

Jenkins Machine Learning Plugin for Data Science

Google Summer of Code Program 2020 Project Proposal

Perinpanayagam Loghi
Email : loghijiaha.16@cse.mrt.ac.lk
Github : Loghijiaha

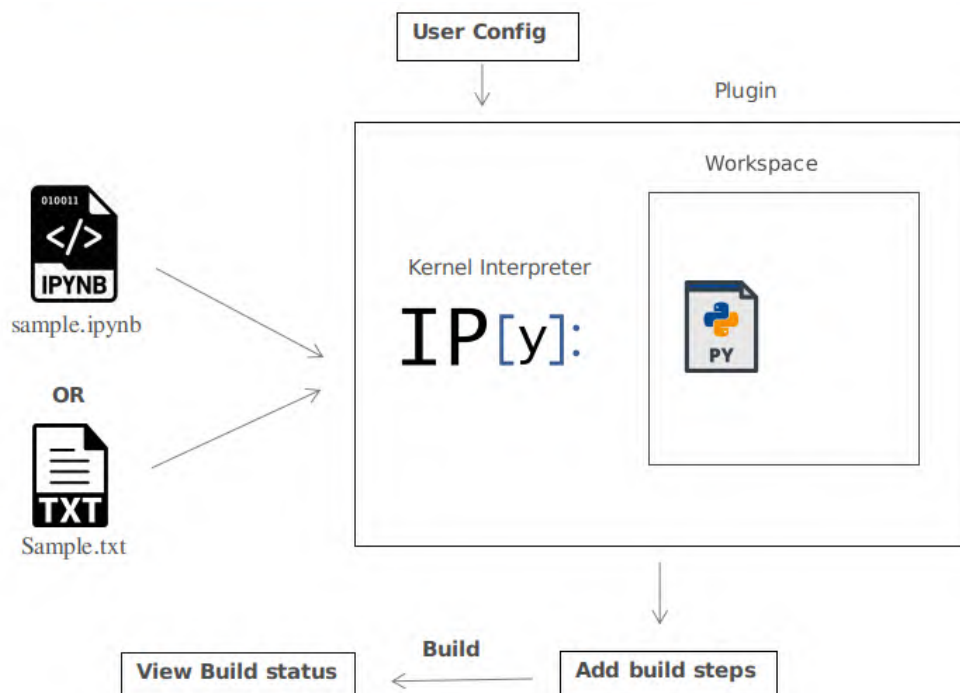
Project Abstract.

The main goal of this project is integrating Machine Learning workflow including Data preprocessing, Model Training, Evaluation and Prediction with Jenkins build tasks[1]. This plugin will be capable of executing code fragments via IPython kernel as currently supported by Jupyter. Version control will be handled as an added advantage of this project.

Project Description.

This Machine Learning plugin will ease workflow of building production grade artifacts or Machine Learning models. Users can directly upload the Jupyter notebook .ipynb file or create a custom python script and invoke the ipython kernel to execute all the given or extracted code segments.

Expected workflow of the plugin should follow the diagram below.



This plugin will be implemented with the following features.

- Kernel Configuration

Users are supposed to provide IPython kernel configurations which can be used to connect to a running kernel or start a new IPython kernel for the build purposes. Users may need different configurations when they connect to the kernel, this should be a must in the whole project. If not, Jenkins automatically uses the default configuration given by Apache Zeppelin.

- ☐ Python path
- ☐ Launch timeout
- ☐ Max result from kernel
- ☐ Gateway server address (eg : 127.0.0.1)
- ☐ Port number (eg : Available JVM gateway port)

- File parser

Since .ipynb, .py and .txt (contains python code) can be parsed to plugin for extracting Machine Learning steps according to the special notations provided in the comment section of code, plugin will have three types of parsers to get the files.

Especially for ipynb parsers, there will be a method NbConvertPy to convert to python script which will be easier for version control(optional) in the code segments[2]. All the parsers should implement the AbstractParser Builder which can initiate the user configuration including the predefined Machine Learning steps[3].

- ☐ Gathering data
- ☐ Preparing that data
- ☐ Choosing a model
- ☐ Training
- ☐ Evaluation
- ☐ Hyperparameter tuning
- ☐ Prediction

In order to achieve the identification of code segments, users should use the `# JenkinsParse_<Code fragment step>` comment on top of each segment.

Example :

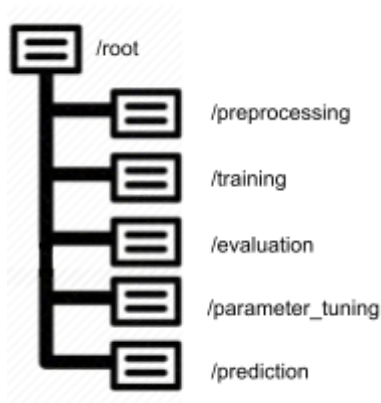
```
# JenkinsParse_Preprocessing
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
test_images = test_images.reshape(test_images.shape[0], 28, 28,
1).astype('float32')

# Normalizing the images to the range of [0., 1.]
train_images /= 255.
test_images /= 255.
```

- Extractor

Extractor will detect the code segments according to user configurations and keep every segment separately. While users define the build steps, it will automatically show off available code segments derived from the parent file.

Example :



- Custom script execution

In the middle, if a user needs to add a custom python script, this feature will allow them to add the code.

The following method will explain how the initiated kernel will last throughout the all build task finishes. Singleton class for interpreters will help to achieve this goal.

```

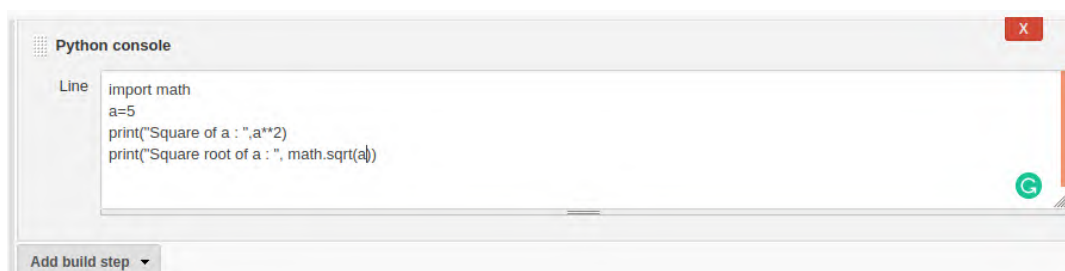
public static IPythonKernelInterpreter getInstance() throws Exception {
    // make sure one instance is initiated for a session.
    if(instance == null){
        try {
            instance = new IPythonKernelInterpreter();
        } catch (InterpreterException e) {
            e.printStackTrace();
        }
    }
    return instance;
}

```

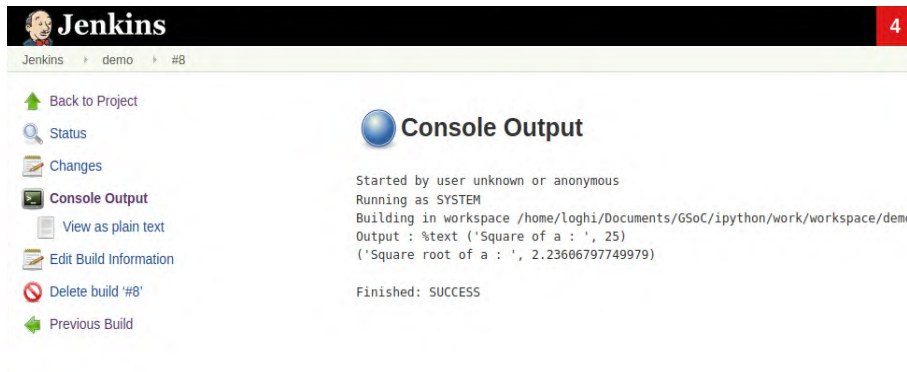
The work I have done so far related to this plugin :

<https://github.com/Loghijiaha/ipython>

Adding new build step :



Result :



The screenshot shows the Jenkins web interface for a build named 'demo' with ID '#8'. On the left, there is a sidebar with links: 'Back to Project', 'Status', 'Changes', 'Console Output' (highlighted), 'View as plain text', 'Edit Build Information', 'Delete build '#8'', and 'Previous Build'. The main area is titled 'Console Output' and displays the following text: 'Started by user unknown or anonymous', 'Running as SYSTEM', 'Building in workspace /home/loghi/Documents/GSoC/ipython/work/workspace/demo', 'Output : %text ('Square of a : ', 25)', '('Square root of a : ', 2.23606797749979)', and 'Finished: SUCCESS'.

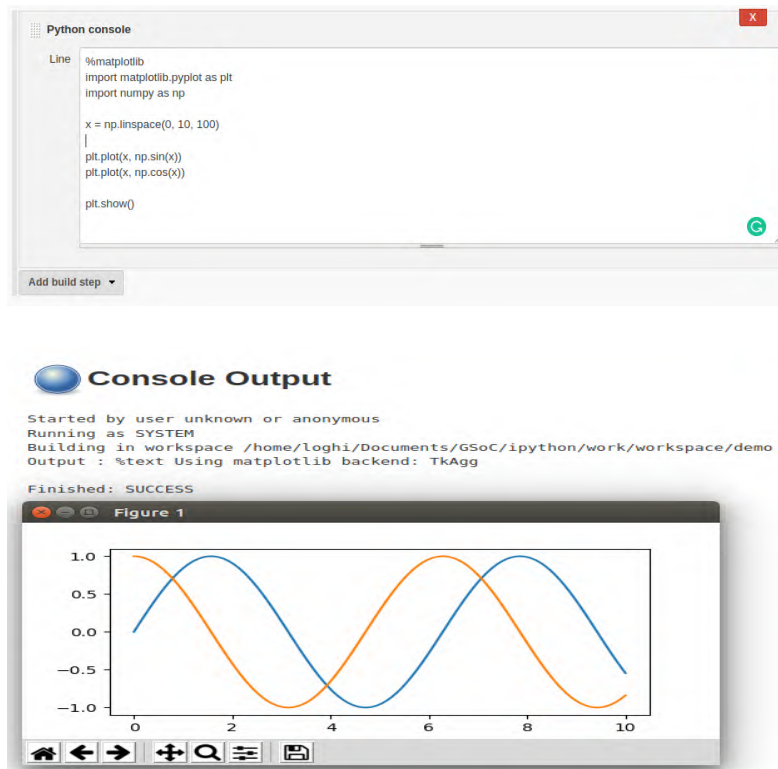
Pipeline test :

```
WorkflowJob p = jenkins.createProject(WorkflowJob.class, "p");
p.setDefinition(new CpsFlowDefinition(
    "node {\n" +
    "    python 'a=10'\n" +
    "    python 'a**2'\n" +
    "}");
    ));
```

- Visual embedded results

Above plugin I developed so far has to be improved with embedded visual data in the results. Since now, the results are viewed in console %text format and visual or plotting images are viewed separately unlike in Jupyter Notebook.

For example :



This visual output should be stucked into the Jenkins dashboard with some information about the graph (like datasets, X axis , Y axis , Legends etc). Evaluation metrics with visuals must be included in the build results also whether the build successes or not.

%matplotlib inline are also giving results with %img. These plain data can be rendered immediately using Jenkins API after build finishes.

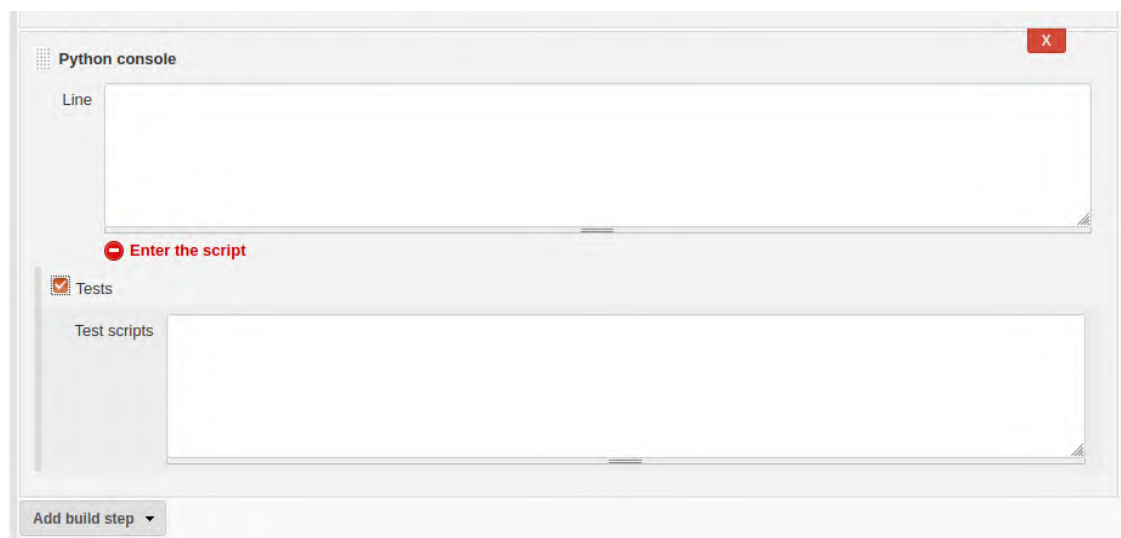
Console Output

```
Started by user unknown or anonymous
Running as SYSTEM
Building in workspace /home/loghi/Documents/GSoC/ipython/work/workspace/demo
Output : %img
iVBORw0KGgoAAAANSUgEugAAAYAAAD8CAYAAABzTgP2AAAABHNCsvQICAgIfAhkiAAAAAwSFlzAAAEgAACxIB0t1+/AAADl0RVh0U29mdHdhcm
UAbWF0cGxvdGxpYiB2ZXJzaW9uIDIuMi4zLCBodHRwOi8vbW0cGxvdGxpYi5vcmcvIxEbQAAIABJREFUeJzsvXd4VNe19
//Zo957AwkQEqiDABkw2NjG9Grc4pa4JHFyU9970+ybm+v3JvGN097k194c23HcbYwxHdtgY20KKEIdiSoJdaHeNfv3x5khAtRGM5ozWzqf55lHM6d
+4TnnrLPXWnstIaXEWMDAwMDAiklvAQYGBgYGroVhGAwMDAwMrsIwDAYGBgYGV2EYBGMdAw0DqzAMg4GBgYHBVRiGwcDAwMDgKgZDYGBgYGBwFYZhM
DAwMDC4CsMwGBgYGBhcbveAkZDeHi4nDZtm4yDAwMDJTi2LFjdVLKiOG2U9IwTJs2jezsbL1lGBgYGCiFE0LCSLYzXEkGBgYGBldhGAYDAwMDg6s
wDIOBgYGBwVUYhsHAWMDA4CoMw2BgYGBgcBUOMQxCiL8LIWqEEHmDrBdCiN8IIUqFEKEEHP7rXtYCFfi+TzsCD0GBgYGBqPHUSOG54FVQ6xfDcywf
B4H/ggghAgFngIWAPOBp4QQIQ75ZGBgYGAwChwyj0FK+ZEQYtoQm2wE
```

Post build will have a JSON generator for creating **result.json** which can be used to analyse build statistics using existing (spotlight or cucumber) reporting plugins.

- Tests

Users can run tests for each code snippet with pre-tested codelets. The parameter values that can be used for testing inputs similar in Scriptler. Isolated code areas will be given to keep the code clean and comprehensive. It will be a foldable component under the code block. If tests are available, then Jenkins will report for the test results to the dashboard.



Benefits to community

This plugin will provide a smooth and effortless interface for data scientists to build production grade ML workflow. Data science community will be much benefited through this plugin with Jenkins automation. It will attract most of the dev-community to use this plugin and increase the intention to use and contribute to Jenkins. This plugin shall give motivation to integrate other data science tools and kernels with Jenkins.

This project will be a remarkable milestone in my Computer Science and Engineering field, if I get selected because Machine Learning is one of the emerging technical fields nowadays. When someone gets benefits through my open source contributions, it would be a great pleasure.

Project Deliverables

- An IPython plugin with pipeline compatible
- Improved config.jelly for the plugin
- File parsers for ipynb, py and text(containing python code) files
- Code segment extractor
- Implement a code editor similar to Scriptler *
- Output console with static visual components
- Version control for each code segment (optional)

*

Code editors	Compatible with Script security plugin	Support languages
Scriptler	Yes	Groovy
Jupyter's code editor	No	Python, R
New editor	No	Python

Proposed Schedule

March 16 - March 31 (Application Period)

- Discuss with mentors about the proposed plan till the application period closes
- Get more familiar with the Jenkins community and testing on a currently developed simple ipython plugin.

April 1 - April 30 (Acceptance Waiting Period)

- Analyse Active choice plugin's codebase to get to know how it is functioning
- Find out the possibilities for integrating Jupyter book instead of new code editor
- Writing a Readme for the POC
- Content writing about the need of Jenkins and Machine Learning integration.
- Research on different case in Jenkins support for Machine Learning

May 04 - June 1 (Community Bonding Period)

- This is an excellent time to introduce myself to the entire community
- Joining with right mailing list
- Asking guidelines about contribution
- Preparing and planning for the project timeline with mentors
- Start improving the ipython kernel interpreter

June 1 - June 29 (Coding period 1)

- Implement the configuration part of Ipython kernel interpreter
- Implement file parser
- Improve current config.jelly
- Improve form validations
- Increase the test coverage and documentation

July 3 - July 27 (Coding period 2)

- Implement code editor (Similar to Sceiptler or Jupyter book)
- Extend the editor for test purposes with pre-tested codelets.
- Testing and bug fixing
- Improve the documentation
- Discuss with mentors about phase 3 coding

July 31 - Aug 24 (Coding period 3)

- Design and implement code segment extractor
- Implement JSON generator for the results
- Integrate whole previous work and testing
- Git integration for code segments (optional)
- User documentation and examples

Future Improvements

- Writing developer documentation
- Git integration on plugin if not implemented during coding phase.
- Implementing Interactive visualizer for data sets and results.
- Community discussions and design to integrate other kernels with Jenkins

Continued Involvement

I will continually support Jenkins contributions. I have already contributed to git plugin and git-client plugin before the GSoC initiation. I will support developers and students who are interested in contributions. As I have experience in mentoring junior students in open source contributions, I would like to contribute proactively in Jenkins.

Conflict of Interests or Commitment

There is no conflict with this summer of code program with my University. Literally, my University encourages students to participate in GSoC every year. Due to the Covid-19 pandemic, there will be minor changes in the schedule which will definitely not affect Google Summer of Code programme.

Major Challenges foreseen

- I faced some challenges with version conflict between jenkins-core and Apache Zeppelin.
- Integrating javascript with jelly to design will be a challenge when designing better UI for the plugin.
- Implementation of file parser and code editor will be challenging tasks.
- Designing the code editor for python should ensure that it is compatible with the Script security plugin.

References

- [1] Project idea for machine learning plugin
<https://jenkins.io/projects/gsoc/2020/project-ideas/machine-learning/>
- [2] Version control for Jupyter Notebook
<https://nextjournal.com/schmudde/how-to-version-control-jupyter>
- [3] Machine Learning steps
<https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e>
- [4] Code snippets for examples
<https://colab.research.google.com/notebooks/intro.ipynb>

Relevant Background Experience

- I have been using Jupyter Notebook and google Colab for my ML projects in my school. <https://github.com/Loghijiaha/Tamil-TextSummarization>
- I have industrial experience in Java from my internship. I also have experience in OOP concepts which is good Software engineering discipline.

Personal

- I am a final year student at University of Moratuwa from Sri Lanka (UTC +5.30).
- The interesting feature of Jenkins is that it is used to accomplish all sorts of tasks including building, testing and deploying. No need to use separate frameworks.
- I learned Jenkins architectures, security measures and plugin developer docs from the official website when contributing to git and git-client plugin. When I was preparing for an interview, I learned interview questions also.

Availability and commitments

There are no commitments and conflicts with participating in GSoC 2020. I can finish my milestones within the timeline. I will be available on the weekends also that would be sufficient to work on the project.

Experience

Free Software Experience/Contributions

- Helped to improve test quality(JUnit3 → JUnit4) in **Git plugin** and **git-client plugins**(Active) for **Jenkinsci**
 - ❑ Merged pull requests
 - ❑ <https://github.com/jenkinsci/git-plugin/pull/834>
 - ❑ <https://github.com/jenkinsci/git-client-plugin/pull/509>
 - ❑ <https://github.com/jenkinsci/git-client-plugin/pull/504>
 - ❑ <https://github.com/jenkinsci/git-client-plugin/pull/503>
 - ❑ Open pull requests
 - ❑ <https://github.com/jenkinsci/git-plugin/pull/846>
- Contributed for SpaCy
 - ❑ Documentation and improvement(Merged)
 - ❑ <https://github.com/sharmalab/Datascope/pull/87>
 - ❑ <https://github.com/explosion/spaCy/pull/3194>
 - ❑ <https://github.com/explosion/spaCy/pull/3154>

Language Skill Set

- | | |
|--------------|-----|
| • Python | 4/5 |
| • Java | 4/5 |
| • Maven | 4/5 |
| • Git | 4/5 |
| • Javascript | 3/5 |

Related Research and Work Experience

- Global Instep internship in Infosys Ltd 2019 - Worked with **Apache pulsar** in spring boot application.
- Assisted in research in Natural Language Processing 2019.