Splunk4Ninjas - Machine Learning and AIOps

Lab Guide

Overview

This lab guide contains the hands-on exercises for the Splunk4Ninjas - Machine Learning and AIOps workshop. Before proceeding with these exercises, please ensure that you have a copy of the workshop slide deck, which will help to put into context the tasks you are carrying out.

Download the workshop slide deck: https://splk.it/S4N-AIOps-Attendee

Prerequisites

Splunk.com Account

In order to complete these exercises, you will need your own Splunk instance. Splunk's hands-on workshops are delivered via the Splunk Show portal and you will need a splunk.com account in order to access this.

If you don't already have a Splunk.com account, please create one here before proceeding with the rest of the workshop.

Splunk Knowledge

A working knowledge of Splunk is assumed for this workshop.

⚠️ Troubleshooting Connectivity

If you experience connectivity issues with accessing either your workshop environment or the event page, please try the following troubleshooting steps. If you still experience issues please reach out to the team running your workshop.

- **Use Google Chrome** (if you're not already)
- If the event page (i.e. *https://show.splunk.com/event/<eventID>*) didn't load when you clicked on the link, try **refreshing the page**
- **Disconnect from VPN** (if you're using one)
- **Clear your browser cache and restart your browser** (if using Google Chrome, go to: Settings > Privacy and security > Clear browsing data)
- **Try using private browsing mode** (e.g. Incognito in Google Chrome) to rule out any cache issues
- **Try using another computer** such as your personal computer - all you need is a web browser! Cloud platforms like AWS can often be blocked on corporate laptops.

Table of Contents

Challenge 1 – Predict Numeric Fields

Predict database response time

1. Go to MLTK's **Experiment** tab and select **Create a New Experiment**.

   Experiment Type: Predict Numeric Fields
   Experiment Title: Predict Database Response Time

   Click **Create**.

2. Load the services_kpis.csv data set into the search bar:

   ```
   | inputlookup service_kpis.csv
   ```

3. In Preprocessing Steps, enter the following:

   Preprocess method: StandardScaler
   Select fields to preprocess: [Select all fields *except* for Database_Response_Time]

   Click **Apply** and wait to complete.

4. In Preprocessing Steps, click "Add a step" and enter the following:

   Preprocess method: FieldSelector
   Select field to predict: Database_Response_Time
   Select the predictor fields: [Select all SS_* fields]
   Type: Numeric
   Mode: Family-wise error rate
   Alpha: 0.05

   Click **Apply** and wait to complete. You'll see an Info message specifying which SS_* fields that FieldSelector identified.

5. In the Algorithm section:

   Algorithm: LinearRegression
   Field to predict: Database_Response_Time
   Fields to use for predicting: [Select all fs_* fields]

   Leave all other settings as default.

   Click **Fit Model** and wait to complete.

6. Scroll to the bottom of the page to find our model's R^2 Statistic, which is a score for how well a regression model predicted data. A model without errors has an R^2 value of 1, but anything between 0.50 to 0.99 is acceptable in the real world. The higher the R^2 value, the better!

   When we're happy with our R^2 value, scroll to the top of the page and click **Save** to save this experiment.

Experiment Title: Predict Database Response Time

Click **Save**.

7. Click **Go to Listings Page**, or go to the **Experiments** tab and look for your experiment under **Predict Numeric Fields**.

8. From the **Predict Database Response Time** experiment, click **Publish** and enter the following:

   New Main Model Title: predictDatabaseResponseTime
   Destination App: Splunk Machine Learning Toolkit

   Click **Submit**.

   You should see a “Models have been published” pop-up, with an SPL snippet. Append this snippet to apply the published model to any future IT infrastructure datasets to predict database response time:

```
... | apply predictDatabaseResponseTime_StandardScaler_1 | apply
predictDatabaseResponseTime_FieldSelector_2 | apply predictDatabaseResponseTime
```

9. To put this new model into practice, let's load up some ITSI KPI data that we want to use to model potential KPI response times. Open the Search tab in MLTK and enter the following:

```
| inputlookup service_kpis_forecast.csv
```

10. Now that we have some sample ITSI KPI data up, lets apply our model to that data, with an extra line of SPL at the end to clean up some of the fields:

```
| inputlookup service_kpis_forecast.csv
| apply predictDatabaseResponseTime_StandardScaler_1
| apply predictDatabaseResponseTime_FieldSelector_2
| apply predictDatabaseResponseTime
| fields - SS*, fs*, *_Time
```

    And as you can see, using our model we are able to generate a prediction of our expected database response times based on the other KPIs within the system. See field: `predicted(Database_Response_Time)`

Challenge 2 – Forecast Time Series

Forecast inbound network throughput

1. Go to MLTK's **Experiment** tab and **Create a New Experiment**.

   Experiment Type: Forecast Time Series
   Experiment Title: Forecast Network Throughput - Bytes In

   Click **Create**.

2. Load the data set into the search bar:

```
index="itsi_summary" indexed_itsi_service_id::b837ff99-b368-4798-bfd0-ac8d24a12aa4
earliest=-4h
| table _time, kpi, alert_value
| xyseries _time kpi alert_value
```

3. In the Algorithm section:

   Algorithm: Kalman Filter
   Field to forecast: Network Throughput - Bytes In
   Method: LLP5
   Future Timespan (in minutes): 30
   Holdback (in minutes): 30
   Confidence Interval: 85

   Leave all other settings as default.

   Click **Forecast** and wait to complete.

ℹ️ Definitions and Best Practice

Future Timespan: This determines how far ahead into the future you want your prediction to be. It is often best to keep this value as small as possible.

Holdback: Reserves some of the data before the model can train on it, and then tests our model on this data after it has been made. It tests how accurate your model is. Because the holdback determines how we evaluate our model, if we want to predict 30 minutes into the future, it is often best practice to also hold back 30 minutes of data so that we get a fair comparison for how our model will perform in the field.

Confidence Interval: This confidence interval determines two other values in addition to your prediction - an upper possible value, and a lower possible value. Users often use these values when determining best or worst case scenarios. With a higher confidence interval, we see that the actual value is within the cloud area. For a lower confidence interval, however, our prediction might not encapsulate the actual value of our system. We want a relatively snug fit between the cloud and the actual value.

Period: Period is specifically for data that repeats after a given period of time. (In our case, this is not applicable, so keep it blank.)

4. Scroll to the bottom of the page and verify that your R^2 Statistic is above 0.7.

5. Open the Forecast visualization panel in search and observe the SPL generated for forecasting inbound network throughput based on historical KPI data. Notice that it's simply your starting data set, appended with the Kalman Filter and your hyperparameter settings:

```
index="itsi_summary" indexed_itsi_service_id::b837ff99-b368-4798-bfd0-ac8d24a12aa4
earliest=-4h
| table _time, kpi, alert_value
| xyseries _time kpi alert_value
| predict "Network Throughput - Bytes In" as prediction algorithm=LLP5 holdback=30
future_timespan=60 upper85=upper85 lower85=lower85
| `forecastviz(60, 30, "Network Throughput - Bytes In", 85)`
```

6. Scroll to the top of the experiment page and hit "Save".

   Experiment Title: Forecast Network Throughput - Bytes In

   Click **Save**.

7. **(Optional)** Repeat this exercise and play around with different Kalman Filter Methods, and observe how it affects the overall forecast result, model performance (R^2 Statistic), and Forecast Outlier count.

Challenge 3 – Detect Outliers

Detect Numeric Outliers: Find anomalies in CPU utilization

1. Go to MLTK's **Experiment** tab and **Create a New Experiment**.

   Experiment Type: Smart Outlier Detection
   Experiment Title: Detect Entity Outliers

   Click **Create**.

2. Load the onpremdb_entity_metrics.csv data set into the search bar:

   ```
   | inputlookup onpremdb_entity_metrics.csv
   ```

3. Click the **Learn** tab on the left menu and under the **Detect Outliers** section, enter:

   Field to analyze: CPU Utilization %
   Split by fields: host
   Distribution type: Normal
   Outlier tolerance threshold: 0.01

   Leave all other fields as default.

   Click **Detect Outliers**.

4. Toggle **Chart Mode** on the right of the screen to "Split". Notice that out of the 3 hosts shown, two are experiencing strangely high amounts of CPU Utilization (identified as outliers), and one is experiencing none.

   

5. On the top right, click **Next** to enter the Review phase of the experiment. Click through each tile of the dashboard, which reports information on the reliability of the model, behavioral similarity of our data, and a summary of discovered outliers.

6. Review the model summary that gives us an overview of the statistical analysis done on each of the fields we split by, such as the min, max, mean, standard deviation, and distance metric of each host.

7. Review the distribution properties of the statistical behavior of our groups - a histogram of the mean and standard deviation with one distinct peak means that most of those groups have similar statistical behavior, as we see here. Seeing two or more distinct peaks lets us know that there are two distinct characteristics causing behavioral differences, etc. The outlier analysis shows the number of outliers we have per the group, as we saw on the previous page.

8. Click **Save** at the top.

   Experiment Title: Detect Entity Outliers

9. Click **Next** at the top to enter the "Operationalize" phase of the experiment. This is where we can create or manage alerts generated by our model, refresh its training, or schedule future training jobs, to make sure our model is analyzing our system using the most up to date system information.

   To save our model for future use, click **Publish Outlier Models**.

   New Main Model Title: detectCPUOutliers
   Destination App: Splunk Machine Learning Toolkit

   Click **Submit**.

10. With your model published, you can now use the following SPL command to run an accurate numerical outlier detection model to determine if future CPU measurements are outliers:

```
... |  apply detectCPUOutliers
```

   Click **Done**.

Detect Categorical Outliers: Find anomalies in servers based on response time

11. Open the Search tab in MLTK and load in the **deviceMetrics.csv** dataset:

```
| inputlookup deviceMetrics.csv
```

12. Encode the dataset:

```
| inputlookup deviceMetrics.csv
| eval {device} = 1
| eval {dest} = 1
| fillnull
```

13. Create the outlier model directly via SPL:

```
| inputlookup deviceMetrics.csv
| eval {device} = 1
| eval {dest} = 1
| fillnull
| anomalydetection action=annotate
| eval isOutlier = if(probable_cause != "", "1", "0")
```

14. Clean up the data:

```
| inputlookup deviceMetrics.csv
| eval {device} = 1
| eval {dest} = 1
| fillnull
| anomalydetection action=annotate
| eval isOutlier = if(probable_cause != "", "1", "0")
| where isOutlier > 0
| fields byte*, response*, dest, isOutlier, probable_cause
| where isOutlier=1
```

This lets us know exactly which servers are suffering anomalies due to response time, and how high those response times are. Notice that you can try this SPL workflow without encoding, and see that the categorical fields are integrated seamlessly as Splunk handles the encoding for you. This is an example of how you can easily operationalize outlier detection in Splunk.

Challenge 4 – Cluster Numeric Events

Match new notables with statistically similar notables

1. Go to MLTK's **Experiment** tab and **Create a New Experiment.**

   Experiment Type: Cluster Numeric Events
   Experiment Title: Cluster Notables

   Click **Create**.

2. Load the summarized list of historical notable events into the search bar, and count the notables by severity. Note down the number of unique severity levels shown in the Raw Data Preview, as this will determine the number of centroids to use within our clustering algorithm later:

```
| inputlookup resolved_notables.csv
| stats count by severity
```

3. Now just load the summarized list of historical notable events:

```
| inputlookup resolved_notables.csv
```

4. In the Algorithm section:

   Algorithm: K-means
   Fields to use for clustering: [Select all fields]
   K (# of centroids): 3

   Leave all other settings as default.

   Click **Cluster** and wait to complete.

5. You may notice an error "Error rendering Scatterplot Matrix visualization". We can ignore this error message and instead use another way to visualize the clustering results. Click **Save** at the top right to save the clustering model.

   Experiment Title: Cluster Notables

   Click **Save**.

   In the pop up, click **Go to Listing Page** which takes you to the **Experiments** tab.

6. From the **Cluster Notables** experiment, click **Publish** and enter the following:

   New Main Model Title: clusterNotables
   Destination App: Splunk Machine Learning Toolkit

   Click **Submit**.

   You should see a "Models have been published" pop-up, with an SPL snippet to invoke the model:

```
... | apply clusterNotables
```

7. Now we visualize our results. Go to the **Search** tab, load our **resolved_notables.csv** dataset, and apply the clustering model:

```
| inputlookup resolved_notables.csv
| apply clusterNotables
```

8. Scroll to the right of the results and we see two new fields - cluster and cluster_distance. To see these in a more human-visual manner, we can apply PCA analysis to draw out the three most important fields that influence the clustering, and graph them in 3D space:

```
| inputlookup resolved_notables.csv
| apply clusterNotables
| fit PCA k=3 *
| rename cluster as clusterId, PC_1 as x, PC_2 as y, PC_3 as z
```

9. Put these results into a “3D Scatterplot” visualization. From here, we can verify that our data forms three distinct clusters by color. Now we will apply this model to new notable events, and use it to identify which resolved_notables can be used to serve as a guide for solving our fresh notables.

   Open up a new browser tab and load one sample from our fresh_notables dataset into the **Search** tab:

```
| inputlookup fresh_notables.csv
| head 1
```

10. Apply our clustering model:

```
| inputlookup fresh_notables.csv
| head 1
| apply clusterNotables
```

    Note down which cluster this notable was put into, and what its cluster distance is:

cluster	
cluster_distance	

    This is very useful because, since our resolved_notables has been clustered using the same model, we can use the cluster group and distance as a way of measuring similarity.

11. Now let’s go back to our resolved_notables dataset with our clustering model:

```
| inputlookup resolved_notables.csv
| apply clusterNotables
```

12. Append a search for members of that cluster with a similar cluster distance:

```
| inputlookup resolved_notables.csv
| apply clusterNotables
| search cluster=[insert cluster] cluster_distance>[insert cluster_distance - 5]
cluster_distance<[insert cluster_distance + 5]
| dedup cluster_distance
```

Now, from around ten thousand events, we have a way of quickly identifying four previous notables which are similar to our current issue. We can then use the drilldown URI to investigate the service which the notable triggered from, or use the event_id to search up that event in ITSI and get more details about how this notable was resolved in the past - by not having to reinvent the wheel, we are able to get to resolving this issue with guidance, all while saving us time and effort.

Challenge 5 – Categorical Predictions

Predict credibility of an alert

1. Go to MLTK's **Experiment** tab and **Create a New Experiment**.

   Experiment Type: Predict Categorical Fields
   Experiment Title: False Positive Notables Prediction

   Click **Create**.

2. Load the **cat_reg.csv** data set into the search bar:

   ```
   | inputlookup notablesTrain.csv
   ```

   Observe our target field and feature fields:

   Target field: predicate
   Feature fields: [All other fields]

3. In **Preprocessing Steps**, enter the following:

   Preprocess method: StandardScaler
   Select fields to preprocess: [Select all fields *except* for predicate]

   Click **Apply** and wait to complete.

4. In **Preprocessing Steps**, click **Add a step** and enter the following:

   Preprocess method: FieldSelector
   Select field to predict: predicate
   Select the predictor fields: [Select all SS_* fields]
   Type: Categorical
   Mode: False Positive Rate
   Alpha: 0.05

   Click **Apply** and wait to complete. You'll see an Info message specifying which SS_* fields that FieldSelector identified.

5. In the Algorithm section:

   Algorithm: RandomForestClassifier
   Field to predict: predicate
   Fields to use for predicting: [Select all fs_* fields]
   Split for training/test: 70/30

   Leave all other settings as default.

   Click **Fit Model** and wait to complete.

6. Scroll to the bottom of the page to find our model's scores of how well the random classifier model performed. The closer these scores are to 1, the better! Also review the confusion matrix, which is a way to express how many of a classifier's predictions were correct, and when incorrect, where the classifier got confused.

   When we're happy with our results, scroll to the top of the page and click **Save** to save this experiment.

   Experiment Title: False Positive Notables Prediction

   Click **Save**.

7. Click **Go to Listings Page**, or go to the **Experiments** tab and look for your experiment under **Predict Categorical Fields**.

8. From the **False Positive Notables Prediction** experiment, click **Publish** and enter the following:

   New Main Model Title: notablesFPClassifier
   Destination App: Splunk Machine Learning Toolkit

   Click **Submit**.

   You should see a "Models have been published" pop-up, with an SPL snippet. Append this snippet to apply the published model to any future notables, to predict the categorical label of your event based on its features:

```
... | apply notablesFPClassifier_StandardScaler_1 | apply
notablesFPClassifier_FieldSelector_2 | apply notablesFPClassifier
```

9. Let's load up some new notable data and use our model. Open the **Search** tab in MLTK and enter the following:

```
| inputlookup notablesTest.csv
```

10. Now that we have some live notable data up, lets apply our model to that data, with an extra line of spl to clean up some of the fields:

```
| inputlookup notablesTest.csv
| apply notablesFPClassifier_StandardScaler_1
| apply notablesFPClassifier_FieldSelector_2
| apply notablesFPClassifier
| fields - SS*, fs*, _time
```

    Using our model we generate a prediction – placed in the field predicted(predicate) – of our true and false positive notables.