Apereo Learning Analytics Processor

Goals

Build an Open source Java based Learning Analytics Processor (LAP) which initially automates the Marist OAAI Student Early Alerts and Risk Assessment model

Establish a framework for automation and execution of learning analytics models (which is possible to extend with additional model pipelines)

Establish input and output specifications for data used for model processing

Implementation

Technicals

- We are developing in GitHub: https://github.com/Apereo-Learning-Analytics-Initiative/LearningAnalyticsProcessor
- We are tracking issues in JIRA: https://jira.sakaiproject.org/browse/LAI
    - Parent story ticket: LAI-3 - LAP 1.0 - Initial learning analytics processor (OAAI automation) OPEN
- The Sakai data extraction is being developed as a separate app: https://github.com/Apereo-Learning-Analytics-Initiative/LAP-Sakai-Extractor

Architecture

LAP (Learning Analytics Processor) is meant to flexible enough to be extended to support many possible models and pipelines for analytics processing. The first one will be Early Alert but we want to ensure we support future additions and even multiple versions of the Early Alert model.

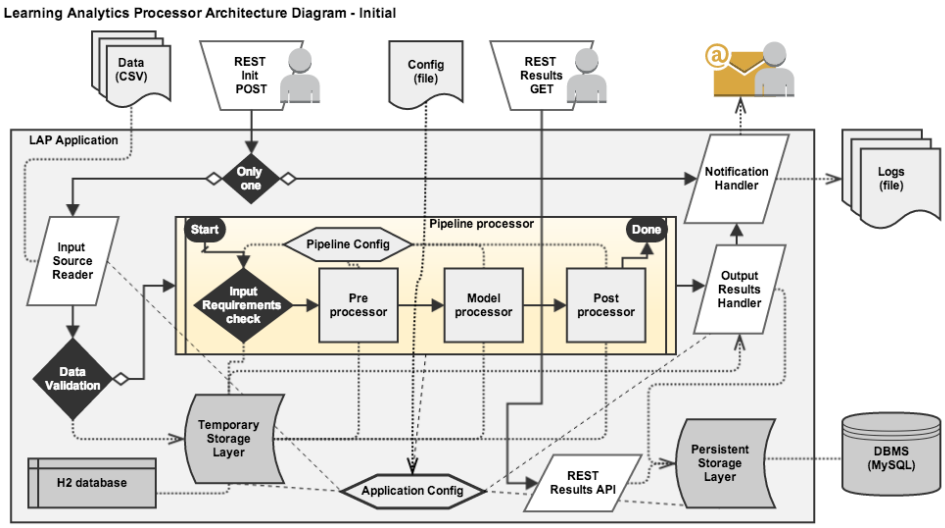
LAP will be setup and installed as a web application. An admin will configure it to look for data from a set of sources (initially just CSV files in a given directory). LAP should be setup to run on a schedule (probably once a day) but that will be under the control of the system admin. LAP will then execute (on schedule or manually) a specific pipeline (initially Marist Early Alert) which will work like so:
1) Check for input sources at the configured location
2) Validate the input sources
3) Initiate the pipeline (verify the sources have required data)
4) Execute the pipeline kettle parts (based on the config)
5) Complete the pipeline (post processing transforms -> temp store)
6) Store output results into the persistent storage
7) Notify that pipeline processing is complete

The 5 main parts of the LAP application architecture are:
A) Input source management
B) Data storage - Temporary (for use during processing) and Persistent (for tracking and results)
C) Configuration management
D) Processing pipeline
E) Output results management

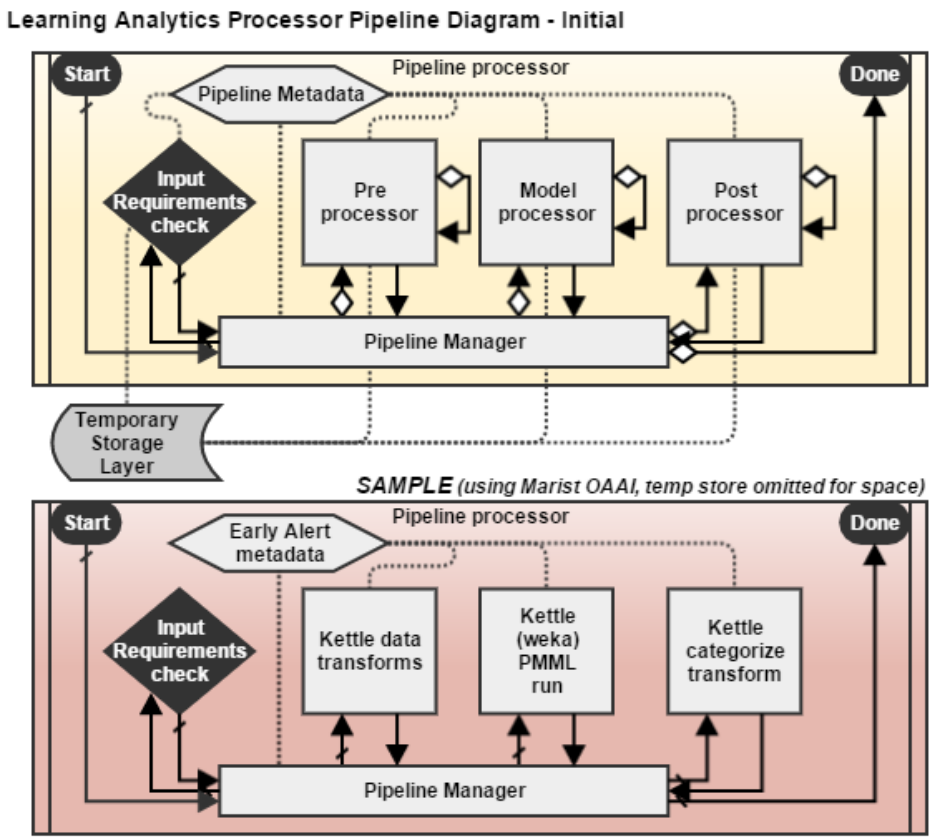
Within the pipeline there are 3 parts:
P1) Pre-processing (kettle or code based)
P2) Model runner (kettle based)
P3) Post-processing (kettle or code based transforms/maps/cleanups)
A pipeline can optionally include each piece so that it may only have a runner or pre-processor or post-processor.

Each pipeline will be defined by a set of metadata which includes:
- name
- description (and recommendations for running the model)
- stat indicators (accuracy, confidence interval, etc.)
- required input fields
- processors (kettle ktr and kjb files, pmml files, etc.)
- output result definition

Diagrams

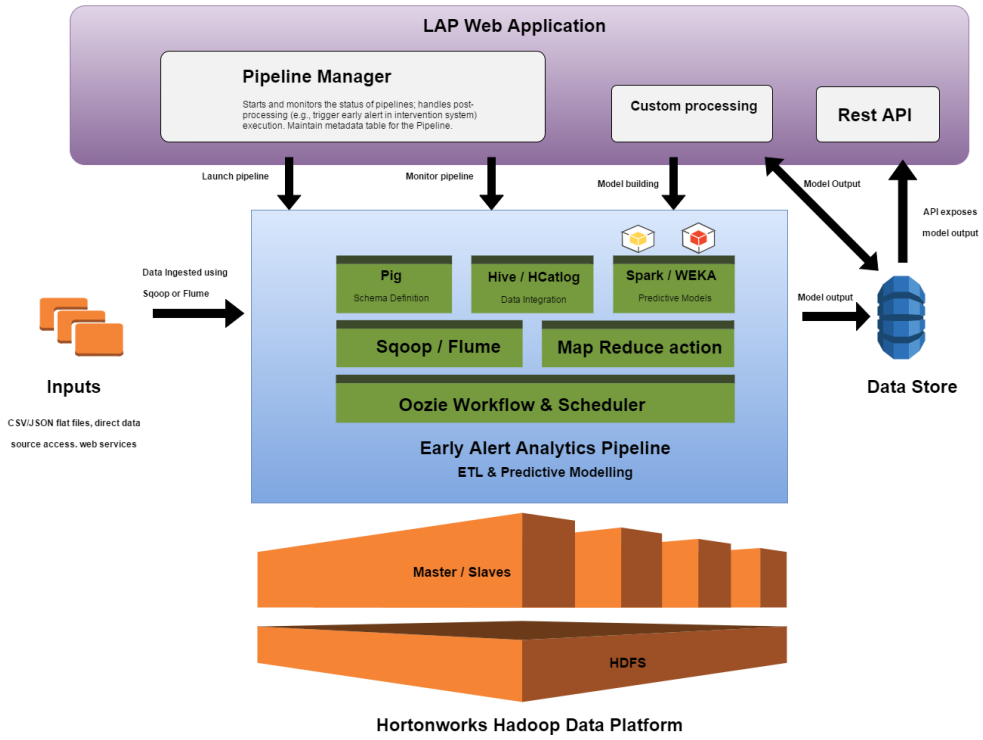


Pipeline



Scalability Roadmap for LAP

Apereo Learning Analytics Processor Architecture



Other stuff

Contacts: Sandeep Markondiah Jayaprakash, Gary Gilbert, Joshua Baron, Aaron Zeckoski

Part of the Apereo Learning Analytics Initiative (LAI)