kmeans message_tfidf_* | ...

Analogous to Pipeline concept from sklearn or Spark MLLib
“Pipeline” Multiple Algorithms

- ML-SPL analytics are *stackable*.

- Very advanced ML use-cases are succinctly expressible.
Tips for Feature Engineering
Tips for Feature Engineering

• Work on aggregates, not raw events.
  – DO NOT use fit on 1,000,000,000 events. DO use stats.

• Use eval to compute new features.

• Use streamstats to construct leading indicators.

• ...

Work on aggregates, not raw events

... | fit KMeans k=10
downloads purchases posts days_active visits_per_day
into user_behavior_clusters

- Use **stats** and lookup tables to construct features:

```
index=activity_logs
| stats count by action user_id
| xyseries user_id action count | fillnull
| lookup user_activity user_id
  OUTPUT days_active visits_per_day
| fit KMeans k=10 ...
```
Use `eval` to compute new features

- Coerce numbers into categories by prepending a string:
  - ... | `eval region_id = “Region ” + region_id` | ...

- Model interactions between features:
  - ... | `eval X_factor = importance * urgency` | ...
  - Use + for categorical fields, * for numeric

- Make non-linear features out of numeric values:
  - ... | `eval temperature = pow(temperature,2)` | ...
  - ... | `eval latency = log(latency)` | ...
Use **streamstats** for leading indicators

```bash
index=application_log OR index=tickets
| timechart span=1d count(failure) as FAILS, count("Change Request") as CHANGES
| reverse
| streamstats window=3 sum(FAILS) as FAILS_NEXT_3DAYS
| reverse
| fit LinearRegression FAILS_NEXT_3DAYS from CHANGES into FAILS_PREDICTION_MODEL
```
Wrap-up
What did we cover?

- Machine Learning + Splunk
- ML-SPL: Machine Learning in SPL
  - What it is
  - How it works
- Overview of Algorithms and Analytics available in ML-SPL
- Tips for Feature Engineering in SPL
What Now?

- Install the ML Toolkit from Splunkbase!
  - [http://tiny.cc/splunkmlapp](http://tiny.cc/splunkmlapp)
- Don’t miss Manish Sainani’s or Adam Oliner’s talks!

- Product Manager: Manish Sainani <msainani@splunk.com>
- Field Expert: Andrew Stein <astein@splunk.com>
- Me: Jacob Leverich <jleverich@splunk.com>
Multi-class classification problems typically modeled as “one-vs-rest”

Some algorithms do NOT support saved models, e.g.:
- DBSCAN and SpectralClustering
ML-SPL Commands

- `fit <ALGORITHM> <TARGET> from <VARIABLES ...> <PARAMETERS> into <MODEL>`
  - Fit (i.e. train) a model from search results

- `apply <MODEL>`
  - Apply a model to obtain predictions from (new) search results

- `summary <MODEL>`
  - Inspect the model inferred by <ALGORITHM> (e.g., display coefficients)