Resilient Machine Learning User Guide

Installation

Complete the following steps to install the Python components:

1. Ensure that the environment is up-to-date, as follows:

sudo pip install --upgrade pip

sudo pip install --upgrade setuptools

sudo pip install --upgrade resilient-circuits

1. Run the following command to install pandas:

sudo pip install pandas-0.23.4-cp27-cp27m-linux\_86\_64.whl

This step is necessary because there is no wheel for pandas on Redhat Enterprise 7 available from pip. Also Redhat Enterprise 7 does not have gcc/g++ installed, so pip cannot build it from source code. A wheel was built locally, using the source code of <https://files.pythonhosted.org/packages/e9/ad/5e92ba493eff96055a23b0a1323a9a803af71ec859ae3243ced86fcbd0a4/pandas-0.23.4.tar.gz>. This is the same source code as the “pip install pandas” downloads.

1. Run the following command to install the package:

sudo pip install --upgrade fn\_machine\_learning-<*version*>.zip

Configure the Python components

The Resilient Circuits components run as an unprivileged user, typically named integration. If you do not already have an integration user configured on your appliance, create it now.

Complete the following steps to configure and run the integration:

1. Using sudo, switch to the integration user, as follows:

sudo su - integration

1. Use one of the following commands to create or update the resilient-circuits configuration file. Use –c for new environments or –u for existing environments.

resilient-circuits config -c

or

resilient-circuits config -u

1. Edit the resilient-circuits configuration file, as follows:
   1. In the [resilient] section, ensure that you provide all the information required to connect to the Resilient platform.
   2. In the [machine\_learning\_predict] section, edit the settings as follows:

active\_model=path\_to\_a\_saved\_model\_file

* 1. In the [machine\_learning] section, edit the settings as follows:

prediction=API\_name\_of\_field\_to\_predict

features=fields\_to\_be\_used\_as\_features\_separated\_by\_comma

algorithm=algorithm\_for\_model

method=optional\_ensemble\_method

split=split\_train\_test\_default\_0.5

max\_count=optional\_limit\_on\_max\_number\_of\_samples\_to\_process

Please note that these are the algorithms and ensemble methods supported currently.

Algorithms:

* Logistic Regression
* SVM
* SVM with Gaussian kernel
* Decision Tree
* Random Forest
* GaussianNB
* BernoulliNB
* K-Nearest Neighbor

Ensemble methods:

* Bagging
* Adaptive Boosting

Deploy Customizations to the Resilient platform

This package contains one function definition, and includes one example workflow and a rule that runs this function.

1. Use the following command to deploy these customizations to the Resilient platform:

resilient-circuits customize

1. Respond to the prompts to deploy the function, message destination, workflow, script, and rule.

Run the Integration

To test the integration package before running it in a production environment, you must run the integration manually with the following command:

resilient-circuits run

The resilient-circuits command starts, loads its components, and continues to run until interrupted. If it stops immediately with an error message, check your configuration values and retry.

﻿Build a Machine Learning Model

This package includes a command line tool to build a machine model. It reads the settings from the [machine\_learning] section of the app.config to build the model. This command line tool has three subcommands.

## Build

This subcommand is used to build a new model. It takes two flags:

|  |  |
| --- | --- |
| -o | Required.  File path to the file to save the built model. |
| -c | Optional.  File path to a CSV file that contains the samples.  If this flag is absent, the tool downloads incidents from the Resilient platform (specified in app.config) as samples.  If this flag is given, the CSV file is used for samples instead. |

Example:

res-ml build -o lg\_adaboost.ml

If the model can be built successfully, an output like this is shown:



## Rebuild

This subcommand is used to rebuild a saved model. It takes two flags:

|  |  |
| --- | --- |
| -i | Required.  File path to the saved model to be rebuilt. |
| -c | Optional.  File path to a CSV file that contains the samples.  If this flag is absent, this tool downloads incidents from the Resilient platform (specified in app.config) as samples.  If this flag is given, the CSV file is used for samples instead. |

Please note that when a model is rebuilt, the predict/features/algorithm/method information is taken from the saved file instead of from app.config. This subcommand is intended for rebuilding/updating a successful model after new samples are available.

Example:

res-ml build -i lg\_adaboost.ml

If the model can be rebuilt successfully, an output like this is shown:



## View

This subcommand is used to view a saved model. It takes one flag:

|  |  |
| --- | --- |
| -i | Required.  File path to the saved model to be rebuilt. |

Example:

res-ml view -i lg\_adaboost.ml

It shows the summary of the saved model.



## Logging

The res-ml command line utility creates a log file called res-ml.log in the current folder you run res-ml. It also generates a CSV file called resilient\_incidents.csv to store all the samples.

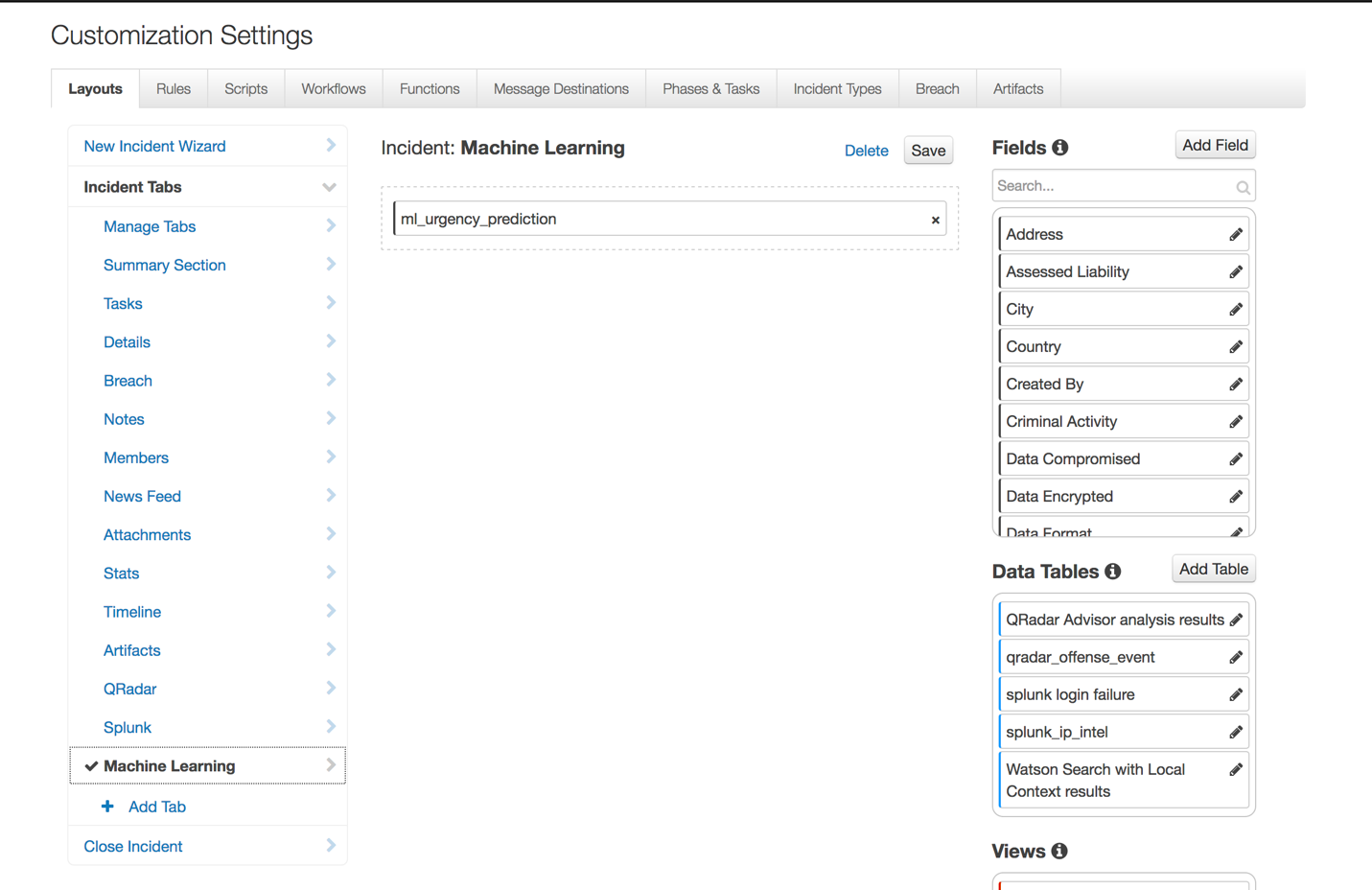
Use a ML Model to Predict

Copy the model you want to use for prediction to the active\_model specified in the app.config. This is easier than modifying the app.config to point to the model file you want to use, because you do not need to restart resilient-circuits.

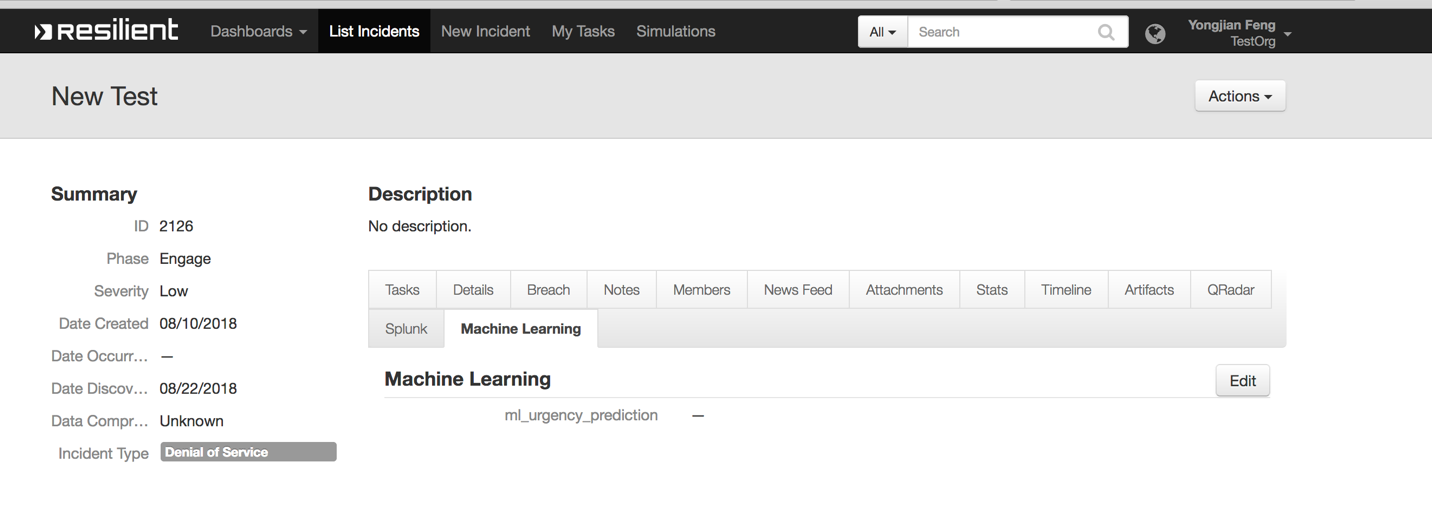
The sample workflow included in this integration is called “ML predict urgency”. A custom field called “ml\_urgency\_prediction” is added to your Resilient platform when you run the following command:

resilient-circuits customize

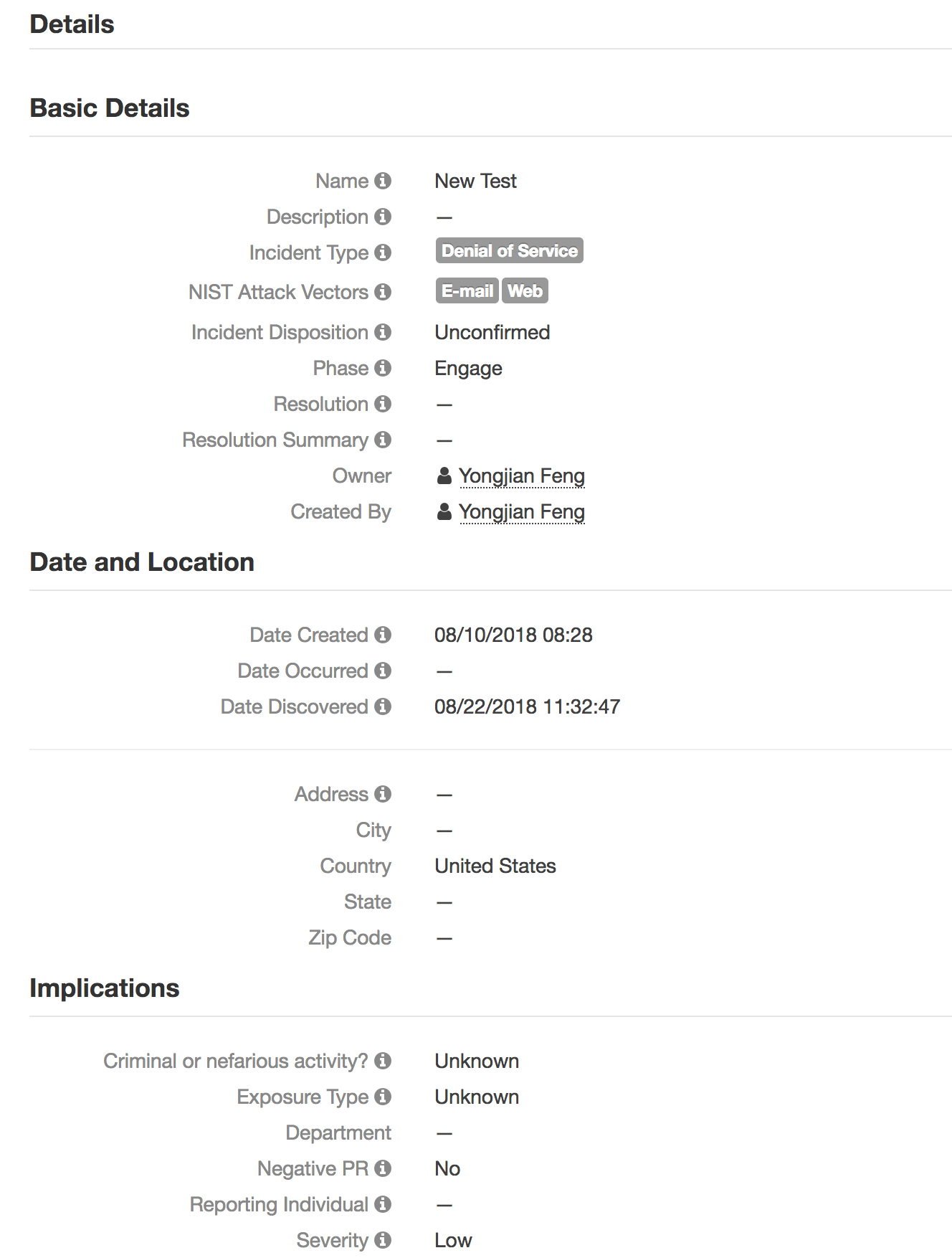
Go to Customization Settings page and create a new incident tab called “Machine Learning”. Then add this custom field into the newly created tab.



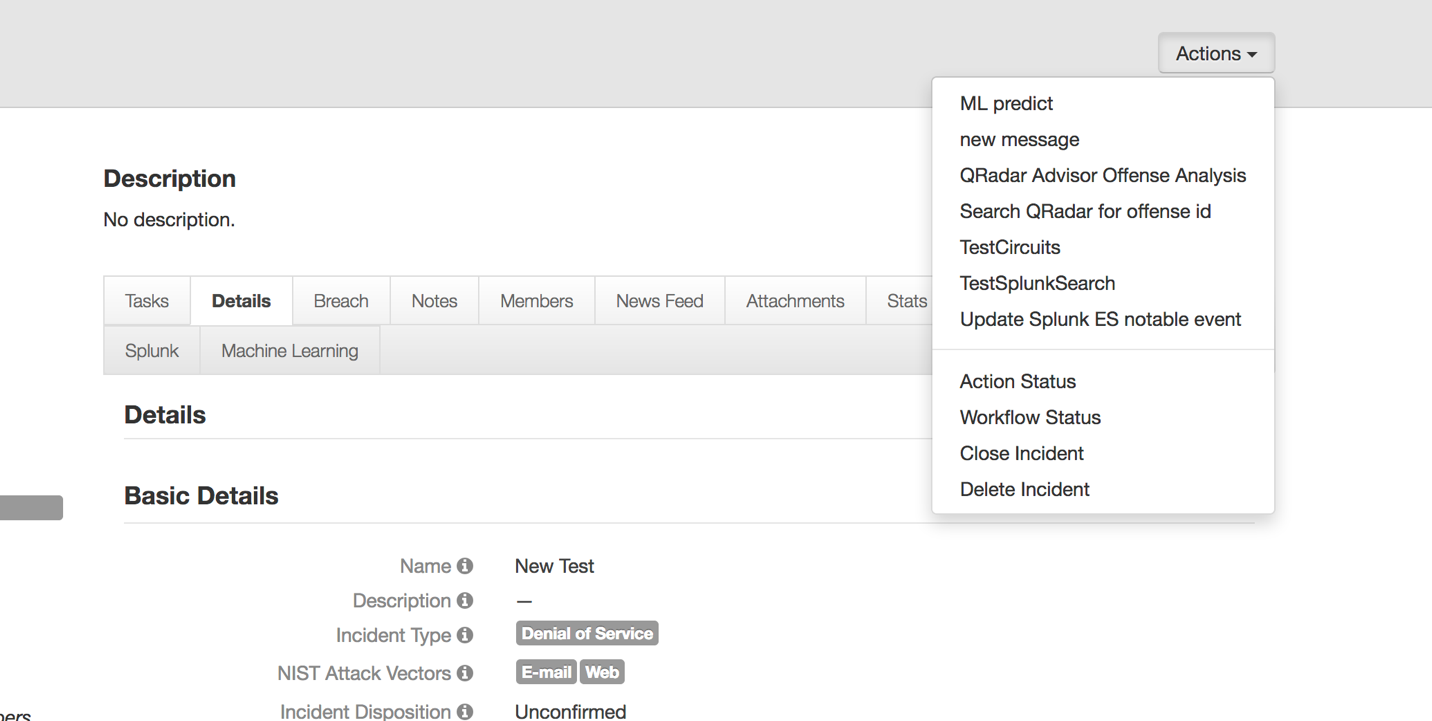
The tab and custom field are in the incident page



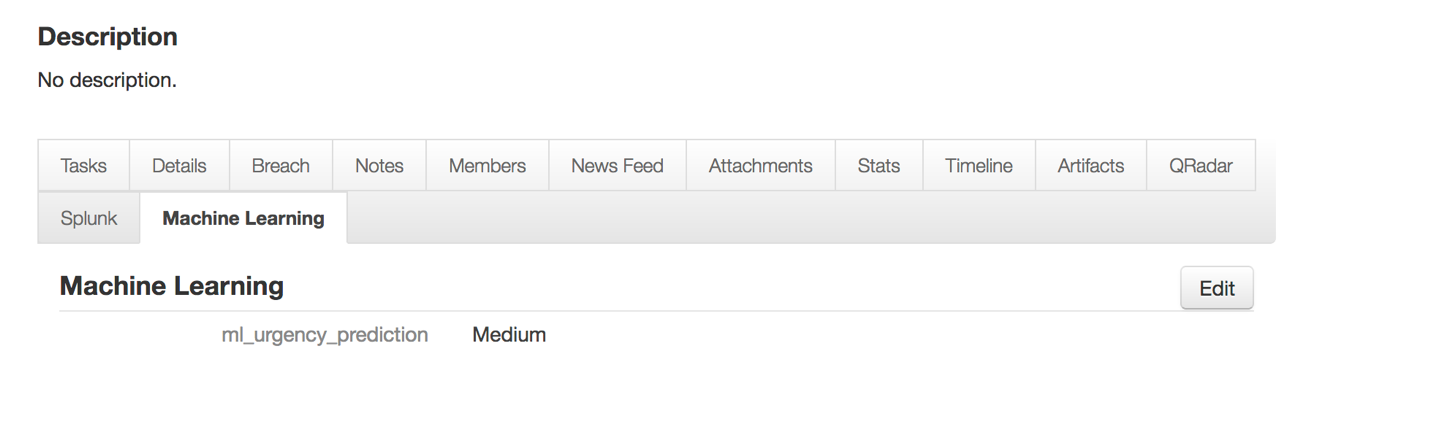
Go to the “Details” tab to make that all the features are non-empty. In this particular example, the features are “NIST Attack Vectors”, “Incident Type”, “Incident Disposition”, and “Negative PR”. From the following screenshot, we can see none of these fields is blank here. Therefore, this an incident we can predict.



From the Actions button in an incident, click “ML predict”.



Wait for the integration to finish its job. The prediction is shown.



Note

In the sample workflow, the prediction is stored in a custom field, ml\_urgency\_prediction. The user needs to manually copy it over to severity\_id (which is the predict field).

If user wants to write the prediction value into severity\_id directly, do this in the post-process script of the sample workflow.

Also, user wants to predict something other than severity\_id, change the post-process script to write to the desired field.

Limitations

Unicode is not supported yet. All features and prediction need to be in ASCII format.

Only classification is implemented. Regression is not yet supported.

Trouble Shooting

These are common errors encountered during machine learning:

## The least populated class in y has only 1 members, which is too few

One particular predict value has only one sample. Say for example, 1000 incidents are used as samples for training to predict “severity\_code”. If out of those 1000 samples, only one has severity\_code = high, then this error will be shown.

As for resolution for this, more samples are needed. Specifically, more samples with severity\_code = high are needed.

## This solver needs samples of at least 2 classes in the data, but ….

This means all the samples use to train the model carry the same predict value. For example, 1000 incidents are used for training to predict “severity\_code”. If all of those 1000 incidents have severity\_code = low, then this error will be shown.

This means this is nothing to learn from these samples for predicting that particular field.

## Not samples to train a model

Note that in the pre-process steps, we get rid of all the samples with missing/blank features. As a result, even if you might have a large number of incidents to start with, after filtering those with missing features, you will end up with less number of or even 0 incidents.

This means you need to select features carefully, or even need to fill in the missing features first if possible.