mFlow

Benjamin Marlin

Sep 06, 2021

CONTENTS:

1 Installation **3**

2 Examples **5**
- 2.1 mFlow.Blocks Package 5
- 2.2 mFlow.Utilities Package 6
- 2.3 mFlow.Workflow Package 6

3 Contributors **11**

4 License **13**

5 Acknowledgments **15**

6 Indices and tables **17**

Python Module Index **19**

Index **21**

mFlow is a Python module for specifying and executing machine learning experimentation workflows. It organizes both data transformations and common experimental procedures into workflow blocks. mFlow focuses on the multi-core parallel setting and interoperates with machine learning model implementations that follow the scikit-learn model class structure. mFlow can also interoperate with Apache Spark and the MD2K Cerebral Cortex library to split workflow execution across a distributed pre-processing and feature extraction phase followed by a multi-core parallel model estimation phase. mFlow uses Pandas dataframes as the primary data structure for representing data sets and results.

mFlow supports a range of experimental workflows with a focus on multi-level data where individual data cases are nested within groups. It supports partitioning data sets both at the instance level and at the group level for most workflows. Experimental designs currently supported by mFlow include:

- Train/Test performance assessment with instance-level partitioning
- Train/test performance assessment with group-level partitioning
- Cross-validation performance assessment with instance-level partitioning
- Cross-validation performance assessment with group-level partitioning
- Within group Train/Test performance assessment with random partitioning within groups
- Within group Train/Test performance assessment with sequential partitioning within groups

mFlow also includes a library of example workflows illustrating the application of data transformations and experimental workflows on open data sets in the mobile health and activity recognition domains.

CHAPTER

ONE

INSTALLATION

mFlow requires Python 3.6 or higher. To install mFlow using pip, clone this repository and run pip install:

```
git clone https://github.com/mlds-lab/mFlow.git
pip3 install ./mFlow
```

CHAPTER

TWO

EXAMPLES

See the Examples directory for a list of mFlow examples that can be run locally or launched in Google Colab.

2.1 mFlow.Blocks Package

2.1.1 mFlow.Blocks.ccwrapper Module

mFlow.Blocks.ccwrapper.**cc_to_pandas**(**args*, ***kwargs*)

mFlow.Blocks.ccwrapper.**ccwrapper**(**args*, ***kwargs*)

2.1.2 mFlow.Blocks.data_loader_extrasensory Module

mFlow.Blocks.data_loader_extrasensory.**extrasensory_data_loader**(***kwargs*)

2.1.3 mFlow.Blocks.data_loader_wesad Module

mFlow.Blocks.data_loader_wesad.**wesad_data_loader**(***kwargs*)

2.1.4 mFlow.Blocks.experimental_protocol Module

mFlow.Blocks.experimental_protocol.**ExpCV**(**args*, ***kwargs*)

mFlow.Blocks.experimental_protocol.**ExpTrainTest**(**args*, ***kwargs*)

mFlow.Blocks.experimental_protocol.**ExpWithin**(**args*, ***kwargs*)

mFlow.Blocks.experimental_protocol.**addTarget**(**args*, ***kwargs*)

mFlow.Blocks.experimental_protocol.**df_to_sk**(*df*)

2.1.5 mFlow.Blocks.filter Module

mFlow.Blocks.filter.**ColumnSelectFilter**(**args*, ***kwargs*)

mFlow.Blocks.filter.**MisingDataColumnFilter**(**args*, ***kwargs*)

mFlow.Blocks.filter.**MisingDataRowFilter**(**args*, ***kwargs*)

mFlow.Blocks.filter.**MisingLabelFilter**(**args*, ***kwargs*)

mFlow.Blocks.filter.**Take**(**args*, ***kwargs*)

2.1.6 mFlow.Blocks.imputer Module

mFlow.Blocks.imputer.**Imputer**(**args*, ***kwargs*)

2.1.7 mFlow.Blocks.normalizer Module

mFlow.Blocks.normalizer.**Normalizer**(**args*, ***kwargs*)

2.1.8 mFlow.Blocks.results_analysis Module

mFlow.Blocks.results_analysis.**DataYieldReport**(**args*, ***kwargs*)

mFlow.Blocks.results_analysis.**ResultsCVSummarize**(**args*, ***kwargs*)

mFlow.Blocks.results_analysis.**ResultsConcat**(**args*, ***kwargs*)

2.2 mFlow.Utilities Package

2.2.1 mFlow.Utilities.utilities Module

mFlow.Utilities.utilities.**getCacheDir**()

mFlow.Utilities.utilities.**getDataDir**()

mFlow.Utilities.utilities.**getmFlowUserHome**()

2.3 mFlow.Workflow Package

2.3.1 mFlow.Workflow.compute_graph Module

class mFlow.Workflow.compute_graph.**node**(*function=None*, *args=[]*, *kwargs={}*, *name=None*, *parents=[]*)

Bases: object

__init__(*function=None*, *args=[]*, *kwargs={}*, *name=None*, *parents=[]*)

get_args()

get_kwargs()

run()

class mFlow.Workflow.compute_graph.**pipelineNode**(*initGraph*, *node_list*, *id*)
Bases: object

__init__(*initGraph*, *node_list*, *id*)

run()

2.3.2 mFlow.Workflow.scheduler Module

mFlow.Workflow.scheduler.**run**(*flow*, *backend='sequential'*, *num_workers=1*, *monitor=False*, *from_scratch=False*)

mFlow.Workflow.scheduler.**run_parallel**(*flow*, *data=None*, *backend='multithread'*, *num_workers=1*, *monitor=False*, *from_scratch=False*)

mFlow.Workflow.scheduler.**run_parallel_pipeline**(*flow*, *data=None*, *backend='multithread_pipeline'*, *num_workers=1*, *monitor=False*, *from_scratch=False*, *refresh_rate=0.05*)

mFlow.Workflow.scheduler.**run_pipeline**(*flow*, *data=None*, *monitor=False*, *from_scratch=False*)

mFlow.Workflow.scheduler.**run_sequential**(*flow*, *data=None*, *monitor=False*, *from_scratch=False*)

2.3.3 mFlow.Workflow.workflow Module

class mFlow.Workflow.workflow.**workflow**(*nodes={}*)
Bases: object

__init__(*nodes={}*)
Workflow constructor

Parameters

- **nodes** (*dict*) – a dictionary of workflow output nodes. Keys are used as tags
- **outputs.** (*to identify*) –

add(*nodes*)
Adds the given list of workflow nodes to the workflow, along with all of their ancestors. Sets the out_tag property of the nodes if not already set.

Parameters **nodes** – a list of workflow nodes (or a single workflow node).

add_edge(*node_from*, *node_to*)
Add a directed edge between nodes

Parameters

- **node_from** – ID of from node
- **node_to** – ID of to node

add_node(*name*, *block*)
Adds a node to the workflow with a specified name and block.

Parameters

- **name** (*str*) – unique name for workflow node
- **block** – the workflow block for this node

add_output(*node*, *tag*)
Adds the given workflow node to the set of workflow outputs with the given tag, then adds all of the node's ancestors to the workflow.

Parameters

- **node** – a workflow node
- **tag** – a string to identifiy the node

add_pipeline_node(*id*, *plNode*)
Add a pipline node to pipeline workflow graph

Parameters

- **id** (*string*) – id for the node
- **plNode** – pipelined workflow node

draw(*refresh=True*)
Display the workflow graph in a Jupyter notebook

Parameters **refresh** (*bool*) – If True, clear the cell before drawing.

drawPipelined(*refresh=True*)
Display pipilined workflow graph in a Jupyter notebook.

Parameters **refresh** (*bool*) – If True, clear the cell before drawing.

pipeline(*initGraph*)
Convert the given workflow graph to a pipelined representation where chanins in the workflow graph are replaced by single node.

Parameters **initGraph** – A workflow graph.

pipelineGraphCreate(*process_dict*)
Convert a dictionary of pipelined workflow nodes to a pipelined workflow graph.

Parameters **process_dict** – dictionary of pipelined workflow nodes.

recursive_add_node(*node*)
Adds the given list of workflow nodes to the workflow, along with all of their ancestors. If the node is already in the workflow, does not add duplicates.

Parameters **nodes** – a list of workflow nodes.

remove(*nodes*)
Removes the given workflow nodes from the workflow, along with all of their descendents.

Parameters **nodes** – a list of workflow nodes (or a single workflow node).

run(*backend='sequential'*, *num_workers=1*, *monitor=False*, *from_scratch=False*)
Run the workflow with the specified backend scheduler.

Parameters

- **backend** (*string*) – The type of scheduling backend to use (sequential | multithread | multiprocess | pipeline | multithread_pipeline | multiprocess_pipeline). See mFlow.Workflow.scheduler for documentation.
- **num_workers** (*int*) – Number of workers to use in parallel backends
- **monitor** (*bool*) – If True, use graphical execution monitorin for Jupyter notebooks
- **from_scratch** (*bool*) – If True, run the workflow from scratch, discarding any cached results.

set_status(*node*, *status*)
Set the status of a workflow node

Parameters

- **node** – a workflow node
- **status** (*string*) – scheduled | notscheduled | running | done

CHAPTER

THREE

CONTRIBUTORS

Link to the list of contributors who participated in this project.

CHAPTER

FOUR

LICENSE

This project is licensed under the BSD 2-Clause - see the license file for details.

CHAPTER

FIVE

ACKNOWLEDGMENTS

- National Institutes of Health - Big Data to Knowledge Initiative Grant: 1U54EB020404
- National Science Foundation Grant: 1823283

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

mFlow.Blocks.ccwrapper, 5
mFlow.Blocks.data_loader_extrasensory, 5
mFlow.Blocks.data_loader_wesad, 5
mFlow.Blocks.experimental_protocol, 5
mFlow.Blocks.filter, 6
mFlow.Blocks.imputer, 6
mFlow.Blocks.normalizer, 6
mFlow.Blocks.results_analysis, 6
mFlow.Utilities.utilities, 6
mFlow.Workflow.compute_graph, 6
mFlow.Workflow.scheduler, 7
mFlow.Workflow.workflow, 7

INDEX

Symbols

__init__() (*mFlow.Workflow.compute_graph.node method*), 6
__init__() (*mFlow.Workflow.compute_graph.pipelineNode method*), 7
__init__() (*mFlow.Workflow.workflow.workflow method*), 7

A

add() (*mFlow.Workflow.workflow.workflow method*), 7
add_edge() (*mFlow.Workflow.workflow.workflow method*), 7
add_node() (*mFlow.Workflow.workflow.workflow method*), 7
add_output() (*mFlow.Workflow.workflow.workflow method*), 7
add_pipeline_node() (*mFlow.Workflow.workflow.workflow method*), 8
addTarget() (*in module mFlow.Blocks.experimental_protocol*), 5

C

cc_to_pandas() (*in module mFlow.Blocks.ccwrapper*), 5
ccwrapper() (*in module mFlow.Blocks.ccwrapper*), 5
ColumnSelectFilter() (*in module mFlow.Blocks.filter*), 6

D

DataYieldReport() (*in module mFlow.Blocks.results_analysis*), 6
df_to_sk() (*in module mFlow.Blocks.experimental_protocol*), 5
draw() (*mFlow.Workflow.workflow.workflow method*), 8
drawPipelined() (*mFlow.Workflow.workflow.workflow method*), 8

E

ExpCV() (*in module mFlow.Blocks.experimental_protocol*), 5
ExpTrainTest() (*in module mFlow.Blocks.experimental_protocol*), 5
ExpWithin() (*in module mFlow.Blocks.experimental_protocol*), 5
extrasensory_data_loader() (*in module mFlow.Blocks.data_loader_extrasensory*), 5

G

get_args() (*mFlow.Workflow.compute_graph.node method*), 6
get_kwargs() (*mFlow.Workflow.compute_graph.node method*), 6
getCacheDir() (*in module mFlow.Utilities.utilities*), 6
getDataDir() (*in module mFlow.Utilities.utilities*), 6
getmFlowUserHome() (*in module mFlow.Utilities.utilities*), 6

I

Imputer() (*in module mFlow.Blocks.imputer*), 6

M

mFlow.Blocks.ccwrapper
    module, 5
mFlow.Blocks.data_loader_extrasensory
    module, 5
mFlow.Blocks.data_loader_wesad
    module, 5
mFlow.Blocks.experimental_protocol
    module, 5
mFlow.Blocks.filter
    module, 6
mFlow.Blocks.imputer
    module, 6
mFlow.Blocks.normalizer
    module, 6
mFlow.Blocks.results_analysis
    module, 6
mFlow.Utilities.utilities
    module, 6
mFlow.Workflow.compute_graph
    module, 6

mFlow.Workflow.scheduler
    module, 7
mFlow.Workflow.workflow
    module, 7
MisingDataColumnFilter() (*in module mFlow.Blocks.filter*), 6
MisingDataRowFilter() (*in module mFlow.Blocks.filter*), 6
MisingLabelFilter() (*in module mFlow.Blocks.filter*), 6
module
    mFlow.Blocks.ccwrapper, 5
    mFlow.Blocks.data_loader_extrasensory, 5
    mFlow.Blocks.data_loader_wesad, 5
    mFlow.Blocks.experimental_protocol, 5
    mFlow.Blocks.filter, 6
    mFlow.Blocks.imputer, 6
    mFlow.Blocks.normalizer, 6
    mFlow.Blocks.results_analysis, 6
    mFlow.Utilities.utilities, 6
    mFlow.Workflow.compute_graph, 6
    mFlow.Workflow.scheduler, 7
    mFlow.Workflow.workflow, 7

N

node (*class in mFlow.Workflow.compute_graph*), 6
Normalizer() (*in module mFlow.Blocks.normalizer*), 6

P

pipeline() (*mFlow.Workflow.workflow.workflow method*), 8
pipelineGraphCreate() (*mFlow.Workflow.workflow.workflow method*), 8
pipelineNode (*class in mFlow.Workflow.compute_graph*), 6

R

recursive_add_node() (*mFlow.Workflow.workflow.workflow method*), 8
remove() (*mFlow.Workflow.workflow.workflow method*), 8
ResultsConcat() (*in module mFlow.Blocks.results_analysis*), 6
ResultsCVSummarize() (*in module mFlow.Blocks.results_analysis*), 6
run() (*in module mFlow.Workflow.scheduler*), 7
run() (*mFlow.Workflow.compute_graph.node method*), 6
run() (*mFlow.Workflow.compute_graph.pipelineNode method*), 7
run() (*mFlow.Workflow.workflow.workflow method*), 8
run_parallel() (*in module mFlow.Workflow.scheduler*), 7
run_parallel_pipeline() (*in module mFlow.Workflow.scheduler*), 7
run_pipeline() (*in module mFlow.Workflow.scheduler*), 7
run_sequential() (*in module mFlow.Workflow.scheduler*), 7

S

set_status() (*mFlow.Workflow.workflow.workflow method*), 8

T

Take() (*in module mFlow.Blocks.filter*), 6

W

wesad_data_loader() (*in module mFlow.Blocks.data_loader_wesad*), 5
workflow (*class in mFlow.Workflow.workflow*), 7