

---

# **PipelineX**

***Release 0.7.5***

**Yusuke Minami**

**Nov 11, 2022**



## CONTENTS:

<b>1</b>	<b>PipelineX</b>	<b>1</b>
<b>2</b>	<b>PipelineX Overview</b>	<b>3</b>
<b>3</b>	<b>Install PipelineX</b>	<b>5</b>
3.1	[Option 1] Install from the PyPI . . . . .	5
3.2	[Option 2] Development install . . . . .	5
3.3	Prepare development environment for PipelineX . . . . .	5
3.4	Prepare Docker environment for PipelineX . . . . .	6
<b>4</b>	<b>Getting Started with PipelineX</b>	<b>7</b>
4.1	Kedro 0.17.x Starter projects . . . . .	7
4.2	Example/Demo Projects tested with Kedro 0.16.x . . . . .	7
<b>5</b>	<b>HatchDict: Python in YAML/JSON</b>	<b>9</b>
5.1	Python objects in YAML/JSON . . . . .	9
5.1.1	Introduction to YAML . . . . .	9
5.1.2	Python tags in YAML . . . . .	11
5.1.3	Alternative to Python tags in YAML . . . . .	12
5.2	Anchor-less aliasing in YAML/JSON . . . . .	13
5.2.1	Aliasing in YAML . . . . .	13
5.2.2	Alternative to aliasing in YAML . . . . .	14
5.3	Python expression in YAML/JSON . . . . .	14
<b>6</b>	<b>Introduction to Kedro</b>	<b>17</b>
6.1	Why the unified data interface framework is needed . . . . .	17
6.2	Kedro overview . . . . .	20
<b>7</b>	<b>Flex-Kedro: Kedro plugin for flexible config</b>	<b>23</b>
7.1	Flex-Kedro-Pipeline: Kedro plugin for quicker pipeline set up . . . . .	23
7.1.1	Dict for nodes . . . . .	23
7.1.2	Sequential nodes . . . . .	23
7.1.3	Decorators without using the method . . . . .	24
7.2	Flex-Kedro-Context: Kedro plugin for YAML lovers . . . . .	25
7.2.1	Define Kedro pipelines in <code>parameters.yml</code> . . . . .	25
7.2.2	Configure Kedro run config in <code>parameters.yml</code> . . . . .	25
7.2.3	Use <code>HatchDict</code> feature in <code>parameters.yml</code> . . . . .	26
7.2.4	Enable caching for Kedro DataSets in <code>catalog.yml</code> . . . . .	26
7.2.5	Use <code>HatchDict</code> feature in <code>catalog.yml</code> . . . . .	26
<b>8</b>	<b>MLflow-on-Kedro: Kedro plugin for MLflow users</b>	<b>27</b>

8.1	How to use MLflow from Kedro projects	27
8.2	Comparison with <code>kedro-mlflow</code> package	29
<b>9</b>	<b>Kedro-Extras: Kedro plugin to use various Python packages</b>	<b>31</b>
9.1	Additional Kedro datasets (data interface sets)	31
9.2	Additional function decorators for benchmarking	32
9.3	Use with PyTorch	33
9.4	Use with PyTorch Ignite	35
9.5	Use with OpenCV	37
<b>10</b>	<b>Story behind PipelineX</b>	<b>39</b>
<b>11</b>	<b>Author</b>	<b>41</b>
<b>12</b>	<b>Contributors are welcome!</b>	<b>43</b>
12.1	How to contribute	43
12.2	Contributor list	43
<b>13</b>	<b>pipelineX</b>	<b>45</b>
13.1	pipelineX package	46
13.1.1	Subpackages	46
13.1.1.1	pipelineX.extras package	46
13.1.1.1.1	Subpackages	46
13.1.1.1.1.1	pipelineX.extras.datasets package	46
13.1.1.1.1.2	Subpackages	46
13.1.1.1.1.3	pipelineX.extras.datasets.httpx package	46
13.1.1.1.1.4	Submodules	46
13.1.1.1.1.5	pipelineX.extras.datasets.httpx.async_api_dataset module	46
13.1.1.1.1.6	pipelineX.extras.datasets.opencv package	47
13.1.1.1.1.7	Submodules	47
13.1.1.1.1.8	pipelineX.extras.datasets.opencv.images_dataset module	47
13.1.1.1.1.9	pipelineX.extras.datasets.pandas package	47
13.1.1.1.1.10	Submodules	47
13.1.1.1.1.11	pipelineX.extras.datasets.pandas.csv_local module	47
13.1.1.1.1.12	pipelineX.extras.datasets.pandas.efficient_csv_local module	49
13.1.1.1.1.13	pipelineX.extras.datasets.pandas.fixed_width_csv_dataset module	49
13.1.1.1.1.14	pipelineX.extras.datasets.pandas.histogram module	50
13.1.1.1.1.15	pipelineX.extras.datasets.pandas.pandas_cat_matrix module	50
13.1.1.1.1.16	pipelineX.extras.datasets.pandas.pandas_describe module	51
13.1.1.1.1.17	pipelineX.extras.datasets.pandas_profiling package	51
13.1.1.1.1.18	Submodules	51
13.1.1.1.1.19	pipelineX.extras.datasets.pandas_profiling.pandas_profiling module	51
13.1.1.1.1.20	pipelineX.extras.datasets.pillow package	52
13.1.1.1.1.21	Submodules	52
13.1.1.1.1.22	pipelineX.extras.datasets.pillow.images_dataset module	52
13.1.1.1.1.23	pipelineX.extras.datasets.requests package	53
13.1.1.1.1.24	Submodules	53
13.1.1.1.1.25	pipelineX.extras.datasets.requests.api_dataset module	53
13.1.1.1.1.26	pipelineX.extras.datasets.seaborn package	56
13.1.1.1.1.27	Submodules	56
13.1.1.1.1.28	pipelineX.extras.datasets.seaborn.seaborn_pairplot module	56
13.1.1.1.1.29	pipelineX.extras.datasets.torchvision package	56
13.1.1.1.1.30	Submodules	56
13.1.1.1.1.31	pipelineX.extras.datasets.torchvision.iterable_images_dataset module	56

13.1.1.1.1.32	Submodules	57
13.1.1.1.1.33	pipelinex.extras.datasets.core module	57
13.1.1.1.1.34	pipelinex.extras.decorators package	57
13.1.1.1.1.35	Submodules	57
13.1.1.1.1.36	pipelinex.extras.decorators.decorators module	57
13.1.1.1.1.37	pipelinex.extras.decorators.memory_profiler module	57
13.1.1.1.1.38	pipelinex.extras.decorators.nvml_profiler module	57
13.1.1.1.1.39	pipelinex.extras.decorators.pandas_decorators module	58
13.1.1.1.1.40	pipelinex.extras.hooks package	58
13.1.1.1.1.41	Submodules	58
13.1.1.1.1.42	pipelinex.extras.hooks.add_catalog_dict module	58
13.1.1.1.1.43	pipelinex.extras.hooks.add_transformers module	58
13.1.1.1.1.44	pipelinex.extras.ops package	59
13.1.1.1.1.45	Subpackages	59
13.1.1.1.1.46	pipelinex.extras.ops.ignite package	59
13.1.1.1.1.47	Subpackages	59
13.1.1.1.1.48	pipelinex.extras.ops.ignite.declaratives package	59
13.1.1.1.1.49	Submodules	59
13.1.1.1.1.50	pipelinex.extras.ops.ignite.declaratives.declarative_trainer module	59
13.1.1.1.1.51	pipelinex.extras.ops.ignite.handlers package	61
13.1.1.1.1.52	Submodules	61
13.1.1.1.1.53	pipelinex.extras.ops.ignite.handlers.flexible_checkpoint module	61
13.1.1.1.1.54	pipelinex.extras.ops.ignite.handlers.time_limit module	62
13.1.1.1.1.55	pipelinex.extras.ops.ignite.metrics package	62
13.1.1.1.1.56	Submodules	62
13.1.1.1.1.57	pipelinex.extras.ops.ignite.metrics.cohen_kappa_score module	62
13.1.1.1.1.58	pipelinex.extras.ops.ignite.metrics.fbeta_score module	63
13.1.1.1.1.59	pipelinex.extras.ops.ignite.metrics.utils module	64
13.1.1.1.1.60	Submodules	64
13.1.1.1.1.61	pipelinex.extras.ops.allennlp_ops module	64
13.1.1.1.1.62	pipelinex.extras.ops.parse_ops module	64
13.1.1.1.1.63	pipelinex.extras.ops.numpy_ops module	64
13.1.1.1.1.64	pipelinex.extras.ops.opencv_ops module	64
13.1.1.1.1.65	pipelinex.extras.ops.pandas_ops module	68
13.1.1.1.1.66	pipelinex.extras.ops.pytorch_ops module	74
13.1.1.1.1.67	pipelinex.extras.ops.shap_ops module	86
13.1.1.1.1.68	pipelinex.extras.ops.skimage_ops module	86
13.1.1.1.1.69	pipelinex.extras.ops.sklearn_ops module	87
13.1.1.1.1.70	pipelinex.extras.transformers package	89
13.1.1.2	pipelinex.flex_kedro package	89
13.1.1.2.1	Subpackages	89
13.1.1.2.1.1	pipelinex.flex_kedro.context package	89
13.1.1.2.1.2	Submodules	89
13.1.1.2.1.3	pipelinex.flex_kedro.context.context module	89
13.1.1.2.1.4	pipelinex.flex_kedro.context.flexible_catalog_context module	89
13.1.1.2.1.5	pipelinex.flex_kedro.context.flexible_context module	89
13.1.1.2.1.6	pipelinex.flex_kedro.context.flexible_parameters_context module	89
13.1.1.2.1.7	pipelinex.flex_kedro.context.flexible_run_context module	91
13.1.1.2.1.8	pipelinex.flex_kedro.context.save_pipeline_json_context module	93
13.1.1.2.1.9	pipelinex.flex_kedro.pipeline package	93
13.1.1.2.1.10	Submodules	93
13.1.1.2.1.11	pipelinex.flex_kedro.pipeline.pipeline module	93
13.1.1.2.1.12	pipelinex.flex_kedro.pipeline.sub_pipeline module	94
13.1.1.2.2	Submodules	95

13.1.1.2.3	pipeline.flex_kedro.configure module . . . . .	95
13.1.1.3	pipeline.hatch_dict package . . . . .	95
13.1.1.3.1	Submodules . . . . .	95
13.1.1.3.2	pipeline.hatch_dict.hatch_dict module . . . . .	95
13.1.1.4	pipeline.mlflow_on_kedro package . . . . .	97
13.1.1.4.1	Subpackages . . . . .	97
13.1.1.4.1.1	pipeline.mlflow_on_kedro.datasets package . . . . .	97
13.1.1.4.1.2	Subpackages . . . . .	97
13.1.1.4.1.3	pipeline.mlflow_on_kedro.datasets.mlflow package . . . . .	97
13.1.1.4.1.4	Submodules . . . . .	97
13.1.1.4.1.5	pipeline.mlflow_on_kedro.datasets.mlflow.mlflow_dataset module . . . . .	97
13.1.1.4.1.6	pipeline.mlflow_on_kedro.decorators package . . . . .	99
13.1.1.4.1.7	Submodules . . . . .	99
13.1.1.4.1.8	pipeline.mlflow_on_kedro.decorators.mlflow_logger module . . . . .	99
13.1.1.4.1.9	pipeline.mlflow_on_kedro.hooks package . . . . .	99
13.1.1.4.1.10	Subpackages . . . . .	99
13.1.1.4.1.11	pipeline.mlflow_on_kedro.hooks.mlflow package . . . . .	99
13.1.1.4.1.12	Submodules . . . . .	99
13.1.1.4.1.13	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_artifacts_logger module . . . . .	99
13.1.1.4.1.14	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger module . . . . .	100
13.1.1.4.1.15	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger module . . . . .	101
13.1.1.4.1.16	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger module . . . . .	102
13.1.1.4.1.17	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger module . . . . .	103
13.1.1.4.1.18	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger module . . . . .	104
13.1.1.4.1.19	pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_utils module . . . . .	105
13.1.1.4.1.20	pipeline.mlflow_on_kedro.transformers package . . . . .	105
13.1.1.4.1.21	Subpackages . . . . .	105
13.1.1.4.1.22	pipeline.mlflow_on_kedro.transformers.mlflow package . . . . .	105
13.1.1.4.1.23	Submodules . . . . .	105
13.1.1.4.1.24	pipeline.mlflow_on_kedro.transformers.mlflow.mlflow_io_time_logger module . . . . .	105
13.1.2	Submodules . . . . .	106
13.1.3	pipeline.utils module . . . . .	106
<b>14</b>	<b>Indices and tables</b>	<b>109</b>
	<b>Python Module Index</b>	<b>111</b>
	<b>Index</b>	<b>113</b>

## PIPELINEX

PipelineX: Python package to build ML pipelines for experimentation with Kedro, MLflow, and more





## PIPELINEX OVERVIEW

PipelineX is a Python package to build ML pipelines for experimentation with Kedro, MLflow, and more

PipelineX provides the following options which can be used independently or together.

- HatchDict: Python in YAML/JSON

`HatchDict` is a Python dict parser that enables you to include Python objects in YAML/JSON files.

Note: `HatchDict` can be used with or without Kedro.

- Flex-Kedro: Kedro plugin for flexible config

- Flex-Kedro-Pipeline: Kedro plugin for quicker pipeline set up

- Flex-Kedro-Context: Kedro plugin for YAML lovers

- MLflow-on-Kedro: Kedro plugin for MLflow users

`MLflow-on-Kedro` provides integration of Kedro with `MLflow` with Kedro DataSets and Hooks.

Note: You do not need to install MLflow if you do not use.

- Kedro-Extras: Kedro plugin to use various Python packages

`Kedro-Extras` provides Kedro DataSets, decorators, and wrappers to use various Python packages such as:

- `<PyTorch>`

- `<Ignite>`

- `<Pandas>`

- `<OpenCV>`

- `<Memory Profiler>`

- `<NVIDIA Management Library>`

Note: You do not need to install Python packages you do not use.

Please refer [here](#) to find out how PipelineX differs from other pipeline/workflow packages: Airflow, Luigi, Gokart, Metaflow, and Kedro.



## INSTALL PIPELINEX

### 3.1 [Option 1] Install from the PyPI

```
pip install pipelinux
```

### 3.2 [Option 2] Development install

This is recommended only if you want to modify the source code of PipelineX.

```
git clone https://github.com/Minyus/pipelinux.git
cd pipelinux
python setup.py develop
```

### 3.3 Prepare development environment for PipelineX

You can install packages and organize development environment with [pipenv](#). Refer the [pipenv](#) document to install pipenv. Once you installed pipenv, you can use pipenv to install and organize your environment.

```
# install dependent libraries
$ pipenv install

# install development libraries
$ pipenv install --dev

# install pipelinux
$ pipenv run install

# install pipelinux via setup.py
$ pipenv run install_dev

# lint python code
$ pipenv run lint

# format python code
$ pipenv run fmt

# sort imports
$ pipenv run sort
```

(continues on next page)

(continued from previous page)

```
# apply mypy to python code
$ pipenv run vet

# get into shell
$ pipenv shell

# run test
$ pipenv run test
```

## 3.4 Prepare Docker environment for PipelineX

```
git clone https://github.com/Minyus/pipelineX.git
cd pipelineX
docker build --tag pipelineX .
docker run --rm -it pipelineX
```

## GETTING STARTED WITH PIPELINEX

### 4.1 Kedro 0.17.x Starter projects

Kedro starters (Cookiecutter templates) to use Kedro, Scikit-learn, MLflow, and PipelineX are available at: [kedro-starters-sklearn](#)

Iris dataset is included and used, but you can easily change to Kaggle Titanic dataset.

### 4.2 Example/Demo Projects tested with Kedro 0.16.x

- [Computer Vision using PyTorch](#)
  - `parameters.yml` at `conf/base/parameters.yml`
  - Essential packages: PyTorch, Ignite, Shap, Kedro, MLflow
  - Application: Image classification
  - Data: MNIST images
  - Model: CNN (Convolutional Neural Network)
  - Loss: Cross-entropy
- [Kaggle competition using PyTorch](#)
  - `parameters.yml` at `kaggle/conf/base/parameters.yml`
  - Essential packages: PyTorch, Ignite, pandas, numpy, Kedro, MLflow
  - Application: [Kaggle competition to predict the results of American Football plays](#)
  - Data: Sparse heatmap-like field images and tabular data
  - Model: Combination of CNN and MLP
  - Loss: Continuous Rank Probability Score (CRPS)
- [Computer Vision using OpenCV](#)
  - `parameters.yml` at `conf/base/parameters.yml`
  - Essential packages: OpenCV, Scikit-image, numpy, TensorFlow (pretrained model), Kedro, MLflow
  - Application: Image processing to estimate the empty area ratio of cuboid container on a truck
  - Data: container images
- [Uplift Modeling using CausalLift](#)

- `parameters.yml` at `conf/base/parameters.yml`
- Essential packages: CausalLift, Scikit-learn, XGBoost, pandas, Kedro
- Application: Uplift Modeling to find which customers should be targeted and which customers should not for a marketing campaign (treatment)
- Data: generated by simulation

## HATCHDICT: PYTHON IN YAML/JSON

API document

### 5.1 Python objects in YAML/JSON

#### 5.1.1 Introduction to YAML

YAML is a common text format used for application config files.

YAML's most notable advantage is allowing users to mix 2 styles, block style and flow style.

Example:

```
import yaml
from pprint import pprint  # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml="""
block_style_demo:
  key1: value1
  key2: value2
flow_style_demo: {key1: value1, key2: value2}
"""
parameters = yaml.safe_load(params_yaml)

print("### 2 styles in YAML ###")
pprint(parameters)
```

```
### 2 styles in YAML ###
{'block_style_demo': {'key1': 'value1', 'key2': 'value2'},
 'flow_style_demo': {'key1': 'value1', 'key2': 'value2'}}
```

To store highly nested (hierarchical) dict or list, YAML is more convenient than hard-coding in Python code.

- YAML's block style, which uses indentation, allows users to omit opening and closing symbols to specify a Python dict or list ({ } or [ ]).
- YAML's flow style, which uses opening and closing symbols, allows users to specify a Python dict or list within a single line.

So simply using YAML with Python will be the best way for Machine Learning experimentation?

Let's check out the next example.

Example:

```
import yaml
from pprint import pprint  # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml = """
model_kind: LogisticRegression
model_params:
  C: 1.23456
  max_iter: 987
  random_state: 42
"""
parameters = yaml.safe_load(params_yaml)

print("### Before ###")
pprint(parameters)

model_kind = parameters.get("model_kind")
model_params_dict = parameters.get("model_params")

if model_kind == "LogisticRegression":
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression(**model_params_dict)

elif model_kind == "DecisionTree":
    from sklearn.tree import DecisionTreeClassifier
    model = DecisionTreeClassifier(**model_params_dict)

elif model_kind == "RandomForest":
    from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier(**model_params_dict)

else:
    raise ValueError("Unsupported model_kind.")

print("\n### After ###")
print(model)
```

```
### Before ###
{'model_kind': 'LogisticRegression',
 'model_params': {'C': 1.23456, 'max_iter': 987, 'random_state': 42}}

### After ###
LogisticRegression(C=1.23456, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, l1_ratio=None, max_iter=987,
 multi_class='warn', n_jobs=None, penalty='l2',
 random_state=42, solver='warn', tol=0.0001, verbose=0,
 warm_start=False)
```

This way is inefficient as we need to add import and if statements for the options in the Python code in addition to modifying the YAML config file.

Any better way?



## 5.1.2 Python tags in YAML

PyYAML provides `UnsafeLoader` which can load Python objects without import.

Example usage of `!!python/object`

```
import yaml
# You do not need `import sklearn.linear_model` using PyYAML's UnsafeLoader

# Read parameters dict from a YAML file in actual use
params_yaml = """
model:
  !!python/object:sklearn.linear_model.LogisticRegression
  C: 1.23456
  max_iter: 987
  random_state: 42
"""

parameters = yaml.unsafe_load(params_yaml) # unsafe_load required

model = parameters.get("model")

print("### model object by PyYAML's UnsafeLoader ###")
print(model)
```

```
### model object by PyYAML's UnsafeLoader ###
LogisticRegression(C=1.23456, class_weight=None, dual=None, fit_intercept=None,
                   intercept_scaling=None, l1_ratio=None, max_iter=987,
                   multi_class=None, n_jobs=None, penalty=None, random_state=42,
                   solver=None, tol=None, verbose=None, warm_start=None)
```

Example usage of `!!python/name`

```
import yaml

# Read parameters dict from a YAML file in actual use
params_yaml = """
numpy_array_func:
  !!python/name:numpy.array
"""

try:
    parameters = yaml.unsafe_load(params_yaml) # unsafe_load required for PyYAML 5.1.
    ↪or later
except:
    parameters = yaml.load(params_yaml)

numpy_array_func = parameters.get("numpy_array_func")

import numpy

assert numpy_array_func == numpy.array
```

PyYAML's `!!python/object` and `!!python/name`, however, has the following problems.

- `!!python/object` or `!!python/name` are too long to write.
- Positional (non-keyword) arguments are apparently not supported.

Any better way?

PipelineX provides the solution.

### 5.1.3 Alternative to Python tags in YAML

PipelineX's HatchDict provides an easier syntax, as follows, to convert Python dictionaries read from YAML or JSON files to Python objects without `import`.

- Use `=` key to specify the package, module, and class/function with `.` separator in `foo_package.bar_module.baz_class` format.
- [Optional] Use `_` key to specify (list of) positional arguments (args) if any.
- [Optional] Add keyword arguments (kwargs) if any.

To return an object instance like PyYAML's `!!python/object`, feed positional and/or keyword arguments. If there is no arguments, just feed null (known as `None` in Python) to `_` key.

To return an uninstantiated (raw) object like PyYAML's `!!python/name`, just feed `=` key without positional nor keyword arguments.

Example alternative to `!!python/object`:

```
from pipelinex import HatchDict
import yaml
from pprint import pprint # pretty-print for clearer look
# You do not need `import sklearn.linear_model` using PipelineX's HatchDict

# Read parameters dict from a YAML file in actual use
params_yaml="""
model:
  =: sklearn.linear_model.LogisticRegression
  C: 1.23456
  max_iter: 987
  random_state: 42
"""
parameters = yaml.safe_load(params_yaml)

model_dict = parameters.get("model")

print("### Before ###")
pprint(model_dict)

model = HatchDict(parameters).get("model")

print("\n### After ###")
print(model)
```

```
### Before ###
{'=: 'sklearn.linear_model.LogisticRegression',
 'C': 1.23456,
 'max_iter': 987,
 'random_state': 42}

### After ###
LogisticRegression(C=1.23456, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=987,
```

(continues on next page)

(continued from previous page)

```
multi_class='warn', n_jobs=None, penalty='l2',
random_state=42, solver='warn', tol=0.0001, verbose=0,
warm_start=False)
```

Example alternative to `!!python/name:`

```
from pipelinex import HatchDict
import yaml

# Read parameters dict from a YAML file in actual use
params_yaml="""
numpy_array_func:
  =: numpy.array
"""
parameters = yaml.safe_load(params_yaml)

numpy_array_func = HatchDict(parameters).get("numpy_array_func")

import numpy

assert numpy_array_func == numpy.array
```

This import-less Python object supports nested objects (objects that receives object arguments) by recursive depth-first search.

For more examples, please see [Use with PyTorch](#).

This import-less Python object feature, inspired by the fact that Kedro uses `load_obj` for file I/O (DataSet), uses `load_obj` copied from [kedro.utils](#) which dynamically imports Python objects using `importlib`, a Python standard library.

## 5.2 Anchor-less aliasing in YAML/JSON

### 5.2.1 Aliasing in YAML

To avoid repeating, YAML natively provides Anchor&Alias [Anchor&Alias](#) feature, and [Jsonnet](#) provides [Variable](#) feature to JSON.

Example:

```
import yaml
from pprint import pprint # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml="""
train_params:
  train_batch_size: &batch_size 32
  val_batch_size: *batch_size
"""
parameters = yaml.safe_load(params_yaml)

train_params_dict = parameters.get("train_params")

print("### Conversion by YAML's Anchor&Alias feature ###")
pprint(train_params_dict)
```

```
### Conversion by YAML's Anchor&Alias feature ###
{'train_batch_size': 32, 'val_batch_size': 32}
```

Unfortunately, YAML and Jsonnet require a medium to share the same value.

This is why PipelineX provides anchor-less aliasing feature.

## 5.2.2 Alternative to aliasing in YAML

You can directly look up another value in the same YAML/JSON file using “\$” key without an anchor nor variable.

To specify the nested key (key in a dict of dict), use “.” as the separator.

Example:

```
from pipelineX import HatchDict
import yaml
from pprint import pprint  # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml="""
train_params:
  train_batch_size: 32
  val_batch_size: {'$': train_params.train_batch_size}
"""
parameters = yaml.safe_load(params_yaml)

train_params_dict = parameters.get("train_params")

print("### Before ###")
pprint(train_params_dict)

train_params = HatchDict(parameters).get("train_params")

print("\n### After ###")
pprint(train_params)
```

```
### Before ###
{'train_batch_size': 32,
 'val_batch_size': {'$': 'train_params.train_batch_size'}}

### After ###
{'train_batch_size': 32, 'val_batch_size': 32}
```

## 5.3 Python expression in YAML/JSON

Strings wrapped in parentheses are evaluated as a Python expression.

```
from pipelineX import HatchDict
import yaml
from pprint import pprint  # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml = """
```

(continues on next page)

(continued from previous page)

```

train_params:
  param1_tuple_python: (1, 2, 3)
  param1_tuple_yaml: !!python/tuple [1, 2, 3]
  param2_formula_python: (2 + 3)
  param3_neg_inf_python: (float("-Inf"))
  param3_neg_inf_yaml: -.Inf
  param4_float_1e9_python: (1e9)
  param4_float_1e9_yaml: 1.0e+09
  param5_int_1e9_python: (int(1e9))
"""
parameters = yaml.load(params_yaml)

train_params_raw = parameters.get("train_params")

print("### Before ###")
pprint(train_params_raw)

train_params_converted = HatchDict(parameters).get("train_params")

print("\n### After ###")
pprint(train_params_converted)

```

```

### Before ###
{'param1_tuple_python': '(1, 2, 3)',
 'param1_tuple_yaml': '(1, 2, 3)',
 'param2_formula_python': '(2 + 3)',
 'param3_neg_inf_python': '(float("-Inf"))',
 'param3_neg_inf_yaml': '-inf',
 'param4_float_1e9_python': '(1e9)',
 'param4_float_1e9_yaml': '1000000000.0',
 'param5_int_1e9_python': '(int(1e9))'}

### After ###
{'param1_tuple_python': (1, 2, 3),
 'param1_tuple_yaml': (1, 2, 3),
 'param2_formula_python': 5,
 'param3_neg_inf_python': -inf,
 'param3_neg_inf_yaml': -inf,
 'param4_float_1e9_python': 1000000000.0,
 'param4_float_1e9_yaml': 1000000000.0,
 'param5_int_1e9_python': 1000000000}

```



## INTRODUCTION TO KEDRO

### 6.1 Why the unified data interface framework is needed

Machine Learning projects involves with loading and saving various data in various ways such as:

- files in local/network file system, Hadoop Distributed File System (HDFS), Amazon S3, Google Cloud Storage
  - e.g. CSV, JSON, YAML, pickle, images, models, etc.
- databases
  - Postgresql, MySQL etc.
- Spark
- REST API (HTTP(S) requests)

It is often the case that many Machine Learning Engineers code both data loading/saving and data transformation mixed in the same Python module or Jupyter notebook during experimentation/prototyping phase and suffer later on because:

- During experimentation/prototyping, we often want to save the intermediate data after each transformation.
- In production environments, we often want to skip saving data to minimize latency and storage space.
- To benchmark the performance or troubleshoot, we often want to switch the data source.
  - e.g. read image files in local storage or download images through REST API

The proposed solution is the unified data interface.

Here is a simple demo example to predict survival on the [Titanic](#).

Common code to define the tasks/operations/transformations:

```
# Define tasks

def train_model(model, df, cols_features, col_target):
    # train a model here
    return model

def run_inference(model, df, cols_features):
    # run inference here
    return df
```

It is notable that you do *not* need to add any Kedro-related code here to use Kedro later on.

Furthermore, you do *not* need to add any MLflow-related code here to use MLflow later on as Kedro hooks provided by PipelineX can handle behind the scenes.

This advantage enables you to keep your pipelines for experimentation/prototyping/benchmarking production-ready.

1. Plain code:

```
# Configure: can be written in a config file (YAML, JSON, etc.)

train_data_filepath = "data/input/train.csv"
train_data_load_args = {"float_precision": "high"}

test_data_filepath = "data/input/test.csv"
test_data_load_args = {"float_precision": "high"}

pred_data_filepath = "data/load/pred.csv"
pred_data_save_args = {"index": False, "float_format": "%.16e"}

model_kind = "LogisticRegression"
model_params_dict = {
    "C": 1.23456
    "max_iter": 987
    "random_state": 42
}

# Run tasks

import pandas as pd

if model_kind == "LogisticRegression":
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression(**model_params_dict)

train_df = pd.read_csv(train_data_filepath, **train_data_load_args)
model = train_model(model, train_df)

test_df = pd.read_csv(test_data_filepath, **test_data_load_args)
pred_df = run_inference(model, test_df)
pred_df.to_csv(pred_data_filepath, **pred_data_save_args)
```

1. Following the data interface framework, objects with `_load`, and `_save` methods, proposed by [Kedro](#) and supported by PipelineX:

```
# Define a data interface: better ones such as "CSVDataSet" are provided by Kedro

import pandas as pd
from pathlib import Path

class CSVDataSet:
    def __init__(self, filepath, load_args={}, save_args={}):
        self._filepath = filepath
        self._load_args = {}
        self._load_args.update(load_args)
        self._save_args = {"index": False}
        self._save_args.update(save_args)

    def _load(self) -> pd.DataFrame:
        return pd.read_csv(self._filepath, **self._load_args)
```

(continues on next page)



(continued from previous page)

```

def _save(self, data: pd.DataFrame) -> None:
    save_path = Path(self._filepath)
    save_path.parent.mkdir(parents=True, exist_ok=True)
    data.to_csv(str(save_path), **self._save_args)

# Configure data interface: can be written in catalog config file using Kedro

train_dataset = CSVDataSet(
    filepath="data/input/train.csv",
    load_args={"float_precision": "high"},
    # save_args={"float_format": "%.16e"}, # You can set save_args for future use
)

test_dataset = CSVDataSet(
    filepath="data/input/test.csv",
    load_args={"float_precision": "high"},
    # save_args={"float_format": "%.16e"}, # You can set save_args for future use
)

pred_dataset = CSVDataSet(
    filepath="data/load/pred.csv",
    # load_args={"float_precision": "high"}, # You can set load_args for future use
    save_args={"float_format": "%.16e"},
)

model_kind = "LogisticRegression"
model_params_dict = {
    "C": 1.23456
    "max_iter": 987
    "random_state": 42
}
cols_features = [
    "Pclass", # The passenger's ticket class
    "Parch", # # of parents / children aboard the Titanic
]
col_target = "Survived" # Column used as the target: whether the passenger survived,
                        ↪ or not

# Run tasks: can be configured as a pipeline using Kedro
# and can be written in parameters config file using PipelineX

if model_kind == "LogisticRegression":
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression(**model_params_dict)

train_df = train_dataset._load()
model = train_model(model, train_df, cols_features, col_target)

test_df = test_dataset._load()
pred_df = run_inference(model, test_df, cols_features)

pred_dataset._save(pred_df)

```

Just following the data interface framework might be somewhat beneficial in the long run, but not enough.

Let's see what Kedro and PipelineX can do.

## 6.2 Kedro overview

Kedro is a Python package to develop pipelines consisting of:

- data interface sets (data loading/saving wrappers, called “DataSets”, that follows the unified data interface framework) such as:
  - `pandas.CSVDataSet`: a CSV file in local or cloud (Amazon S3, Google Cloud Storage) utilizing `filesystem_spec (fsspec)`
  - `pickle.PickleDataSet`: a pickle file in local or cloud (Amazon S3, Google Cloud Storage) utilizing `filesystem_spec (fsspec)`
  - `pandas.SQLTableDataSet`: a table data in an SQL database supported by `SQLAlchemy`
  - data interface sets for Spark, Google BigQuery, Feather, HDF, Parquet, Matplotlib, NetworkX, Excel, and more provided by Kedro
  - Custom data interface sets provided by Kedro users
- tasks/operations/transformations (called “Nodes”) provided by Kedro users such as:
  - data pre-processing
  - training a model
  - inference using a model
- inter-task dependency provided by Kedro users

Kedro pipelines can be run sequentially or in parallel.

Regarding Kedro, please see:

- [<Kedro’s document>](#)
- [<YouTube playlist: Writing Data Pipelines with Kedro>](#)
- [<Python Packages for Pipeline/Workflow>](#)

Here is a simple example Kedro project.

```
# catalog.yml

train_df:
  type: pandas.CSVDataSet # short for kedro.extras.datasets.pandas.CSVDataSet
  filepath: data/input/train.csv
  load_args:
    float_precision: high
  # save_args: # You can set save_args for future use
  # float_format": "%.16e"

test_df:
  type: pandas.CSVDataSet # short for kedro.extras.datasets.pandas.CSVDataSet
  filepath: data/input/test.csv
  load_args:
    float_precision: high
  # save_args: # You can set save_args for future use
  # float_format": "%.16e"

pred_df:
  type: pandas.CSVDataSet # short for kedro.extras.datasets.pandas.CSVDataSet
  filepath: data/load/pred.csv
```

(continues on next page)

(continued from previous page)

```
# load_args: # You can set load_args for future use
# float_precision: high
save_args:
    float_format: "%.16e"
```

```
# parameters.yml

model:
    !!python/object:sklearn.linear_model.LogisticRegression
    C: 1.23456
    max_iter: 987
    random_state: 42
cols_features: # Columns used as features in the Titanic data table
    - Pclass # The passenger's ticket class
    - Parch # # of parents / children aboard the Titanic
col_target: Survived # Column used as the target: whether the passenger survived or_
↪not
```

```
# pipeline.py

from kedro.pipeline import Pipeline, node

from my_module import train_model, run_inference

def create_pipeline(**kwargs):
    return Pipeline(
        [
            node(
                func=train_model,
                inputs=["params:model", "train_df", "params:cols_features",
↪"params:col_target"],
                outputs="model",
            ),
            node(
                func=run_inference,
                inputs=["model", "test_df", "params:cols_features"],
                outputs="pred_df",
            ),
        ]
    )
```

```
# run.py

from kedro.runner import SequentialRunner

# Set up ProjectContext here

context = ProjectContext()
context.run(pipeline_name="__default__", runner=SequentialRunner())
```

Kedro pipelines can be visualized using `kedro-viz`.

Kedro pipelines can be productionized using:

- `kedro-airflow`: converts a Kedro pipeline into Airflow Python operators.
- `kedro-docker`: builds a Docker image that can run a Kedro pipeline

- `kedro-argo`: converts a Kedro pipeline into an Argo (backend of Kubeflow) pipeline

## **FLEX-KEDRO: KEDRO PLUGIN FOR FLEXIBLE CONFIG**

[API document](#)

Flex-Kedro provides more options to configure Kedro projects flexibly and thus quickly by KFlex-Kedro-Pipeline and Flex-Kedro-Context features.

### **7.1 Flex-Kedro-Pipeline: Kedro plugin for quicker pipeline set up**

If you want to define Kedro pipelines quickly, you can consider to use `pipelinex.FlexiblePipeline` instead of `kedro.pipeline.Pipeline`. `pipelinex.FlexiblePipeline` adds the following options to `kedro.pipeline.Pipeline`.

#### **7.1.1 Dict for nodes**

To define each node, dict can be used instead of `kedro.pipeline.node`.

Example:

```
pipelinex.FlexiblePipeline(  
    nodes=[dict(func=task_func1, inputs="my_input", outputs="my_output")]  
)
```

will be equivalent to:

```
kedro.pipeline.Pipeline(  
    nodes=[  
        kedro.pipeline.node(func=task_func1, inputs="my_input", outputs="my_output")  
    ]  
)
```

#### **7.1.2 Sequential nodes**

For sub-pipelines consisting of nodes of only single input and single output, you can optionally use Sequential API similar to PyTorch (`torch.nn.Sequential`) and Keras (`tf.keras.Sequential`)

Example:

```
pipelinex.FlexiblePipeline(  
    nodes=[  
        dict(  

```

(continues on next page)

(continued from previous page)

```

        func=[task_func1, task_func2, task_func3],
        inputs="my_input",
        outputs="my_output",
    )
]
)

```

will be equivalent to:

```

kedro.pipeline.Pipeline(
    nodes=[
        kedro.pipeline.node(
            func=task_func1, inputs="my_input", outputs="my_output__001"
        ),
        kedro.pipeline.node(
            func=task_func2, inputs="my_output__001", outputs="my_output__002"
        ),
        kedro.pipeline.node(
            func=task_func3, inputs="my_output__002", outputs="my_output"
        ),
    ]
)

```

### 7.1.3 Decorators without using the method

- Optionally specify the Python function decorator(s) to apply to multiple nodes under the pipeline using `decorator` argument instead of using `decorate` method of `kedro.pipeline.Pipeline`.

Example:

```

pipeline.FlexiblePipeline(
    nodes=[
        kedro.pipeline.node(func=task_func1, inputs="my_input", outputs="my_output
        ↪")
    ],
    decorator=[task_deco, task_deco],
)

```

will be equivalent to:

```

kedro.pipeline.Pipeline(
    nodes=[
        kedro.pipeline.node(func=task_func1, inputs="my_input", outputs="my_output
        ↪")
    ]
).decorate(task_deco, task_deco)

```

- Optionally specify the default python module (path of .py file) if you do not want to repeat the same (deep and/or long) Python module (e.g. `foo.bar.my_task1`, `foo.bar.my_task2`, etc.)

## 7.2 Flex-Kedro-Context: Kedro plugin for YAML lovers

If you want to take advantage of YAML more than Kedro supports, you can consider to use `pipelinex.FlexibleContext` instead of `kedro.framework.context.KedroContext`. `pipelinex.FlexibleContext` adds preprocess of `parameters.yml` and `catalog.yml` to `kedro.framework.context.KedroContext` to provide flexibility. This option is for YAML lovers only. If you don't like YAML very much, skip this one.

### 7.2.1 Define Kedro pipelines in `parameters.yml`

You can define the inter-task dependency (DAG) for Kedro pipelines in `parameters.yml` using `PIPELINES` key. To define each Kedro pipeline, you can use the `kedro.pipeline.Pipeline` or its variant such as `pipelinex.FlexiblePipeline` as shown below.

```
# parameters.yml

PIPELINES:
  __default__:
    =: pipelinex.FlexiblePipeline
    module: # Optionally specify the default Python module so you can omit the module_
    ↪ name to which functions belongs
    decorator: # Optionally specify function decorator(s) to apply to each node
    nodes:
      - inputs: ["params:model", train_df, "params:cols_features", "params:col_target
    ↪ "]
        func: sklearn_demo.train_model
        outputs: model

      - inputs: [model, test_df, "params:cols_features"]
        func: sklearn_demo.run_inference
        outputs: pred_df
```

### 7.2.2 Configure Kedro run config in `parameters.yml`

You can specify the run config in `parameters.yml` using `RUN_CONFIG` key instead of specifying the args for `kedro run` command for every run.

You can still set the args for `kedro run` to overwrite.

In addition to the args for `kedro run`, you can opt to run only missing nodes (skip tasks which have already been run to resume pipeline using the intermediate data files or databases.) by `only_missing` key.

```
# parameters.yml

RUN_CONFIG:
  pipeline_name: __default__
  runner: SequentialRunner # Set to "ParallelRunner" to run in parallel
  only_missing: False # Set True to run only missing nodes
  tags: # None
  node_names: # None
  from_nodes: # None
  to_nodes: # None
  from_inputs: # None
  load_versions: # None
```

### 7.2.3 Use HatchDict feature in parameters.yml

You can use HatchDict feature in parameters.yml.

```
# parameters.yml

model:
  =: sklearn.linear_model.LogisticRegression
  C: 1.23456
  max_iter: 987
  random_state: 42
cols_features: # Columns used as features in the Titanic data table
  - Pclass # The passenger's ticket class
  - Parch # # of parents / children aboard the Titanic
col_target: Survived # Column used as the target: whether the passenger survived or_
↪not
```

### 7.2.4 Enable caching for Kedro DataSets in catalog.yml

Enable caching using `cached` key set to `True` if you do not want Kedro to load the data from disk/database which were in the memory. (`kedro.io.CachedDataSet` is used under the hood.)

### 7.2.5 Use HatchDict feature in catalog.yml

You can use HatchDict feature in catalog.yml.



## MLFLOW-ON-KEDRO: KEDRO PLUGIN FOR MLFLOW USERS

[API document](#)

### 8.1 How to use MLflow from Kedro projects

Kedro DataSet and Hooks (callbacks) are provided to use MLflow without adding any MLflow-related code in the node (task) functions.

- `pipelinex.MLflowDataSet`

Kedro Dataset that saves data to or loads data from MLflow depending on `dataset` argument as follows.

- If set to “p”, the value will be saved/loaded as an MLflow parameter (string).
- If set to “m”, the value will be saved/loaded as an MLflow metric (numeric).
- If set to “a”, the value will be saved/loaded based on the data type.
  - \* If the data type is either {float, int}, the value will be saved/loaded as an MLflow metric.
  - \* If the data type is either {str, list, tuple, set}, the value will be saved/load as an MLflow parameter.
  - \* If the data type is dict, the value will be flattened with dot (“.”) as the separator and then saved/loaded as either an MLflow metric or parameter based on each data type as explained above.
- If set to either {“json”, “csv”, “xls”, “parquet”, “png”, “jpg”, “jpeg”, “img”, “pkl”, “txt”, “yaml”, “yml”}, the backend dataset instance will be created accordingly to save/load as an MLflow artifact.
- If set to a Kedro DataSet object or a dictionary, it will be used as the backend dataset to save/load as an MLflow artifact.
- If set to None (default), MLflow logging will be skipped.

Regarding all the options, please see the [API document](#)

- Kedro Hooks
  - `pipelinex.MLflowBasicLoggerHook`: Configures MLflow logging and logs duration time for the pipeline to MLflow.
  - `pipelinex.MLflowArtifactsLoggerHook`: Logs artifacts of specified file paths and dataset names to MLflow.
  - `pipelinex.MLflowDataSetsLoggerHook`: Logs datasets of (list of) float/int and str classes to MLflow.
  - `pipelinex.MLflowTimeLoggerHook`: Logs duration time for each node (task) to MLflow and optionally visualizes the execution logs as a Gantt chart by `plotly.figure_factory.create_gantt` if `plotly` is installed.

- `pipelineX.AddTransformersHook`: Adds Kedro transformers such as:
  - \* `pipelineX.MLflowIOTimeLoggerTransformer`: Logs duration time to load and save each dataset with args:

Regarding all the options, please see the [API document](#)

MLflow-ready Kedro projects can be generated by the [Kedro starters](#) (Cookiecutter template) which include the following example config:

```
# catalog.yml

# Write a pickle file & upload to MLflow
model:
  type: pipelineX.MLflowDataSet
  dataset: pkl

# Write a csv file & upload to MLflow
pred_df:
  type: pipelineX.MLflowDataSet
  dataset: csv

# Write an MLflow metric
score:
  type: pipelineX.MLflowDataSet
  dataset: m
```

```
# catalog.py (alternative to catalog.yml)

catalog_dict = {
    "model": MLflowDataSet(dataset="pkl"), # Write a pickle file & upload to MLflow
    "pred_df": MLflowDataSet(dataset="csv"), # Write a csv file & upload to MLflow
    "score": MLflowDataSet(dataset="m"), # Write an MLflow metric
}
```

```
# mlflow_config.py

import pipelineX

mlflow_hooks = (
    pipelineX.MLflowBasicLoggerHook(
        uri="sqlite:///mlruns/sqlite.db",
        experiment_name="experiment_001",
        artifact_location="./mlruns/experiment_001",
        offset_hours=0,
    ),
    pipelineX.MLflowCatalogLoggerHook(
        auto=True,
    ),
    pipelineX.MLflowArtifactsLoggerHook(
        filepaths_before_pipeline_run=["conf/base/parameters.yml"],
        filepaths_after_pipeline_run=[
            "logs/info.log",
            "logs/errors.log",
        ],
    ),
    pipelineX.MLflowEnvVarsLoggerHook(
        param_env_vars=["HOSTNAME"],
    ),
)
```

(continues on next page)

(continued from previous page)

```

        metric_env_vars=[],
    ),
    pipelinex.MLflowTimeLoggerHook(),
    pipelinex.AddTransformersHook(
        transformers=[pipelinex.MLflowIOTimeLoggerTransformer()],
    ),
)

```

## 8.2 Comparison with kedro-mlflow package

Both PipelineX's MLflow-on-Kedro and [kedro-mlflow](#) provide integration of MLflow to Kedro. Here are the comparisons.

- Features supported by both PipelineX and kedro-mlflow
  - Kedro DataSets and Hooks to log (save/upload) artifacts, parameters, and metrics to MLflow.
  - Truncate MLflow parameter values to 250 characters to avoid error due to MLflow parameter length limit.
  - Dict values can be flattened using dot (".") as the separator to log each value inside the dict separately.
- Features supported by only PipelineX
  - [Time logging] Option to log execution time for each task (Kedro node) as MLflow metrics
  - [Gantt logging] Option to log Gantt chart HTML file that visualizes execution time using Plotly as an MLflow artifact (inspired by [Apache Airflow](#))
  - [Automatic backend Kedro DataSets for common artifacts] Option to specify a common file extension ({"json", "csv", "xls", "parquet", "png", "jpg", "jpeg", "img", "pkl", "txt", "yaml", "yml"}) so the Kedro DataSet object will be created behind the scene instead of manually specifying a Kedro DataSet including filepath in the catalog (inspired by [Kedro Wings](#)).
  - [Automatic logging for MLflow parameters and metrics] Option to log each dataset not listed in the catalog as MLflow parameter or metric, instead of manually specifying a Kedro DataSet in the catalog.
    - \* If the data type is either {float, int}, the value will be saved/loaded as an MLflow metric.
    - \* If the data type is either {str, list, tuple, set}, the value will be saved/load as an MLflow parameter.
    - \* If the data type is dict, the value will be flattened with dot (".") as the separator and then saved/loaded as either an MLflow metric or parameter based on each data type as explained above.
    - \* For example, "data\_loading\_config": {"train": {"batch\_size": 32}} will be logged as MLflow metric of "data\_loading\_config.train.batch\_size": 32
  - [Flexible config per DataSet] For each Kedro DataSet, it is possible to configure differently. For example, a dict value can be logged as an MLflow parameter (string) as is while another one can be logged as an MLflow metric after being flattened.
  - [Direct artifact logging] Option to specify the paths of any data to log as MLflow artifacts after Kedro pipeline runs without using a Kedro DataSet, which is useful if you want to save local files (e.g. info/warning/error log files, intermediate model weights saved by Machine Learning packages such as PyTorch and TensorFlow, etc.)
  - [Environment Variable logging] Option to log Environment Variables
  - [Downloading] Option to download MLflow artifacts, params, metrics from an existing MLflow experiment run using the Kedro DataSet

- [Up to date] Support for Kedro 0.17.x (released in Dec 2020) or later
- Features provided by only kedro-mlflow
  - A wrapper for MLflow's `log_model`
  - Configure MLflow logging in a YAML file
  - Option to use MLflow tag or raise error if MLflow parameter values exceed 250 characters

---

## KEDRO-EXTRAS: KEDRO PLUGIN TO USE VARIOUS PYTHON PACKAGES

---

[API document](#)

Kedro-Extras provides Kedro DataSets and decorators not available in [kedro.extras](#).

Contributors who are willing to help preparing the test code and send pull request to Kedro following Kedro's [CONTRIBUTING.md](#) are welcomed.

### 9.1 Additional Kedro datasets (data interface sets)

[pipelinex.extras.datasets](#) provides the following Kedro Datasets (data interface sets) mainly for Computer Vision applications using PyTorch/torchvision, OpenCV, and Scikit-image.

- [pipelinex.ImagesLocalDataSet](#)
  - loads/saves multiple numpy arrays (RGB, BGR, or monochrome image) from/to a folder in local storage using [pillow](#) package, working like `kedro.extras.datasets.pillow.ImageDataSet` and `kedro.io.PartitionedDataSet` with conversion between numpy arrays and Pillow images.
  - an example project is at [pipelinex\\_image\\_processing](#)
- [pipelinex.APIDataset](#)
  - modified version of [kedro.extras.APIDataset](#) with more flexible options including downloading multiple contents (such as images and json) by HTTP requests to multiple URLs using [requests](#) package
  - an example project is at [pipelinex\\_image\\_processing](#)
- [pipelinex.AsyncAPIDataset](#)
  - downloads multiple contents (such as images and json) by asynchronous HTTP requests to multiple URLs using [httpx](#) package
  - an example project is at [pipelinex\\_image\\_processing](#)
- [pipelinex.IterableImagesDataSet](#)
  - wrapper of `torchvision.datasets.ImageFolder` that loads images in a folder as an iterable data loader to use with PyTorch.
- [pipelinex.PandasProfilingDataSet](#)
  - generates a pandas dataframe summary report using [pandas-profiling](#)
- more data interface sets for pandas dataframe summarization/visualization provided by PipelineX

## 9.2 Additional function decorators for benchmarking

`pipelinex.extras.decorators` provides Python decorators for benchmarking.

- `log_time`
  - logs the duration time of a function (difference of timestamp before and after running the function).
  - Slightly modified version of Kedro's `log_time`
- `mem_profile`
  - logs the peak memory usage during running the function.
  - `memory_profiler` needs to be installed.
  - Slightly modified version of Kedro's `mem_profile`
- `nvml_profile`
  - logs the difference of NVIDIA GPU usage before and after running the function.
  - `pynvml` or `py3nvml` needs to be installed.

```
from pipelinex import log_time
from pipelinex import mem_profile # Need to install memory_profiler for memory_
↪profiling
from pipelinex import nvml_profile # Need to install pynvml for NVIDIA GPU profiling
from time import sleep
import logging

logging.basicConfig(level=logging.INFO)

@nvml_profile
@mem_profile
@log_time
def foo_func(i=1):
    sleep(0.5) # Needed to avoid the bug reported at https://github.com/
↪pythonprofilers/memory_profiler/issues/216
    return "a" * i

output = foo_func(100_000_000)
```

```
INFO:pipelinex.decorators.decorators:Running 'foo_func' took 549ms [0.549s]
INFO:pipelinex.decorators.memory_profiler:Running 'foo_func' consumed 579.02MiB_
↪memory at peak time
INFO:pipelinex.decorators.nvml_profiler:Ran: 'foo_func', NVML returned: {'_Driver_
↪Version': '418.67', '_NVML_Version': '10.418.67', 'Device_Count': 1, 'Devices': [{'_
↪Name': 'Tesla P100-PCIE-16GB', 'Total_Memory': 17071734784, 'Free_Memory':_
↪17071669248, 'Used_Memory': 65536, 'GPU_Utilization_Rate': 0, 'Memory_Utilization_
↪Rate': 0}]], Used memory diff: [0]
```

## 9.3 Use with PyTorch

To develop a simple neural network, it is convenient to use Sequential API (e.g. `torch.nn.Sequential`, `tf.keras.Sequential`).

- Hardcoded:

```
from torch.nn import Sequential, Conv2d, ReLU

model = Sequential(
    Conv2d(in_channels=3, out_channels=16, kernel_size=[3, 3]),
    ReLU(),
)

print("### model object by hard-coding ###")
print(model)
```

```
### model object by hard-coding ###
Sequential(
  (0): Conv2d(3, 16, kernel_size=[3, 3], stride=(1, 1))
  (1): ReLU()
)
```

- Using import-less Python object feature:

```
from pipelinex import HatchDict
import yaml
from pprint import pprint  # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml="""
model:
  =: torch.nn.Sequential
  _:
    - {=: torch.nn.Conv2d, in_channels: 3, out_channels: 16, kernel_size: [3, 3]}
    - {=: torch.nn.ReLU, _: }
"""
parameters = yaml.safe_load(params_yaml)

model_dict = parameters.get("model")

print("### Before ###")
pprint(model_dict)

model = HatchDict(parameters).get("model")

print("\n### After ###")
print(model)
```

```
### Before ###
{'=: 'torch.nn.Sequential',
 '_': [{'=': 'torch.nn.Conv2d',
        'in_channels': 3,
        'kernel_size': [3, 3],
        'out_channels': 16},
```

(continues on next page)

(continued from previous page)

```

        {'=': 'torch.nn.ReLU', '_': None}}})

### After ###
Sequential(
  (0): Conv2d(3, 16, kernel_size=[3, 3], stride=(1, 1))
  (1): ReLU()
)

```

In addition to Sequential, TensorFlow/Keras provides modules to merge branches such as `tf.keras.layers.Concatenate`, but PyTorch provides only functional interface such as `torch.cat`.

PipelineX provides modules to merge branches such as `ModuleConcat`, `ModuleSum`, and `ModuleAvg`.

- Hardcoded:

```

from torch.nn import Sequential, Conv2d, AvgPool2d, ReLU
from pipelineX import ModuleConcat

model = Sequential(
    ModuleConcat(
        Conv2d(in_channels=3, out_channels=16, kernel_size=[3, 3], stride=[2, 2],
padding=[1, 1]),
        AvgPool2d(kernel_size=[3, 3], stride=[2, 2], padding=[1, 1]),
    ),
    ReLU(),
)
print("### model object by hard-coding ###")
print(model)

```

```

### model object by hard-coding ###
Sequential(
  (0): ModuleConcat(
    (0): Conv2d(3, 16, kernel_size=[3, 3], stride=[2, 2], padding=[1, 1])
    (1): AvgPool2d(kernel_size=[3, 3], stride=[2, 2], padding=[1, 1])
  )
  (1): ReLU()
)

```

- Using import-less Python object feature:

```

from pipelineX import HatchDict
import yaml
from pprint import pprint # pretty-print for clearer look

# Read parameters dict from a YAML file in actual use
params_yaml="""
model:
  =: torch.nn.Sequential
  -:
    - =: pipelineX.ModuleConcat
      -:
        - {=: torch.nn.Conv2d, in_channels: 3, out_channels: 16, kernel_size: [3, 3],
stride: [2, 2], padding: [1, 1]}
        - {=: torch.nn.AvgPool2d, kernel_size: [3, 3], stride: [2, 2], padding: [1,
1]}
        - {=: torch.nn.ReLU, _: }
"""

```

(continues on next page)



(continued from previous page)

```

parameters = yaml.safe_load(params_yaml)

model_dict = parameters.get("model")

print("### Before ###")
pprint(model_dict)

model = HatchDict(parameters).get("model")

print("\n### After ###")
print(model)

```

```

### Before ###
{'=': 'torch.nn.Sequential',
 '_': [{'=': 'pipelinex.ModuleConcat',
          '_': [{'=': 'torch.nn.Conv2d',
                    'in_channels': 3,
                    'kernel_size': [3, 3],
                    'out_channels': 16,
                    'padding': [1, 1],
                    'stride': [2, 2]},
                  {'=': 'torch.nn.AvgPool2d',
                    'kernel_size': [3, 3],
                    'padding': [1, 1],
                    'stride': [2, 2]}]},
        {'=': 'torch.nn.ReLU', '_': None}]}

### After ###
Sequential(
  (0): ModuleConcat(
    (0): Conv2d(3, 16, kernel_size=[3, 3], stride=[2, 2], padding=[1, 1])
    (1): AvgPool2d(kernel_size=[3, 3], stride=[2, 2], padding=[1, 1])
  )
  (1): ReLU()
)

```

## 9.4 Use with PyTorch Ignite

Wrappers of PyTorch Ignite provides most of features available in Ignite, including integration with MLflow, in an easy declarative way.

In addition, the following optional features are available in PipelineX.

- Use only partial samples in dataset (Useful for quick preliminary check before using the whole dataset)
- Time limit for training (Useful for code-only (Kernel-only) Kaggle competitions with time limit)

Here are the arguments for `NetworkTrain`:

```

loss_fn (callable): Loss function used to train.
    Accepts an instance of loss functions at https://pytorch.org/docs/stable/nn.html
    ↪ #loss-functions
epochs (int, optional): Max epochs to train
seed (int, optional): Random seed for training.
optimizer (torch.optim, optional): Optimizer used to train.

```

(continues on next page)

(continued from previous page)

```

    Accepts optimizers at https://pytorch.org/docs/stable/optim.html
optimizer_params (dict, optional): Parameters for optimizer.
train_data_loader_params (dict, optional): Parameters for data loader for training.
    Accepts args at https://pytorch.org/docs/stable/data.html#torch.utils.data.
    ↪ DataLoader
val_data_loader_params (dict, optional): Parameters for data loader for validation.
    Accepts args at https://pytorch.org/docs/stable/data.html#torch.utils.data.
    ↪ DataLoader
evaluation_metrics (dict, optional): Metrics to compute for evaluation.
    Accepts dict of metrics at https://pytorch.org/ignite/metrics.html
evaluate_train_data (str, optional): When to compute evaluation_metrics using
    ↪ training dataset.
    Accepts events at https://pytorch.org/ignite/engine.html#ignite.engine.Events
evaluate_val_data (str, optional): When to compute evaluation_metrics using
    ↪ validation dataset.
    Accepts events at https://pytorch.org/ignite/engine.html#ignite.engine.Events
progress_update (bool, optional): Whether to show progress bar using tqdm package
scheduler (ignite.contrib.handle.param_scheduler.ParamScheduler, optional): Param
    ↪ scheduler.
    Accepts a ParamScheduler at
    https://pytorch.org/ignite/contrib/handlers.html#module-ignite.contrib.handlers.
    ↪ param_scheduler
scheduler_params (dict, optional): Parameters for scheduler
model_checkpoint (ignite.handlers.ModelCheckpoint, optional): Model Checkpoint.
    Accepts a ModelCheckpoint at https://pytorch.org/ignite/handlers.html#ignite.handlers.ModelCheckpoint
    ↪ handlers.ModelCheckpoint
model_checkpoint_params (dict, optional): Parameters for ModelCheckpoint at
    https://pytorch.org/ignite/handlers.html#ignite.handlers.ModelCheckpoint
early_stopping_params (dict, optional): Parameters for EarlyStopping at
    https://pytorch.org/ignite/handlers.html#ignite.handlers.EarlyStopping
time_limit (int, optional): Time limit for training in seconds.
train_dataset_size_limit (int, optional): If specified, only the subset of training
    ↪ dataset is used.
    Useful for quick preliminary check before using the whole dataset.
val_dataset_size_limit (int, optional): If specified, only the subset of validation
    ↪ dataset is used.
    useful for quick preliminary check before using the whole dataset.
cudnn_deterministic (bool, optional): Value for torch.backends.cudnn.deterministic.
    See https://pytorch.org/docs/stable/notes/randomness.html for details.
cudnn_benchmark (bool, optional): Value for torch.backends.cudnn.benchmark.
    See https://pytorch.org/docs/stable/notes/randomness.html for details.
mlflow_logging (bool, optional): If True and MLflow is installed, MLflow logging is
    ↪ enabled.

```

Please see the [example code using MNIST dataset](#) prepared based on the [original code](#).

It is also possible to use:

- [FlexibleModelCheckpoint](#) handler which enables to use timestamp in the model checkpoint file name to clarify which one is the latest.
- [CohenKappaScore](#) metric which can compute Quadratic Weighted Kappa Metric used in some Kaggle competitions. See [sklearn.metrics.cohen\\_kappa\\_score](#) for details.

It is planned to port some [code used with PyTorch Ignite](#) to [PyTorch Ignite](#) repository once test and example codes are prepared.

## 9.5 Use with OpenCV

A challenge of image processing is that the parameters and algorithms that work with an image often do not work with another image. You will want to output intermediate images from each image processing pipeline step for visual check during development, but you will not want to output all the intermediate images to save time and disk space in production.

Wrappers of OpenCV and `ImagesLocalDataSet` are the solution. You can concentrate on developping your image processing pipeline for an image (3-D or 2-D numpy array), and it will run for all the images in a folder.

If you are devepping an image processing pipeline consisting of 5 steps and you have 10 images, for example, you can check 10 generated images in each of 5 folders, 50 images in total, during development.



## STORY BEHIND PIPELINEX

When I was working on a Deep Learning project, it was very time-consuming to develop the pipeline for experimentation. I wanted 2 features.

First one was an option to resume the pipeline using the intermediate data files instead of running the whole pipeline. This was important for rapid Machine/Deep Learning experimentation.

Second one was modularity, which means keeping the 3 components, task processing, file/database access, and DAG definition, independent. This was important for efficient software engineering.

After this project, I explored for a long-term solution. I researched about 3 Python packages for pipeline development, Airflow, Luigi, and Kedro, but none of these could be a solution.

Luigi provided resuming feature, but did not offer modularity. Kedro offered modularity, but did not provide resuming feature.

After this research, I decided to develop my own package that works on top of Kedro. Besides, I added syntactic sugars including Sequential API similar to Keras and PyTorch to define DAG. Furthermore, I added integration with MLflow, PyTorch, Ignite, pandas, OpenCV, etc. while working on more Machine/Deep Learning projects.

After I confirmed my package worked well with the Kaggle competition, I released it as PipelineX.



---

CHAPTER  
**ELEVEN**

---

**AUTHOR**

Yusuke Minami @Minyus

- <Linkedin>
- <Twitter>





## CONTRIBUTORS ARE WELCOME!

### 12.1 How to contribute

Please see [CONTRIBUTING.md](#) for details.

### 12.2 Contributor list

- <[@shibuiwilliam](#)>
- <[@MarchRaBBiT](#)>





PIPELINEX

13.1 pipelinux package

13.1.1 Subpackages

13.1.1.1 pipelinux.extras package

13.1.1.1.1 Subpackages

13.1.1.1.1.1 pipelinux.extras.datasets package

13.1.1.1.1.2 Subpackages

13.1.1.1.1.3 pipelinux.extras.datasets.httpx package

13.1.1.1.1.4 Submodules

13.1.1.1.1.5 pipelinux.extras.datasets.httpx.async\_api\_dataset module

```
class pipelinux.extras.datasets.httpx.async_api_dataset.AsyncAPIDataset(url=None,
                                method='GET',
                                data=None,
                                params=None,
                                headers=None,
                                auth=None,
                                timeout=60,
                                attributes="",
                                skip_errors=False,
                                transforms=[],
                                session_config={},
                                pool_config={'http://':
                                              {'max_retries':
                                               0,
                                               'pool_block':
                                                False,
                                               'pool_connections':
                                                10,
                                               'pool_maxsize':
                                                10},
                                              'https://':
```

```
pipelinex.extras.datasets.httpx.async_api_dataset.asyncio_run(aw)
```

```
async pipelinex.extras.datasets.httpx.async_api_dataset.request_coroutine(session,
                                                                           method,
                                                                           url,
                                                                           re-
                                                                           quest_args)
```

```
async pipelinex.extras.datasets.httpx.async_api_dataset.requests_coroutine(session_config,
                                                                           method,
                                                                           url_list,
                                                                           re-
                                                                           quest_args)
```

#### 13.1.1.1.1.6 pipelinex.extras.datasets.opencv package

#### 13.1.1.1.1.7 Submodules

#### 13.1.1.1.1.8 pipelinex.extras.datasets.opencv.images\_dataset module

```
class pipelinex.extras.datasets.opencv.images_dataset.OpenCVImagesLocalDataSet(filepath,
                                                                           load_args=None,
                                                                           save_args=None,
                                                                           chan-
                                                                           nel_first=False,
                                                                           ver-
                                                                           sion=None)
```

Bases: kedro.io.core.AbstractVersionedDataSet

**\_\_init\_\_**(filepath, load\_args=None, save\_args=None, channel\_first=False, version=None)  
Creates a new instance of AbstractVersionedDataSet.

##### Parameters

- **filepath** (str) – Filepath in POSIX format to a file.
- **version** (Optional[Version]) – If specified, should be an instance of kedro.io.core.Version. If its load attribute is None, the latest version will be loaded. If its save attribute is None, save version will be autogenerated.
- **exists\_function** – Function that is used for determining whether a path exists in a filesystem.
- **glob\_function** – Function that is used for finding all paths in a filesystem, which match a given pattern.

#### 13.1.1.1.1.9 pipelinex.extras.datasets.pandas package

#### 13.1.1.1.1.10 Submodules

#### 13.1.1.1.1.11 pipelinex.extras.datasets.pandas.csv\_local module

CSVLocalDataSet loads and saves data to a local csv file. The underlying functionality is supported by pandas, so it supports all allowed pandas options for loading and saving csv files.

```
class pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet (filepath,  
                                                                    load_args=None,  
                                                                    save_args=None,  
                                                                    ver-  
                                                                    sion=None)
```

Bases: `kedro.io.core.AbstractVersionedDataSet`

`CSVLocalDataSet` loads and saves data to a local csv file. The underlying functionality is supported by `pandas`, so it supports all allowed `pandas` options for loading and saving csv files.

Example:

```
from kedro.io import CSVLocalDataSet  
import pandas as pd  
  
data = pd.DataFrame({'col1': [1, 2], 'col2': [4, 5],  
                    'col3': [5, 6]})  
data_set = CSVLocalDataSet(filepath="test.csv",  
                            load_args=None,  
                            save_args={"index": False})  
  
data_set.save(data)  
reloaded = data_set.load()  
  
assert data.equals(reloaded)
```

**DEFAULT\_LOAD\_ARGS:** `Dict[str, Any] = {}`

**DEFAULT\_SAVE\_ARGS:** `Dict[str, Any] = {'index': False}`

**\_\_init\_\_** (*filepath*, *load\_args*=None, *save\_args*=None, *version*=None)

Creates a new instance of `CSVLocalDataSet` pointing to a concrete filepath.

#### Parameters

- **filepath** (`str`) – path to a csv file.
- **load\_args** (`Optional[Dict[str, Any]]`) – Pandas options for loading csv files. Here you can find all available arguments: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) All defaults are preserved.
- **save\_args** (`Optional[Dict[str, Any]]`) – Pandas options for saving csv files. Here you can find all available arguments: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html) All defaults are preserved, but “index”, which is set to `False`.
- **version** (`Optional[Version]`) – If specified, should be an instance of `kedro.io.core.Version`. If its `load` attribute is `None`, the latest version will be loaded. If its `save` attribute is `None`, save version will be autogenerated.

**Raises** `ValueError` – If ‘filepath’ looks like a remote path.

### 13.1.1.1.12 `pipelinex.extras.datasets.pandas.efficient_csv_local` module

```
class pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSVLocalDataSet (*args,
                                                                                   pre-
                                                                                   view_args,
                                                                                   mar-
                                                                                   gin=100.0,
                                                                                   ver-
                                                                                   bose=True,
                                                                                   **kwargs)
```

Bases: `pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet`

**DEFAULT\_LOAD\_ARGS:** `Dict[str, Any] = {'engine': 'c', 'keep_default_na': False, 'na_val': None}`

**DEFAULT\_PREVIEW\_ARGS:** `Dict[str, Any] = {'low_memory': False, 'nrows': None}`

**\_\_init\_\_** (\*args, preview\_args=None, margin=100.0, verbose=True, \*\*kwargs)  
Creates a new instance of `PandasDescribeDataSet` pointing to a concrete filepath.

#### Parameters

- **args** – Positional arguments for `CSVLocalDataSet`
- **preview\_args** (Optional[`Dict[str, Any]`]) – Arguments passed on to `df.describe`. See <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html> for details.
- **kwargs** – Keyword arguments for `CSVLocalDataSet`

```
pipelinex.extras.datasets.pandas.efficient_csv_local.dict_string_val_prefix (d,
                                                                                   pre-
                                                                                   fix)
```

```
pipelinex.extras.datasets.pandas.efficient_csv_local.dict_val_replace_except (d,
                                                                                   to_except,
                                                                                   new_value)
```

### 13.1.1.1.13 `pipelinex.extras.datasets.pandas.fixed_width_csv_dataset` module

```
class pipelinex.extras.datasets.pandas.fixed_width_csv_dataset.FixedWidthCSVDataSet (*args,
                                                                                   en-
                                                                                   able_fixed_width=True,
                                                                                   num_decimal_places=9,
                                                                                   **kwargs)
```

Bases: `kedro.extras.datasets.pandas.csv_dataset.CSVDataSet`

`CSVDataSet` loads/saves data from/to a CSV file using an underlying filesystem (e.g.: local, S3, GCS). It uses `pandas` to handle the CSV file.

**\_\_init\_\_** (\*args, enable\_fixed\_width=True, num\_decimal\_places=9, \*\*kwargs)  
Creates a `FixedWidthCSVDataSet` pointing to a concrete CSV file on a specific filesystem. :param filepath: Filepath in POSIX format to a CSV file prefixed with a protocol like `s3://`.

If prefix is not provided, `file` protocol (local filesystem) will be used. The prefix should be any protocol supported by `fsspec`. Note: `http(s)` doesn't support versioning.

#### Parameters

- **load\_args** – Pandas options for loading CSV files. Here you can find all available arguments: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html) All defaults are preserved.
- **save\_args** – Pandas options for saving CSV files. Here you can find all available arguments: [https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to\\_csv.html](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html) All defaults are preserved, but “index”, which is set to False.
- **version** – If specified, should be an instance of `kedro.io.core.Version`. If its `load` attribute is `None`, the latest version will be loaded. If its `save` attribute is `None`, save version will be autogenerated.
- **credentials** – Credentials required to get access to the underlying filesystem. E.g. for `GCSFileSystem` it should look like `{“token”: None}`.
- **fs\_args** – Extra arguments to pass into underlying filesystem class constructor (e.g. `{“project”: “my-project”}` for `GCSFileSystem`), as well as to pass to the filesystem’s `open` method through nested keys `open_args_load` and `open_args_save`. Here you can find all available arguments for `open`: <https://filesystem-spec.readthedocs.io/en/latest/api.html#fspec.spec.AbstractFileSystem.open> All defaults are preserved, except `mode`, which is set to `r` when loading and to `w` when saving.
- **enable\_fixed\_width** (`bool`) – Save to a CSV file with each column width fixed among all the rows and the header to improve readability for humans.
- **num\_decimal\_places** (`int`) – Number of decimal places for float values to save.

```
pipelinex.extras.datasets.pandas.fixed_width_csv_dataset.fix_width(df,  
                                                                    num_decimal_places=9)
```

#### 13.1.1.1.14 pipelinex.extras.datasets.pandas.histogram module

```
class pipelinex.extras.datasets.pandas.histogram.HistogramDataSet(filepath,  
                                                                    save_args=None,  
                                                                    hist_args=None)  
  
Bases: kedro.io.core.AbstractDataSet  
  
__init__(filepath, save_args=None, hist_args=None)  
    Initialize self. See help(type(self)) for accurate signature.
```

#### 13.1.1.1.15 pipelinex.extras.datasets.pandas.pandas\_cat\_matrix module

```
class pipelinex.extras.datasets.pandas.pandas_cat_matrix.PandasCatMatrixDataSet(*args,  
                                                                                de-  
                                                                                scribe_args={},  
                                                                                **kwargs)  
  
Bases: pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet  
  
PandasDescribeDataSet saves output of df.describe.  
  
__init__(*args, describe_args={}, **kwargs)  
    Creates a new instance of PandasCatMatrixDataSet pointing to a concrete filepath.  
  
Parameters  
    • args – Positional arguments for CSVLocalDataSet
```



- **describe\_args** (Dict[str, Any]) – Arguments passed on to `df.describe`. See <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html> for details.
- **kwargs** – Keyword arguments for `CSVLocalDataSet`

#### 13.1.1.1.16 `pipelinex.extras.datasets.pandas.pandas_describe` module

```
class pipelinex.extras.datasets.pandas.pandas_describe.PandasDescribeDataSet (*args,
                                                                              de-
                                                                              scribe_args={},
                                                                              **kwargs)
```

Bases: `pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet`

`PandasDescribeDataSet` saves output of `df.describe`.

```
__init__ (*args, describe_args={}, **kwargs)
```

Creates a new instance of `PandasDescribeDataSet` pointing to a concrete filepath.

##### Parameters

- **args** – Positional arguments for `CSVLocalDataSet`
- **describe\_args** (Dict[str, Any]) – Arguments passed on to `df.describe`. See <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html> for details.
- **kwargs** – Keyword arguments for `CSVLocalDataSet`

#### 13.1.1.1.17 `pipelinex.extras.datasets.pandas_profiling` package

#### 13.1.1.1.18 Submodules

#### 13.1.1.1.19 `pipelinex.extras.datasets.pandas_profiling.pandas_profiling` module

```
class pipelinex.extras.datasets.pandas_profiling.pandas_profiling.PandasProfilingDataSet (fil
```

Bases: `kedro.io.core.AbstractVersionedDataSet`

`PandasProfilingDataSet` is an `AbstractVersionedDataSet` to generate pandas profiling report. See <https://github.com/pandas-profiling/pandas-profiling> for details.

```
DEFAULT_SAVE_ARGS: Dict[str, Any] = {}
```

```
__init__ (filepath, save_args=None, sample_args=None, version=None)
```

Creates a new instance of `PandasProfilingDataSet` pointing to a concrete filepath.

##### Parameters

- **filepath** (str) – path to a local yaml file.
- **save\_args** (Optional[Dict[str, Any]]) – Arguments passed on to `df.profile_report` such as title. See <https://pandas-profiling.github.io/pandas-profiling/>

docs/ for details. See [https://github.com/pandas-profiling/pandas-profiling/blob/master/pandas\\_profiling/config\\_default.yaml](https://github.com/pandas-profiling/pandas-profiling/blob/master/pandas_profiling/config_default.yaml) for default values.

- **sample\_args** (Optional[Dict[str, Any]]) – Arguments passed on to df.sample. See <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html> for details.
- **version** (Optional[Version]) – If specified, should be an instance of `kedro.io.core.Version`. If its `load` attribute is `None`, the latest version will be loaded. If its `save` attribute is `None`, save version will be autogenerated.

#### 13.1.1.1.1.20 `pipelineX.extras.datasets.pillow` package

##### 13.1.1.1.1.21 Submodules

##### 13.1.1.1.1.22 `pipelineX.extras.datasets.pillow.images_dataset` module

```
class pipelineX.extras.datasets.pillow.images_dataset.ImagesLocalDataSet(path,  
                                load_args=None,  
                                save_args={'suffix':  
                                'jpg'},  
                                channel_first=False,  
                                reverse_color=False,  
                                version=None)
```

Bases: `kedro.io.core.AbstractVersionedDataSet`

Loads/saves a dict of numpy 3-D or 2-D arrays from/to a folder containing images.

Works like `kedro.extras.datasets.pillow.ImageDataSet` and `kedro.io.PartitionedDataSet` with conversion between numpy arrays and Pillow images.

```
__init__(path, load_args=None, save_args={'suffix': 'jpg'}, channel_first=False, reverse_color=False, version=None)
```

#### Parameters

- **path** (str) – The folder path containing images
- **load\_args** (Optional[Dict[str, Any]]) – Args fed to <https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.open>
- **save\_args** (Dict[str, Any]) – Args, e.g.
  - *suffix*: file suffix such as “jpg”
  - *upper*: optionally used as the upper pixel value corresponding to 0xFF (255) for linear scaling to ensure the pixel value is between 0 and 255.
  - *lower*: optionally used as the lower pixel value corresponding to 0x00 (0) for linear scaling to ensure the pixel value is between 0 and 255.
  - *mode*: fed to <https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.fromarray>
  - Any other args fed to <https://pillow.readthedocs.io/en/stable/reference/Image.html#PIL.Image.Image.save>

- **channel\_first** – If true, the first dimension of 3-D array is treated as channel (color) as in PyTorch. If false, the last dimension of the 3-D array is treated as channel (color) as in TensorFlow, Pillow, and OpenCV.
- **reverse\_color** – If true, the order of channel (color) is reversed (RGB to BGR when loading, BGR to RGB when saving). Set true to use packages such as OpenCV which uses BGR order natively.
- **version** (Optional[Version]) – If specified, should be an instance of `kedro.io.core.Version`. If its `load` attribute is `None`, the latest version will be loaded. If its `save` attribute is `None`, save version will be autogenerated.

```
class pipelinex.extras.datasets.pillow.images_dataset.Np3DArrDataset (a)
    Bases: object
```

```
    __init__ (a)
        Initialize self. See help(type(self)) for accurate signature.
```

```
class pipelinex.extras.datasets.pillow.images_dataset.Np3DArrDatasetFromList (a,
                                                                              transform=None)
    Bases: object
```

```
    __init__ (a, transform=None)
        Initialize self. See help(type(self)) for accurate signature.
```

```
pipelinex.extras.datasets.pillow.images_dataset.load_image (load_path, load_args,
                                                             as_numpy=False,
                                                             channel_first=False,
                                                             reverse_color=False)
```

```
pipelinex.extras.datasets.pillow.images_dataset.scale (**kwargs)
```

#### 13.1.1.1.1.23 pipelinex.extras.datasets.requests package

#### 13.1.1.1.1.24 Submodules

#### 13.1.1.1.1.25 pipelinex.extras.datasets.requests.api\_dataset module

`APIDataSet` loads the data from HTTP(S) APIs and returns them into either as string or json Dict. It uses the python requests library: <https://requests.readthedocs.io/en/master/>

```
class pipelinex.extras.datasets.requests.api_dataset.APIDataset (url=None,
                                                                method='GET',
                                                                data=None,
                                                                params=None,
                                                                headers=None,
                                                                auth=None,
                                                                timeout=60,
                                                                attribute="",
                                                                skip_errors=False,
                                                                transforms=[], session_config={},
                                                                pool_config={'http://':
                                                                {'max_retries':
                                                                0,
                                                                'pool_block':
                                                                False,
                                                                'pool_connections':
                                                                10,
                                                                'pool_maxsize':
                                                                10}, 'https://':
                                                                {'max_retries':
                                                                0,
                                                                'pool_block':
                                                                False,
                                                                'pool_connections':
                                                                10,
                                                                'pool_maxsize':
                                                                10}})
```

Bases: `kedro.io.core.AbstractDataSet`

APIDataset loads the data from HTTP(S) APIs. It uses the python requests library: <https://requests.readthedocs.io/en/master/>

Example:

```
from kedro.extras.datasets.api import APIDataset

data_set = APIDataset(
    url="https://quickstats.nass.usda.gov"
    params={
        "key": "SOME_TOKEN",
        "format": "JSON",
        "commodity_desc": "CORN",
        "statisticcat_desc": "YIELD",
        "agg_level_desc": "STATE",
        "year": 2000
    }
)
data = data_set.load()
```

```
__init__(url=None, method='GET', data=None, params=None, headers=None, auth=None,
         timeout=60, attribute="", skip_errors=False, transforms=[], session_config={},
         pool_config={'http://': {'max_retries': 0, 'pool_block': False, 'pool_connections': 10,
                                   'pool_maxsize': 10}, 'https://': {'max_retries': 0, 'pool_block': False, 'pool_connections':
                                   10, 'pool_maxsize': 10}})
```

Creates a new instance of APIDataSet to fetch data from an API endpoint.

#### Parameters

- **url** (Union[str, List[str], Dict[str, str], None]) – The API URL endpoint.
- **method** (str) – The Method of the request, GET, POST, PUT, DELETE, HEAD, etc...
- **data** (Optional[Any]) – The request payload, used for POST, PUT, etc requests <https://requests.readthedocs.io/en/master/user/quickstart/#more-complicated-post-requests>
- **params** (Optional[Dict[str, Any]]) – The url parameters of the API. <https://requests.readthedocs.io/en/master/user/quickstart/#passing-parameters-in-urls>
- **headers** (Optional[Dict[str, Any]]) – The HTTP headers. <https://requests.readthedocs.io/en/master/user/quickstart/#custom-headers>
- **auth** (Union[Tuple[str], AuthBase, None]) – Anything requests accepts. Normally it's either ('login', 'password'), or AuthBase, HTTPBasicAuth instance for more complex cases.
- **timeout** (int) – The wait time in seconds for a response, defaults to 1 minute. <https://requests.readthedocs.io/en/master/user/quickstart/#timeouts>
- **attribute** (str) – The attribute of response to return. Normally it's either *text*, which returns pure text, *json*, which returns JSON in Python Dict format, *content*, which returns a raw content, or `` (empty string), which returns the response object itself. Defaults to `` (empty string).
- **skip\_errors** (bool) – If True, exceptions will not interrupt loading data and be returned instead of the expected responses by `_load` method. Defaults to False.
- **transforms** (List[callable]) – List of callables to transform the output.
- **session\_config** (Dict[str, Any]) – Dict of arguments fed to the session.
- **pool\_config** (Dict[str, Dict[str, Any]]) – Dict of mounting prefix key to Dict of requests.adapters.HTTPAdapter param key to value. <https://requests.readthedocs.io/en/master/user/advanced/#transport-adapters> <https://urllib3.readthedocs.io/en/latest/advanced-usage.html>

### 13.1.1.1.1.26 `pipelinex.extras.datasets.seaborn` package

#### 13.1.1.1.1.27 Submodules

#### 13.1.1.1.1.28 `pipelinex.extras.datasets.seaborn.seaborn_pairplot` module

```
class pipelinex.extras.datasets.seaborn.seaborn_pairplot.SeabornPairPlotDataSet (filepath,  
                                                                    save_args=None,  
                                                                    sam-  
                                                                    ple_args=None,  
                                                                    ver-  
                                                                    sion=None)
```

Bases: `kedro.io.core.AbstractVersionedDataSet`

**DEFAULT\_SAVE\_ARGS:** `Dict[str, Any] = {}`

**\_\_init\_\_** (filepath, save\_args=None, sample\_args=None, version=None)

Creates a new instance of `AbstractVersionedDataSet`.

##### Parameters

- **filepath** (`str`) – Filepath in POSIX format to a file.
- **version** (`Optional[Version]`) – If specified, should be an instance of `kedro.io.core.Version`. If its `load` attribute is `None`, the latest version will be loaded. If its `save` attribute is `None`, save version will be autogenerated.
- **exists\_function** – Function that is used for determining whether a path exists in a filesystem.
- **glob\_function** – Function that is used for finding all paths in a filesystem, which match a given pattern.

### 13.1.1.1.1.29 `pipelinex.extras.datasets.torchvision` package

#### 13.1.1.1.1.30 Submodules

#### 13.1.1.1.1.31 `pipelinex.extras.datasets.torchvision.iterable_images_dataset` module

```
class pipelinex.extras.datasets.torchvision.iterable_images_dataset.IterableImagesDataSet (filepath,  
                                                                    load_args=None,  
                                                                    save_args=None,  
                                                                    version=None)
```

Bases: `kedro.io.core.AbstractVersionedDataSet`

Loads a folder containing images as an iterable. Wrapper of: <https://pytorch.org/docs/stable/torchvision/datasets.html#imagefolder>

**\_\_init\_\_** (filepath, load\_args=None, save\_args=None, version=None)

##### Parameters

- **filepath** (`str`) – *root* fed to: <https://pytorch.org/docs/stable/torchvision/datasets.html#imagefolder>
- **load\_args** (`Optional[Dict[str, Any]]`) – Args fed to: <https://pytorch.org/docs/stable/torchvision/datasets.html#imagefolder>

- **save\_args** (Optional[Dict[str, Any]]) – Ignored as saving is not supported.
- **version** (Optional[Version]) – If specified, should be an instance of `kedro.io.core.Version`. If its load attribute is None, the latest version will be loaded.

#### 13.1.1.1.1.32 Submodules

#### 13.1.1.1.1.33 `pipelineX.extras.datasets.core` module

#### 13.1.1.1.1.34 `pipelineX.extras.decorators` package

#### 13.1.1.1.1.35 Submodules

#### 13.1.1.1.1.36 `pipelineX.extras.decorators.decorators` module

`pipelineX.extras.decorators.decorators.log_time` (*func*)

A function decorator which logs the time taken for executing a function.

**Parameters** **func** (Callable) – The function to be logged.

**Return type** Callable

**Returns** A wrapped function, which will execute the provided function and log the running time.

#### 13.1.1.1.1.37 `pipelineX.extras.decorators.memory_profiler` module

`pipelineX.extras.decorators.memory_profiler.mem_profile` (*func*)

A function decorator which profiles the memory used when executing the function. The logged memory is collected by using the `memory_profiler` python module and includes memory used by children processes. The usage is collected by taking memory snapshots every 100ms. This decorator will only work with functions taking at least 0.5s to execute due to a bug in the `memory_profiler` python module. For more information about the bug, please see [https://github.com/pythonprofilers/memory\\_profiler/issues/216](https://github.com/pythonprofilers/memory_profiler/issues/216)

**Parameters** **func** (Callable) – The function to be profiled.

**Return type** Callable

**Returns** A wrapped function, which will execute the provided function and log its max memory usage upon completion.

#### 13.1.1.1.1.38 `pipelineX.extras.decorators.nvml_profiler` module

`pipelineX.extras.decorators.nvml_profiler.get_nv_info` ()

`pipelineX.extras.decorators.nvml_profiler.nvml_profile` (*func*)

**Return type** Callable

### 13.1.1.1.1.39 `pipelinex.extras.decorators.pandas_decorators` module

`pipelinex.extras.decorators.pandas_decorators.df_set_index(cols)`  
decorator with arguments

**Return type** Callable

`pipelinex.extras.decorators.pandas_decorators.log_df_summary(func)`

**Return type** Callable

`pipelinex.extras.decorators.pandas_decorators.total_seconds_to_datetime(cols, origin='1970-01-01')`  
decorator with arguments

**Return type** Callable

### 13.1.1.1.1.40 `pipelinex.extras.hooks` package

#### 13.1.1.1.1.41 Submodules

### 13.1.1.1.1.42 `pipelinex.extras.hooks.add_catalog_dict` module

**class** `pipelinex.extras.hooks.add_catalog_dict.AddCatalogDictHook(catalog_dict)`  
Bases: object

Hook to add data sets.

**\_\_init\_\_**(*catalog\_dict*)

**Parameters** *catalog\_dict* (Dict[str, AbstractDataSet]) – catalog\_dict to add.

**after\_catalog\_created**(*catalog*)

**Return type** None

### 13.1.1.1.1.43 `pipelinex.extras.hooks.add_transformers` module

**class** `pipelinex.extras.hooks.add_transformers.AddTransformersHook(transformers)`  
Bases: object

Hook to add transformers.

**\_\_init\_\_**(*transformers*)

**Parameters** *transformers* (Union[AbstractTransformer, List[AbstractTransformer], Tuple[AbstractTransformer]]) – transformers to add.

**after\_catalog\_created**(*catalog*)

**Return type** None



#### 13.1.1.1.1.44 pipelinex.extras.ops package

#### 13.1.1.1.1.45 Subpackages

#### 13.1.1.1.1.46 pipelinex.extras.ops.ignite package

#### 13.1.1.1.1.47 Subpackages

#### 13.1.1.1.1.48 pipelinex.extras.ops.ignite.declaratives package

#### 13.1.1.1.1.49 Submodules

#### 13.1.1.1.1.50 pipelinex.extras.ops.ignite.declaratives.declarative\_trainer module

```

class pipelinex.extras.ops.ignite.declaratives.declarative_trainer.NetworkTrain(loss_fn=None,
                                         epochs=None,
                                         seed=None,
                                         op-
                                         ti-
                                         mizer=None,
                                         op-
                                         ti-
                                         mizer_params=
                                         train_data_loade
                                         val_data_loade
                                         eval-
                                         u-
                                         a-
                                         tion_metrics=N
                                         eval-
                                         u-
                                         ate_train_data=
                                         eval-
                                         u-
                                         ate_val_data=N
                                         progress_updat
                                         sched-
                                         uler=None,
                                         sched-
                                         uler_params={
                                         model_checkpo
                                         model_checkpo
                                         early_stopping_
                                         time_limit=Non
                                         train_dataset_s
                                         val_dataset_siz
                                         cudnn_determin
                                         cudnn_benchmc
                                         mlflow_logging
                                         train_params=,

```

Bases: object

Create a trainer for a supervised PyTorch model.

#### Parameters

- **loss\_fn** (*callable*) – Loss function used to train. Accepts an instance of loss functions at <https://pytorch.org/docs/stable/nn.html#loss-functions>
- **epochs** (*int, optional*) – Max epochs to train
- **seed** (*int, optional*) – Random seed for training.
- **optimizer** (*torch.optim, optional*) – Optimizer used to train. Accepts optimizers at <https://pytorch.org/docs/stable/optim.html>
- **optimizer\_params** (*dict, optional*) – Parameters for optimizer.
- **train\_data\_loader\_params** (*dict, optional*) – Parameters for data loader for training. Accepts args at <https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader>
- **val\_data\_loader\_params** (*dict, optional*) – Parameters for data loader for validation. Accepts args at <https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader>
- **evaluation\_metrics** (*dict, optional*) – Metrics to compute for evaluation. Accepts dict of metrics at <https://pytorch.org/ignite/metrics.html>
- **evaluate\_train\_data** (*str, optional*) – When to compute evaluation\_metrics using training dataset. Accepts events at <https://pytorch.org/ignite/engine.html#ignite.engine.Events>
- **evaluate\_val\_data** (*str, optional*) – When to compute evaluation\_metrics using validation dataset. Accepts events at <https://pytorch.org/ignite/engine.html#ignite.engine.Events>
- **progress\_update** (*bool, optional*) – Whether to show progress bar using tqdm package
- **scheduler** (*ignite.contrib.handlers.param\_scheduler.ParamScheduler, optional*) – Param scheduler. Accepts a ParamScheduler at [https://pytorch.org/ignite/contrib/handlers.html#module-ignite.contrib.handlers.param\\_scheduler](https://pytorch.org/ignite/contrib/handlers.html#module-ignite.contrib.handlers.param_scheduler)
- **scheduler\_params** (*dict, optional*) – Parameters for scheduler
- **model\_checkpoint** (*ignite.handlers.ModelCheckpoint, optional*) – Model Checkpoint. Accepts a ModelCheckpoint at <https://pytorch.org/ignite/handlers.html#ignite.handlers.ModelCheckpoint>
- **model\_checkpoint\_params** (*dict, optional*) – Parameters for ModelCheckpoint at <https://pytorch.org/ignite/handlers.html#ignite.handlers.ModelCheckpoint>
- **early\_stopping\_params** (*dict, optional*) – Parameters for EarlyStopping at <https://pytorch.org/ignite/handlers.html#ignite.handlers.EarlyStopping>
- **time\_limit** (*int, optional*) – Time limit for training in seconds.
- **train\_dataset\_size\_limit** (*int, optional*) – If specified, only the subset of training dataset is used. Useful for quick preliminary check before using the whole dataset.

- **val\_dataset\_size\_limit** (*int, optional*) – If specified, only the subset of validation dataset is used. useful for quick preliminary check before using the whole dataset.
- **cudnn\_deterministic** (*bool, optional*) – Value for `torch.backends.cudnn.deterministic`. See <https://pytorch.org/docs/stable/notes/randomness.html> for details.
- **cudnn\_benchmark** (*bool, optional*) – Value for `torch.backends.cudnn.benchmark`. See <https://pytorch.org/docs/stable/notes/randomness.html> for details.
- **mlflow\_logging** (*bool, optional*) – If True and MLflow is installed, MLflow logging is enabled.

**Returns** a callable to train a PyTorch model.

**Return type** `trainer` (callable)

```
__init__(loss_fn=None, epochs=None, seed=None, optimizer=None, optimizer_params={},
         train_data_loader_params={}, val_data_loader_params={}, evaluation_metrics=None,
         evaluate_train_data=None, evaluate_val_data=None, progress_update=None, scheduler=None,
         scheduler_params={}, model_checkpoint=None, model_checkpoint_params={},
         early_stopping_params={}, time_limit=None, train_dataset_size_limit=None,
         val_dataset_size_limit=None, cudnn_deterministic=None, cudnn_benchmark=None,
         mlflow_logging=True, train_params={})
```

Initialize self. See `help(type(self))` for accurate signature.

#### 13.1.1.1.1.51 `pipelinex.extras.ops.ignite.handlers` package

#### 13.1.1.1.1.52 Submodules

#### 13.1.1.1.1.53 `pipelinex.extras.ops.ignite.handlers.flexible_checkpoint` module

```
class pipelinex.extras.ops.ignite.handlers.flexible_checkpoint.FlexibleModelCheckpoint (dirname, filename_prefix,
file-
name
offset_hours, filename_format=None, suffix_format=None,
off-
set_h
file-
name
suffix_format=None,
fix_for
*args, **kwargs)
```

Bases: `pipelinex.extras.ops.ignite.handlers.flexible_checkpoint.ModelCheckpoint`

```
__init__(dirname, filename_prefix, offset_hours=0, filename_format=None, suffix_format=None,
         *args, **kwargs)
```

Initialize self. See `help(type(self))` for accurate signature.

#### 13.1.1.1.154 `pipelinex.extras.ops.ignite.handlers.time_limit` module

**class** `pipelinex.extras.ops.ignite.handlers.time_limit.TimeLimit` (*limit\_sec=3600*)

Bases: `object`

**\_\_init\_\_** (*limit\_sec=3600*)

Time limit for training.

**Parameters** **limit\_sec** (*int, optional*) – Time limit in seconds. Defaults to 3600.

#### 13.1.1.1.155 `pipelinex.extras.ops.ignite.metrics` package

#### 13.1.1.1.156 Submodules

#### 13.1.1.1.157 `pipelinex.extras.ops.ignite.metrics.cohen_kappa_score` module

**class** `pipelinex.extras.ops.ignite.metrics.cohen_kappa_score.CohenKappaScore` (*\*args, \*\*kwargs*)

Bases: `ignite.metrics.metric.Metric`

Calculates the cohen kappa score. - *update* must receive output of the form (*y\_pred*, *y*) or {'y\_pred': *y\_pred*, 'y': *y*}.

**\_\_init\_\_** (*\*args, \*\*kwargs*)

Initialize self. See `help(type(self))` for accurate signature.

**compute** ()

Computes the metric based on it's accumulated state.

By default, this is called at the end of each epoch.

**Returns**

the actual quantity of interest. However, if a `Mapping` is returned, it will be (shallow) flattened into `engine.state.metrics` when `completed()` is called.

**Return type** Any

**Raises** **NotComputableError** – raised when the metric cannot be computed.

**reset** ()

Resets the metric to it's initial state.

By default, this is called at the start of each epoch.

**Return type** None

**update** (*output*)

Updates the metric's state using the passed batch output.

By default, this is called once for each batch.

**Parameters** **output** (`Sequence[Tensor]`) – the is the output from the engine's process function.

**Return type** None

### 13.1.1.1.1.58 `pipelinex.extras.ops.ignite.metrics.fbeta_score` module

```
class pipelinex.extras.ops.ignite.metrics.fbeta_score.FbetaScore(beta=1, out-  
put_transform=<function FbetaS-  
core.<lambda>>, aver-  
age='macro', is_multilabel=False,  
de-  
vice=None)
```

Bases: `ignite.metrics.metric.Metric`

```
__init__(beta=1, output_transform=<function FbetaScore.<lambda>>, average='macro',  
is_multilabel=False, device=None)  
Initialize self. See help(type(self)) for accurate signature.
```

```
compute()  
Computes the metric based on it's accumulated state.  
By default, this is called at the end of each epoch.
```

#### Returns

the actual quantity of interest. However, if a Mapping is returned, it will be (shallow) flattened into `engine.state.metrics` when `completed()` is called.

**Return type** Any

**Raises** `NotComputableError` – raised when the metric cannot be computed.

```
reset()  
Resets the metric to it's initial state.  
By default, this is called at the start of each epoch.
```

**Return type** None

```
update(output)  
Updates the metric's state using the passed batch output.  
By default, this is called once for each batch.
```

**Parameters** `output` (`Sequence[Tensor]`) – the is the output from the engine's process function.

**Return type** None

#### 13.1.1.1.1.59 `pipelinex.extras.ops.ignite.metrics.utils` module

```
class pipelinex.extras.ops.ignite.metrics.utils.ClassificationOutputTransform (num_classes=None)
    Bases: object
    __init__ (num_classes=None)
        Initialize self. See help(type(self)) for accurate signature.
```

#### 13.1.1.1.1.60 Submodules

#### 13.1.1.1.1.61 `pipelinex.extras.ops.allennlp_ops` module

```
class pipelinex.extras.ops.allennlp_ops.AllennlpReaderToDict (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.
```

#### 13.1.1.1.1.62 `pipelinex.extras.ops argparse_ops` module

```
class pipelinex.extras.ops.argparse_ops.FeedArgsDict (func, args={}, force_return=None)
    Bases: object
    __init__ (func, args={}, force_return=None)
        Initialize self. See help(type(self)) for accurate signature.
pipelinex.extras.ops.argparse_ops.namespace (d)
```

#### 13.1.1.1.1.63 `pipelinex.extras.ops.numpy_ops` module

```
class pipelinex.extras.ops.numpy_ops.ReverseChannel (channel_first=False)
    Bases: object
    __init__ (channel_first=False)
        Initialize self. See help(type(self)) for accurate signature.
pipelinex.extras.ops.numpy_ops.reverse_channel (a, channel_first=False)
pipelinex.extras.ops.numpy_ops.to_channel_first_arr (a)
pipelinex.extras.ops.numpy_ops.to_channel_last_arr (a)
```

#### 13.1.1.1.1.64 `pipelinex.extras.ops.opencv_ops` module

```
class pipelinex.extras.ops.opencv_ops.CvBGR2Gray (*args, **kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict
    __init__ (*args, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.
    fn = 'cvtColor'
class pipelinex.extras.ops.opencv_ops.CvBGR2HSV (*args, **kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict
```

```

    __init__ (*args, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    fn = 'cvtColor'

class pipelinex.extras.ops.opencv_ops.CvBilateralFilter (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'bilateralFilter'

class pipelinex.extras.ops.opencv_ops.CvBlur (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'blur'

class pipelinex.extras.ops.opencv_ops.CvBoxFilter (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'boxFilter'

class pipelinex.extras.ops.opencv_ops.CvCanny (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'Canny'

class pipelinex.extras.ops.opencv_ops.CvCvtColor (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'cvtColor'

class pipelinex.extras.ops.opencv_ops.CvDiagonalEdgeFilter2d (kernel_type=2,
                                                                **kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvModuleL2

    __init__ (kernel_type=2, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.opencv_ops.CvDictToDict (**kwargs)
    Bases: pipelinex.utils.DictToDict

    module = <module 'cv2' from '/home/docs/checkouts/readthedocs.org/user_builds/pipeline

class pipelinex.extras.ops.opencv_ops.CvDilate (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'dilate'

class pipelinex.extras.ops.opencv_ops.CvEqualizeHist (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'equalizeHist'

class pipelinex.extras.ops.opencv_ops.CvErode (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'erode'

class pipelinex.extras.ops.opencv_ops.CvFilter2d (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'filter2D'

class pipelinex.extras.ops.opencv_ops.CvGaussianBlur (**kwargs)
    Bases: pipelinex.extras.ops.opencv_ops.CvDictToDict

    fn = 'GaussianBlur'

```

```
class pipelindex.extras.ops.opencv_ops.CvHoughLinesP (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    fn = 'HoughLinesP'

class pipelindex.extras.ops.opencv_ops.CvLine (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    fn = 'line'

class pipelindex.extras.ops.opencv_ops.CvMedianBlur (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    fn = 'medianBlur'

class pipelindex.extras.ops.opencv_ops.CvModuleConcat (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleListMerge

class pipelindex.extras.ops.opencv_ops.CvModuleL1 (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleStack

class pipelindex.extras.ops.opencv_ops.CvModuleL2 (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleStack

class pipelindex.extras.ops.opencv_ops.CvModuleListMerge (*modules)
    Bases: object

    __init__ (*modules)
        Initialize self. See help(type(self)) for accurate signature.

class pipelindex.extras.ops.opencv_ops.CvModuleMean (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleStack

class pipelindex.extras.ops.opencv_ops.CvModuleStack (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleListMerge

class pipelindex.extras.ops.opencv_ops.CvModuleSum (*modules)
    Bases: pipelindex.extras.ops.opencv_ops.CvModuleStack

class pipelindex.extras.ops.opencv_ops.CvResize (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    fn = 'resize'

class pipelindex.extras.ops.opencv_ops.CvScale (width, height)
    Bases: object

    __init__ (width, height)
        Initialize self. See help(type(self)) for accurate signature.

class pipelindex.extras.ops.opencv_ops.CvSobel (ddepth='CV_64F', **kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    __init__ (ddepth='CV_64F', **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    fn = 'Sobel'

class pipelindex.extras.ops.opencv_ops.CvThreshold (type='THRESH_BINARY',
                                                    **kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.CvDictToDict

    __init__ (type='THRESH_BINARY', **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    fn = 'threshold'
```



```

class pipelindex.extras.ops.opencv_ops.NpAbs (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'abs'

class pipelindex.extras.ops.opencv_ops.NpConcat (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'concatenate'

class pipelindex.extras.ops.opencv_ops.NpDictToDict (**kwargs)
    Bases: pipelindex.utils.DictToDict

    module = <module 'numpy' from '/home/docs/checkouts/readthedocs.org/user_builds/pipeli...

class pipelindex.extras.ops.opencv_ops.NpFullLike (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'full_like'

class pipelindex.extras.ops.opencv_ops.NpMean (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'mean'

class pipelindex.extras.ops.opencv_ops.NpOnesLike (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'ones_like'

class pipelindex.extras.ops.opencv_ops.NpSqrt (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'sqrt'

class pipelindex.extras.ops.opencv_ops.NpSquare (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'square'

class pipelindex.extras.ops.opencv_ops.NpStack (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'stack'

class pipelindex.extras.ops.opencv_ops.NpSum (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'sum'

class pipelindex.extras.ops.opencv_ops.NpZerosLike (**kwargs)
    Bases: pipelindex.extras.ops.opencv_ops.NpDictToDict

    fn = 'zeros_like'

pipelindex.extras.ops.opencv_ops.expand_repeat (a, repeats=1, axis=None)

pipelindex.extras.ops.opencv_ops.fit_to_1 (a)

pipelindex.extras.ops.opencv_ops.fit_to_uint8 (a)

pipelindex.extras.ops.opencv_ops.mix_up (*imgs)

pipelindex.extras.ops.opencv_ops.overlay (*imgs)

pipelindex.extras.ops.opencv_ops.sum_up (*imgs)

```

### 13.1.1.1.1.65 `pipelinex.extras.ops.pandas_ops` module

```
class pipelinex.extras.ops.pandas_ops.DfAddRowStat (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfAgg (groupby=None,          columns=None,
                                             keep_others=False,      method=None,
                                             **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask
    method = 'agg'

class pipelinex.extras.ops.pandas_ops.DfAggregate (groupby=None,  columns=None,
                                                    keep_others=False, method=None,
                                                    **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask
    method = 'aggregate'

class pipelinex.extras.ops.pandas_ops.DfApply (groupby=None,      columns=None,
                                                keep_others=False,   method=None,
                                                **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask
    method = 'apply'

class pipelinex.extras.ops.pandas_ops.DfApplymap (groupby=None,   columns=None,
                                                    keep_others=False,   method=None,
                                                    **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask
    method = 'applymap'

class pipelinex.extras.ops.pandas_ops.DfAssignColumns (names=None,
                                                         name_fmt='{:03d}')
    Bases: object
    __init__ (names=None, name_fmt='{:03d}')
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfBaseMethod (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.
    method = None

class pipelinex.extras.ops.pandas_ops.DfBaseTask (groupby=None,   columns=None,
                                                    keep_others=False,   method=None,
                                                    **kwargs)
    Bases: object
    __init__ (groupby=None, columns=None, keep_others=False, method=None, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.
    method = None

class pipelinex.extras.ops.pandas_ops.DfColApply (func, **kwargs)
    Bases: object
```

```

    __init__(func, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfConcat (new_col_name=None,
                                                  new_col_values=None, col_id=None,
                                                  sort=False)

    Bases: object

    __init__(new_col_name=None, new_col_values=None, col_id=None, sort=False)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfCondReplace (flag, columns, value=nan, re-
                                                         place_if_flag=True, **kwargs)

    Bases: object

    __init__(flag, columns, value=nan, replace_if_flag=True, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfDrop (**kwargs)
    Bases: pipelinetx.extras.ops.pandas_ops.DfBaseMethod

    method = 'drop'

class pipelinetx.extras.ops.pandas_ops.DfDropDuplicates (**kwargs)
    Bases: pipelinetx.extras.ops.pandas_ops.DfBaseMethod

    method = 'drop_duplicates'

class pipelinetx.extras.ops.pandas_ops.DfDropFilter (**kwargs)
    Bases: object

    __init__(**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfDtypesApply (func, **kwargs)
    Bases: object

    __init__(func, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfDuplicate (columns)
    Bases: object

    __init__(columns)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfEval (expr, parser='pandas', engine=None, true-
                                                div=True)

    Bases: object

    __init__(expr, parser='pandas', engine=None, truediv=True)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinetx.extras.ops.pandas_ops.DfEwm (groupby=None, columns=None,
                                                keep_others=False, method=None,
                                                **kwargs)
    Bases: pipelinetx.extras.ops.pandas_ops.DfBaseTask

    method = 'ewm'

class pipelinetx.extras.ops.pandas_ops.DfExpanding (groupby=None, columns=None,
                                                      keep_others=False, method=None,
                                                      **kwargs)
    Bases: pipelinetx.extras.ops.pandas_ops.DfBaseTask

```

```
    method = 'expanding'

class pipelineX.extras.ops.pandas_ops.DfFillna (groupby=None,          columns=None,
                                                keep_others=False,      method=None,
                                                **kwargs)
    Bases: pipelineX.extras.ops.pandas_ops.DfBaseTask
    method = 'fillna'

class pipelineX.extras.ops.pandas_ops.DfFilter (groupby=None,          columns=None,
                                                keep_others=False,      method=None,
                                                **kwargs)
    Bases: pipelineX.extras.ops.pandas_ops.DfBaseTask

class pipelineX.extras.ops.pandas_ops.DfFilterCols (groupby=None,     columns=None,
                                                keep_others=False,
                                                method=None, **kwargs)
    Bases: pipelineX.extras.ops.pandas_ops.DfFilter

class pipelineX.extras.ops.pandas_ops.DfFocusTransform (focus,          columns,
                                                         groupby=None,
                                                         keep_others=False,
                                                         func='max', **kwargs)
    Bases: object
    __init__ (focus, columns, groupby=None, keep_others=False, func='max', **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelineX.extras.ops.pandas_ops.DfGetColIndexes (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelineX.extras.ops.pandas_ops.DfGetCols
    Bases: object

class pipelineX.extras.ops.pandas_ops.DfGetDummies (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelineX.extras.ops.pandas_ops.DfGroupby (**kwargs)
    Bases: pipelineX.extras.ops.pandas_ops.DfBaseMethod
    method = 'groupby'

class pipelineX.extras.ops.pandas_ops.DfHead (groupby=None,           columns=None,
                                                keep_others=False,      method=None,
                                                **kwargs)
    Bases: pipelineX.extras.ops.pandas_ops.DfBaseTask
    method = 'head'

class pipelineX.extras.ops.pandas_ops.DfMap (arg, prefix="", suffix="", **kwargs)
    Bases: object
    __init__ (arg, prefix="", suffix="", **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelineX.extras.ops.pandas_ops.DfMerge (**kwargs)
    Bases: object
```

```

    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfNgroup (groupby=None,          columns=None,
                                                keep_others=False,    method=None,
                                                **kwargs)

    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'ngroup'

class pipelinex.extras.ops.pandas_ops.DfPipe (groupby=None,          columns=None,
                                                keep_others=False,    method=None,
                                                **kwargs)

    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'pipe'

class pipelinex.extras.ops.pandas_ops.DfQuery (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'query'

class pipelinex.extras.ops.pandas_ops.DfRelative (focus, columns, groupby=None)
    Bases: object

    __init__ (focus, columns, groupby=None)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfRename (index=None,          columns=None,
                                                copy=True, level=None)

    Bases: object

    __init__ (index=None, columns=None, copy=True, level=None)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfResample (groupby=None,      columns=None,
                                                keep_others=False,    method=None,
                                                **kwargs)

    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'resample'

class pipelinex.extras.ops.pandas_ops.DfResetIndex (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'reset_index'

class pipelinex.extras.ops.pandas_ops.DfRolling (groupby=None,      columns=None,
                                                keep_others=False,    method=None,
                                                **kwargs)

    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'rolling'

class pipelinex.extras.ops.pandas_ops.DfRowApply (func, **kwargs)
    Bases: object

    __init__ (func, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfSample (**kwargs)
    Bases: object

    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

```

```
class pipelinex.extras.ops.pandas_ops.DfSelectDtypes (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'select_dtypes'

class pipelinex.extras.ops.pandas_ops.DfSelectDtypesCols (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfSelectDtypes

class pipelinex.extras.ops.pandas_ops.DfSetIndex (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'set_index'

class pipelinex.extras.ops.pandas_ops.DfShift (groupby=None,          columns=None,
                                                keep_others=False,      method=None,
                                                **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'shift'

class pipelinex.extras.ops.pandas_ops.DfSlice (**kwargs)
    Bases: object

    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfSortValues (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'sort_values'

class pipelinex.extras.ops.pandas_ops.DfSpatialFeatures (output='distance',
                                                         coo_cols=['X', 'Y'],
                                                         groupby=None,
                                                         ord=None,
                                                         unit_distance=1.0,
                                                         affinity_scale=1.0,    bi-
                                                         nary_affinity=False,
                                                         min_affinity=1e-06,
                                                         col_name_fmt='feat_{:03d}',
                                                         keep_others=True,
                                                         sort=True)
    Bases: object

    __init__ (output='distance', coo_cols=['X', 'Y'], groupby=None, ord=None, unit_distance=1.0, affinity_scale=1.0, binary_affinity=False, min_affinity=1e-06, col_name_fmt='feat_{:03d}', keep_others=True, sort=True)

        Available values for output: distance affinity laplacian eigenvalues eigenvectors n_connected

class pipelinex.extras.ops.pandas_ops.DfStrftime (cols, **kwargs)
    Bases: object

    __init__ (cols, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfTail (groupby=None,          columns=None,
                                                keep_others=False,      method=None,
                                                **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask

    method = 'tail'
```

```

class pipelinex.extras.ops.pandas_ops.DfToDatetime (cols=None, **kwargs)
    Bases: object
    __init__ (cols=None, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfToTimedelta (cols, **kwargs)
    Bases: object
    __init__ (cols, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfTotalSeconds (cols, **kwargs)
    Bases: object
    __init__ (cols, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.DfTransform (groupby=None, columns=None,
                                                    keep_others=False, method=None,
                                                    **kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseTask
    method = 'transform'

class pipelinex.extras.ops.pandas_ops.NestedDictToDf (row_oriented=True, index_name='index',
                                                       reset_index=True)
    Bases: object
    __init__ (row_oriented=True, index_name='index', reset_index=True)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.pandas_ops.SrMap (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

pipelinex.extras.ops.pandas_ops.affinity_matrix (coo_2darr, ord=None,
                                                  unit_distance=1.0, affinity_scale=1.0,
                                                  binary_affinity=False, min_affinity=1e-06,
                                                  zero_diag=True)

pipelinex.extras.ops.pandas_ops.degree_matrix (affinity_2darr)

pipelinex.extras.ops.pandas_ops.distance_matrix (coo_2darr, ord=None)

pipelinex.extras.ops.pandas_ops.distance_to_affinity (dist_2darr, unit_distance=1.0,
                                                       affinity_scale=1.0, binary_affinity=False,
                                                       min_affinity=1e-06)

pipelinex.extras.ops.pandas_ops.eigen (a, return_values=True, values_as_square_matrix=False,
                                       return_vectors=False, sort=False)

pipelinex.extras.ops.pandas_ops.laplacian_eigen (coo_2darr, return_values=True,
                                                  return_vectors=False, ord=None,
                                                  unit_distance=1.0, affinity_scale=1.0,
                                                  binary_affinity=False, min_affinity=1e-06,
                                                  sort=False)

```

```
pipelineX.extras.ops.pandas_ops.laplacian_matrix(coo_2darr, ord=None,  
                                                  unit_distance=1.0, affin-  
                                                  ity_scale=1.0, binary_affinity=False,  
                                                  min_affinity=1e-06)  
  
pipelineX.extras.ops.pandas_ops.nested_dict_to_df(d, row_oriented=True, in-  
                                                  dex_name='index', re-  
                                                  set_index=True)  
  
pipelineX.extras.ops.pandas_ops.row_vector_to_square_matrix(a)
```

### 13.1.1.1.1.66 pipelineX.extras.ops.pytorch\_ops module

```
class pipelineX.extras.ops.pytorch_ops.CrossEntropyLoss2d(weight=None,  
                                                         size_average=None,  
                                                         ignore_index=- 100,  
                                                         reduce=None, re-  
                                                         duction='mean', la-  
                                                         bel_smoothing=0.0)
```

Bases: `torch.nn.modules.loss.CrossEntropyLoss`

**forward** (*input*, *target*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**ignore\_index:** `int`

**label\_smoothing:** `float`

```
class pipelineX.extras.ops.pytorch_ops.ModuleAvg(*args:  
                                                  torch.nn.modules.module.Module)  
class pipelineX.extras.ops.pytorch_ops.ModuleAvg(arg: OrderedDict[str, Module])  
Bases: pipelineX.extras.ops.pytorch_ops.ModuleListMerge
```

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
class pipelineX.extras.ops.pytorch_ops.ModuleBottleneck2d(in_channels,  
                                                         out_channels, ker-  
                                                         nel_size=(1, 1),  
                                                         stride=(1, 1),  
                                                         mid_channels=None,  
                                                         batch_norm=None,  
                                                         activation=None,  
                                                         **kwargs)
```



Bases: `torch.nn.modules.container.Sequential`

**\_\_init\_\_** (*in\_channels, out\_channels, kernel\_size=(1, 1), stride=(1, 1), mid\_channels=None, batch\_norm=None, activation=None, \*\*kwargs*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**class** `pipelinex.extras.ops.pytorch_ops.ModuleConcat` (\*args: `torch.nn.modules.module.Module`)

**class** `pipelinex.extras.ops.pytorch_ops.ModuleConcat` (arg: `OrderedDict[str, Module]`)

Bases: `pipelinex.extras.ops.pytorch_ops.ModuleListMerge`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**class** `pipelinex.extras.ops.pytorch_ops.ModuleConcatSkip` (\*modules)

Bases: `pipelinex.extras.ops.pytorch_ops.ModuleConcat`

**\_\_init\_\_** (\*modules)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**class** `pipelinex.extras.ops.pytorch_ops.ModuleConvWrap` (*batchnorm=None, activation=None, \*args, \*\*kwargs*)

Bases: `torch.nn.modules.container.Sequential`

**\_\_init\_\_** (*batchnorm=None, activation=None, \*args, \*\*kwargs*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**core** = `None`

**class** `pipelinex.extras.ops.pytorch_ops.ModuleListMerge` (\*args:

`torch.nn.modules.module.Module`)

**class** `pipelinex.extras.ops.pytorch_ops.ModuleListMerge` (arg: `OrderedDict[str, Module]`)

Bases: `torch.nn.modules.container.Sequential`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**class** `pipelinex.extras.ops.pytorch_ops.ModuleProd` (\*args:

`torch.nn.modules.module.Module`)

**class** `pipelinex.extras.ops.pytorch_ops.ModuleProd` (arg: `OrderedDict[str, Module]`)

Bases: `pipelinex.extras.ops.pytorch_ops.ModuleListMerge`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
class pipelinex.extras.ops.pytorch_ops.ModuleSum(*args:
                                                    torch.nn.modules.module.Module)
class pipelinex.extras.ops.pytorch_ops.ModuleSum(arg: OrderedDict[str, Module])
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleListMerge
    forward(input)
        Defines the computation performed at every call.
        Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
class pipelinex.extras.ops.pytorch_ops.ModuleSumSkip(*modules)
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleSum
    __init__(*modules)
        Initializes internal Module state, shared by both nn.Module and ScriptModule.
class pipelinex.extras.ops.pytorch_ops.NLLoss(weight=None, size_average=None,
                                                ignore_index=-100, reduce=None,
                                                reduction='mean')
    Bases: torch.nn.modules.loss.NLLoss
    The negative likelihood loss. To compute Cross Entropy Loss, there are 3 options. NLLoss with
    torch.nn.Softmax torch.nn.NLLoss with torch.nn.LogSoftmax torch.nn.CrossEntropyLoss
    forward(input, target)
        Defines the computation performed at every call.
        Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**ignore\_index:** int

```
class pipelinex.extras.ops.pytorch_ops.Pool1dMixIn(keepdim=False)
    Bases: object
    __init__(keepdim=False)
        Initialize self. See help(type(self)) for accurate signature.
class pipelinex.extras.ops.pytorch_ops.Pool2dMixIn(keepdim=False)
    Bases: object
    __init__(keepdim=False)
        Initialize self. See help(type(self)) for accurate signature.
class pipelinex.extras.ops.pytorch_ops.Pool3dMixIn(keepdim=False)
    Bases: object
```

---

```
__init__(keepdim=False)
    Initialize self. See help(type(self)) for accurate signature.
```

```
class pipelinex.extras.ops.pytorch_ops.StatModule(dim, keepdim=False)
    Bases: torch.nn.modules.module.Module
```

```
__init__(dim, keepdim=False)
    Initializes internal Module state, shared by both nn.Module and ScriptModule.
```

```
training: bool
```

```
class pipelinex.extras.ops.pytorch_ops.StepBinary(size, desc=False, compare=None,
                                                    dtype=None)
    Bases: torch.nn.modules.module.Module
```

```
__init__(size, desc=False, compare=None, dtype=None)
    Initializes internal Module state, shared by both nn.Module and ScriptModule.
```

```
forward(input)
    Defines the computation performed at every call.

    Should be overridden by all subclasses.
```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
training: bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorAvgPool1d(batchnorm=None, activation=None, *args,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap
```

```
core
    alias of torch.nn.modules.pooling.AvgPool1d
```

```
training: bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorAvgPool2d(batchnorm=None, activation=None, *args,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap
```

```
core
    alias of torch.nn.modules.pooling.AvgPool2d
```

```
training: bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorAvgPool3d(batchnorm=None, activation=None, *args,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap
```

```
core
    alias of torch.nn.modules.pooling.AvgPool3d
```

```
training: bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorClamp(min=None, max=None)
    Bases: torch.nn.modules.module.Module
```

**\_\_init\_\_** (*min=None, max=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorClampMax` (*max=None*)

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_** (*max=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorClampMin` (*min=None*)

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_** (*min=None*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorConstantLinear` (*weight=1, bias=0*)

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_** (*weight=1, bias=0*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

training:    bool

class pipelineX.extras.ops.pytorch_ops.TensorConv1d(batchnorm=None,      activa-
                                                    tion=None, *args, **kwargs)
    Bases: pipelineX.extras.ops.pytorch_ops.ModuleConvWrap

    core
        alias of torch.nn.modules.conv.Conv1d

    training:    bool

class pipelineX.extras.ops.pytorch_ops.TensorConv2d(batchnorm=None,      activa-
                                                    tion=None, *args, **kwargs)
    Bases: pipelineX.extras.ops.pytorch_ops.ModuleConvWrap

    core
        alias of torch.nn.modules.conv.Conv2d

    training:    bool

class pipelineX.extras.ops.pytorch_ops.TensorConv3d(batchnorm=None,      activa-
                                                    tion=None, *args, **kwargs)
    Bases: pipelineX.extras.ops.pytorch_ops.ModuleConvWrap

    core
        alias of torch.nn.modules.conv.Conv3d

    training:    bool

class pipelineX.extras.ops.pytorch_ops.TensorCumsum(dim=1)
    Bases: torch.nn.modules.module.Module

    __init__(dim=1)
        Initializes internal Module state, shared by both nn.Module and ScriptModule.

    forward(input)
        Defines the computation performed at every call.

        Should be overridden by all subclasses.

```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```

training:    bool

class pipelineX.extras.ops.pytorch_ops.TensorExp
    Bases: torch.nn.modules.module.Module

    forward(input)
        Defines the computation performed at every call.

        Should be overridden by all subclasses.

```

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorFlatten`  
Bases: `torch.nn.modules.module.Module`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorForward` (*func=None*)  
Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_** (*func=None*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool1d` (*keepdim=False*)  
Bases: `pipelinex.extras.ops.pytorch_ops.Pool1dMixIn`, `pipelinex.extras.ops.pytorch_ops.TensorMean`

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool2d` (*keepdim=False*)  
Bases: `pipelinex.extras.ops.pytorch_ops.Pool2dMixIn`, `pipelinex.extras.ops.pytorch_ops.TensorMean`

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool3d` (*keepdim=False*)  
Bases: `pipelinex.extras.ops.pytorch_ops.Pool3dMixIn`, `pipelinex.extras.ops.pytorch_ops.TensorMean`

**training:** `bool`

```

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool1d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool1dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMax
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool2d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool2dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMax
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool3d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool3dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMax
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool1d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool1dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMin
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool2d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool2dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMin
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool3d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool3dMixIn, pipelinex.extras.ops.pytorch_ops.TensorMin
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool1d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool1dMixIn, pipelinex.extras.ops.pytorch_ops.TensorRange
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool2d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool2dMixIn, pipelinex.extras.ops.pytorch_ops.TensorRange
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool3d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool3dMixIn, pipelinex.extras.ops.pytorch_ops.TensorRange
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool1d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool1dMixIn, pipelinex.extras.ops.pytorch_ops.TensorSum
    training: bool

class pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool2d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool2dMixIn, pipelinex.extras.ops.pytorch_ops.TensorSum

```

**training:** bool

```
class pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool3d(keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.Pool3dMixIn, pipelinex.extras.ops.pytorch_ops.TensorSum
```

**training:** bool

```
class pipelinex.extras.ops.pytorch_ops.TensorIdentity
    Bases: torch.nn.modules.module.Module
```

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class pipelinex.extras.ops.pytorch_ops.TensorLog
    Bases: torch.nn.modules.module.Module
```

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class pipelinex.extras.ops.pytorch_ops.TensorMax(dim, keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.StatModule, torch.nn.modules.module.Module
```

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class pipelinex.extras.ops.pytorch_ops.TensorMaxPool1d(batchnorm=None, activation=None, *args,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap
```

**core**

alias of `torch.nn.modules.pooling.MaxPool1d`



**training:** bool

```
class pipelinetx.extras.ops.pytorch_ops.TensorMaxPool2d (batchnorm=None,      ac-
                                                         tivation=None,      *args,
                                                         **kwargs)
```

Bases: `pipelinetx.extras.ops.pytorch_ops.ModuleConvWrap`

**core**

alias of torch.nn.modules.pooling.MaxPool2d

**training:** bool

```
class pipelinetx.extras.ops.pytorch_ops.TensorMaxPool3d (batchnorm=None,      ac-
                                                         tivation=None,      *args,
                                                         **kwargs)
```

Bases: `pipelinetx.extras.ops.pytorch_ops.ModuleConvWrap`

**core**

alias of torch.nn.modules.pooling.MaxPool3d

**training:** bool

```
class pipelinetx.extras.ops.pytorch_ops.TensorMean (dim, keepdim=False)
```

Bases: `pipelinetx.extras.ops.pytorch_ops.StatModule`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class pipelinetx.extras.ops.pytorch_ops.TensorMin (dim, keepdim=False)
```

Bases: `pipelinetx.extras.ops.pytorch_ops.StatModule`, `torch.nn.modules.module.Module`

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the Module instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** bool

```
class pipelinetx.extras.ops.pytorch_ops.TensorNearestPad (lower=1, upper=1)
```

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_** (*lower=1, upper=1*)

Initializes internal Module state, shared by both nn.Module and ScriptModule.

**forward** (*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorProba(dim=1)`

Bases: `torch.nn.modules.module.Module`

**\_\_init\_\_**(*dim=1*)

Initializes internal Module state, shared by both `nn.Module` and `ScriptModule`.

**forward**(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorRange(dim, keepdim=False)`

Bases: `pipelinex.extras.ops.pytorch_ops.StatModule`, `torch.nn.modules.module.Module`

**forward**(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

**class** `pipelinex.extras.ops.pytorch_ops.TensorSkip`

Bases: `torch.nn.modules.module.Module`

**forward**(*input*)

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

**training:** `bool`

---

```
class pipelinex.extras.ops.pytorch_ops.TensorSlice (start=0, end=None, step=1)
    Bases: torch.nn.modules.module.Module
```

```
    __init__ (start=0, end=None, step=1)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
    forward (input)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
    training:    bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorSqueeze (dim=None)
    Bases: torch.nn.modules.module.Module
```

```
    __init__ (dim=None)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
    forward (input)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
    training:    bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorSum (dim, keepdim=False)
    Bases: pipelinex.extras.ops.pytorch_ops.StatModule
```

```
    forward (input)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
    training:    bool
```

```
class pipelinex.extras.ops.pytorch_ops.TensorUnsqueeze (dim)
    Bases: torch.nn.modules.module.Module
```

```
    __init__ (dim)
```

Initializes internal Module state, shared by both nn.Module and ScriptModule.

```
    forward (input)
```

Defines the computation performed at every call.

Should be overridden by all subclasses.

---

**Note:** Although the recipe for forward pass needs to be defined within this function, one should call the `Module` instance afterwards instead of this since the former takes care of running the registered hooks while the latter silently ignores them.

---

```
training:  bool

pipelinex.extras.ops.pytorch_ops.as_tuple(x)
pipelinex.extras.ops.pytorch_ops.element_wise_average(tt_list)
pipelinex.extras.ops.pytorch_ops.element_wise_prod(tt_list)
pipelinex.extras.ops.pytorch_ops.element_wise_sum(tt_list)
pipelinex.extras.ops.pytorch_ops.nl_loss(input, *args, **kwargs)
pipelinex.extras.ops.pytorch_ops.setup_conv_params(kernel_size=1,    dilation=None,
                                                    padding=None,    stride=None,
                                                    raise_error=False,    *args,
                                                    **kwargs)
pipelinex.extras.ops.pytorch_ops.step_binary(input, output_size, compare=<built-in
                                                    method ge of type object>)
pipelinex.extras.ops.pytorch_ops.tensor_max(input, dim, keepdim=False)
pipelinex.extras.ops.pytorch_ops.tensor_min(input, dim, keepdim=False)
pipelinex.extras.ops.pytorch_ops.to_array(input)
pipelinex.extras.ops.pytorch_ops.to_channel_first_tensor(a)
pipelinex.extras.ops.pytorch_ops.to_channel_last_tensor(a)
```

#### 13.1.1.1.1.67 pipelinex.extras.ops.shap\_ops module

```
class pipelinex.extras.ops.shap_ops.ExplainModel(**kwargs)
    Bases: object
    __init__(**kwargs)
        Initialize self. See help(type(self)) for accurate signature.

class pipelinex.extras.ops.shap_ops.Scale(**kwargs)
    Bases: object
    __init__(**kwargs)
        Initialize self. See help(type(self)) for accurate signature.
```

#### 13.1.1.1.1.68 pipelinex.extras.ops.skimage\_ops module

```
class pipelinex.extras.ops.skimage_ops.SkimageMarkBoundaries(**kwargs)
    Bases: pipelinex.extras.ops.skimage\_ops.SkimageSegmentationDictToDict
    fn = 'mark_boundaries'

class pipelinex.extras.ops.skimage_ops.SkimageSegmentationDictToDict(**kwargs)
    Bases: pipelinex.utils.DictToDict
    module = <module 'skimage.segmentation' from '/home/docs/checkouts/readthedocs.org/use
```

```

class pipelinex.extras.ops.skimage_ops.SkimageSegmentationFelzenszwalb(**kwargs)
    Bases: pipelinex.extras.ops.skimage_ops.SkimageSegmentationDictToDict
    fn = 'felzenszwalb'

class pipelinex.extras.ops.skimage_ops.SkimageSegmentationQuickshift(**kwargs)
    Bases: pipelinex.extras.ops.skimage_ops.SkimageSegmentationDictToDict
    fn = 'quickshift'

class pipelinex.extras.ops.skimage_ops.SkimageSegmentationSlic(**kwargs)
    Bases: pipelinex.extras.ops.skimage_ops.SkimageSegmentationDictToDict
    fn = 'slic'

class pipelinex.extras.ops.skimage_ops.SkimageSegmentationWatershed(**kwargs)
    Bases: pipelinex.extras.ops.skimage_ops.SkimageSegmentationDictToDict
    fn = 'watershed'

```

### 13.1.1.1.169 pipelinex.extras.ops.sklearn\_ops module

```

class pipelinex.extras.ops.sklearn_ops.DfBaseTransformer(cols=None,          tar-
                                                         get_col=None,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.sklearn_ops.ZeroToZeroTransformer

    __init__(cols=None, target_col=None, **kwargs)
        Initialize self. See help(type(self)) for accurate signature.

    fit(df)

    fit_transform(df)
        Fit to data, then transform it.

    Fits transformer to X and y with optional parameters fit_params and returns a transformed version of X.

```

#### Parameters

- **X** (array-like of shape (n\_samples, n\_features)) – Input samples.
- **y** (array-like of shape (n\_samples,) or (n\_samples, n\_outputs), default=None) – Target values (None for unsupervised transformations).
- **\*\*fit\_params** (dict) – Additional fit parameters.

**Returns** **X\_new** – Transformed array.

**Return type** ndarray array of shape (n\_samples, n\_features\_new)

**transform**(df)

```

class pipelinex.extras.ops.sklearn_ops.DfMinMaxScaler(cols=None, target_col=None,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.sklearn_ops.DfBaseTransformer, sklearn.
            preprocessing._data.MinMaxScaler

class pipelinex.extras.ops.sklearn_ops.DfQuantileTransformer(cols=None, tar-
                                                         get_col=None,
                                                         **kwargs)
    Bases: pipelinex.extras.ops.sklearn_ops.DfBaseTransformer, sklearn.
            preprocessing._data.QuantileTransformer

```

```
class pipelinex.extras.ops.sklearn_ops.DfStandardScaler(cols=None, tar-  
                                                    get_col=None, **kwargs)  
    Bases: pipelinex.extras.ops.sklearn_ops.DfBaseTransformer, sklearn.  
           preprocessing._data.StandardScaler
```

```
class pipelinex.extras.ops.sklearn_ops.DfTrainTestSplit(**kwargs)  
    Bases: object
```

```
    __init__(**kwargs)  
        Initialize self. See help(type(self)) for accurate signature.
```

```
class pipelinex.extras.ops.sklearn_ops.EstimatorTransformer  
    Bases: sklearn.base.TransformerMixin, sklearn.base.BaseEstimator
```

```
class pipelinex.extras.ops.sklearn_ops.ZeroToZeroTransformer(zero_to_zero=False,  
                                                            **kwargs)  
    Bases: pipelinex.extras.ops.sklearn_ops.EstimatorTransformer
```

```
    __init__(zero_to_zero=False, **kwargs)  
        Initialize self. See help(type(self)) for accurate signature.
```

```
    fit_transform(X)  
        Fit to data, then transform it.
```

Fits transformer to *X* and *y* with optional parameters *fit\_params* and returns a transformed version of *X*.

#### Parameters

- **X**(*array-like of shape (n\_samples, n\_features)*) – Input samples.
- **y** (*array-like of shape (n\_samples,) or (n\_samples, n\_outputs), default=None*) – Target values (None for unsupervised transformations).
- **\*\*fit\_params** (*dict*) – Additional fit parameters.

**Returns** **X\_new** – Transformed array.

**Return type** ndarray array of shape (n\_samples, n\_features\_new)

```
    transform(X)
```

```
pipelinex.extras.ops.sklearn_ops.extract_from_df(df, cols, target_col)
```

### 13.1.1.1.1.70 `pipelinex.extras.transformers` package

### 13.1.1.2 `pipelinex.flex_kedro` package

#### 13.1.1.2.1 Subpackages

#### 13.1.1.2.1.1 `pipelinex.flex_kedro.context` package

##### 13.1.1.2.1.2 Submodules

##### 13.1.1.2.1.3 `pipelinex.flex_kedro.context.context` module

##### 13.1.1.2.1.4 `pipelinex.flex_kedro.context.flexible_catalog_context` module

```
class pipelinex.flex_kedro.context.flexible_catalog_context.FlexibleCatalogContext (package_name, project_path, env=None, extra_params=None)
```

Bases: `kedro.framework.context.context.KedroContext`

Convert Kedrex's Syntactic Sugar to pure Kedro Catalog.

##### 13.1.1.2.1.5 `pipelinex.flex_kedro.context.flexible_context` module

```
class pipelinex.flex_kedro.context.flexible_context.FlexibleContext (*args, **kwargs)
    Bases:
        pipelinex.flex_kedro.context.flexible_parameters_context.FlexibleParametersContext,
        pipelinex.flex_kedro.context.flexible_catalog_context.FlexibleCatalogContext,
        pipelinex.flex_kedro.context.flexible_run_context.FlexibleRunContext
```

`project_name = 'KedroProject'`

`project_version = '0.17.7'`

```
class pipelinex.flex_kedro.context.flexible_context.MLflowFlexibleContext (*args, **kwargs)
```

Bases: `pipelinex.flex_kedro.context.flexible_context.FlexibleContext`

Deprecated alias for `FlexibleContext` for backward-compatibility

##### 13.1.1.2.1.6 `pipelinex.flex_kedro.context.flexible_parameters_context` module

```
class pipelinex.flex_kedro.context.flexible_parameters_context.FlexibleParametersContext (*args, **kwargs)
```

Bases: `kedro.framework.context.context.KedroContext`

`__init__` (\*args, \*\*kwargs)

Create a context object by providing the root of a Kedro project and the environment configuration subfolders (see `kedro.config.ConfigLoader`)

**Raises** `KedroContextError` – If there is a mismatch between Kedro project version and package version.

**Parameters**

- **package\_name** – Package name for the Kedro project the context is created for.
- **project\_path** – Project path to define the context for.
- **env** – Optional argument for configuration default environment to be used for running the pipeline. If not specified, it defaults to “local”.
- **extra\_params** – Optional dictionary containing extra project parameters. If specified, will update (and therefore take precedence over) the parameters retrieved from the project configuration.

**property params**

Read-only property referring to Kedro’s parameters for this context.

**Return type** Dict[str, Any]

**Returns**

Parameters defined in *parameters.yml* with the addition of any extra parameters passed at initialization.

**run** (\*args, \*\*kwargs)

Runs the pipeline with a specified runner.

**Parameters**

- **tags** – An optional list of node tags which should be used to filter the nodes of the Pipeline. If specified, only the nodes containing *any* of these tags will be run.
- **runner** – An optional parameter specifying the runner that you want to run the pipeline with.
- **node\_names** – An optional list of node names which should be used to filter the nodes of the Pipeline. If specified, only the nodes with these names will be run.
- **from\_nodes** – An optional list of node names which should be used as a starting point of the new Pipeline.
- **to\_nodes** – An optional list of node names which should be used as an end point of the new Pipeline.
- **from\_inputs** – An optional list of input datasets which should be used as a starting point of the new Pipeline.
- **to\_outputs** – An optional list of output datasets which should be used as an end point of the new Pipeline.
- **load\_versions** – An optional flag to specify a particular dataset version timestamp to load.
- **pipeline\_name** – Name of the Pipeline to execute. Defaults to “\_\_default\_\_”.

**Raises**

- **KedroContextError** – If the resulting Pipeline is empty or incorrect tags are provided.
- **Exception** – Any uncaught exception will be re-raised after being passed to “on\_pipeline\_error”.

**Returns** Any node outputs that cannot be processed by the DataCatalog. These are returned in a dictionary, where the keys are defined by the node outputs.

```
pipelinex.flex_kedro.context.flexible_parameters_context.import_modules(modules=None)
```



### 13.1.1.2.1.7 pipelinex.flex\_kedro.context.flexible\_run\_context module

```
class pipelinex.flex_kedro.context.flexible_run_context.FlexibleRunContext (package_name,  
project_path,  
env=None,  
ex-  
tra_params=None)
```

Bases: *pipelinex.flex\_kedro.context.save\_pipeline\_json\_context.*  
*SavePipelineJsonContext, pipelinex.flex\_kedro.context.flexible\_run\_context.*  
*StringRunnerOptionContext, pipelinex.flex\_kedro.context.*  
*flexible\_run\_context.OnlyMissingOptionContext*

```
class pipelinex.flex_kedro.context.flexible_run_context.OnlyMissingOptionContext (package_name,  
project_path,  
env=None,  
ex-  
tra_params=None)
```

Bases: *kedro.framework.context.context.KedroContext*

Users can override the remaining methods from the parent class here, or create new ones (e.g. as required by plugins)

```
run (tags=None, runner=None, node_names=None, from_nodes=None, to_nodes=None,  
from_inputs=None, load_versions=None, pipeline_name=None, only_missing=False)  
Runs the pipeline with a specified runner.
```

#### Parameters

- **tags** (Optional[Iterable[str]]) – An optional list of node tags which should be used to filter the nodes of the Pipeline. If specified, only the nodes containing *any* of these tags will be run.
- **runner** (Optional[AbstractRunner]) – An optional parameter specifying the runner that you want to run the pipeline with.
- **node\_names** (Optional[Iterable[str]]) – An optional list of node names which should be used to filter the nodes of the Pipeline. If specified, only the nodes with these names will be run.
- **from\_nodes** (Optional[Iterable[str]]) – An optional list of node names which should be used as a starting point of the new Pipeline.
- **to\_nodes** (Optional[Iterable[str]]) – An optional list of node names which should be used as an end point of the new Pipeline.
- **from\_inputs** (Optional[Iterable[str]]) – An optional list of input datasets which should be used as a starting point of the new Pipeline.
- **load\_versions** (Optional[Dict[str, str]]) – An optional flag to specify a particular dataset version timestamp to load.
- **pipeline\_name** (Optional[str]) – Name of the Pipeline to execute. Defaults to “\_\_default\_\_”.
- **only\_missing** (bool) – An option to run only missing nodes.

#### Raises

- **KedroContextError** – If the resulting Pipeline is empty or incorrect tags are provided.

- **Exception** – Any uncaught exception will be re-raised after being passed to ``on\_pipeline\_error``.

**Return type** Dict[str, Any]

**Returns** Any node outputs that cannot be processed by the DataCatalog. These are returned in a dictionary, where the keys are defined by the node outputs.

```
class pipelinex.flex_kedro.context.flexible_run_context.StringRunnerOptionContext (package_name, project_path, env=None, extra_params=None)
```

Bases: kedro.framework.context.context.KedroContext

Allow to specify runner by string.

**run** (\*args, runner=None, \*\*kwargs)

Runs the pipeline with a specified runner.

#### Parameters

- **tags** – An optional list of node tags which should be used to filter the nodes of the Pipeline. If specified, only the nodes containing *any* of these tags will be run.
- **runner** (Union[AbstractRunner, str, None]) – An optional parameter specifying the runner that you want to run the pipeline with.
- **node\_names** – An optional list of node names which should be used to filter the nodes of the Pipeline. If specified, only the nodes with these names will be run.
- **from\_nodes** – An optional list of node names which should be used as a starting point of the new Pipeline.
- **to\_nodes** – An optional list of node names which should be used as an end point of the new Pipeline.
- **from\_inputs** – An optional list of input datasets which should be used as a starting point of the new Pipeline.
- **to\_outputs** – An optional list of output datasets which should be used as an end point of the new Pipeline.
- **load\_versions** – An optional flag to specify a particular dataset version timestamp to load.
- **pipeline\_name** – Name of the Pipeline to execute. Defaults to “\_\_default\_\_”.

#### Raises

- **KedroContextError** – If the resulting Pipeline is empty or incorrect tags are provided.
- **Exception** – Any uncaught exception will be re-raised after being passed to ``on\_pipeline\_error``.

**Return type** Dict[str, Any]

**Returns** Any node outputs that cannot be processed by the DataCatalog. These are returned in a dictionary, where the keys are defined by the node outputs.

### 13.1.1.2.1.8 `pipelinex.flex_kedro.context.save_pipeline_json_context` module

`class pipelinex.flex_kedro.context.save_pipeline_json_context.SavePipelineJsonContext` (package

project  
env=N  
ex-  
tra\_pa

Bases: `kedro.framework.context.context.KedroContext`

### 13.1.1.2.1.9 `pipelinex.flex_kedro.pipeline` package

#### 13.1.1.2.1.10 Submodules

### 13.1.1.2.1.11 `pipelinex.flex_kedro.pipeline.pipeline` module

`class pipelinex.flex_kedro.pipeline.pipeline.FlexiblePipeline` (`nodes`, \*, `parameters_in_inputs=False`, `module=""`, `decorator=[]`, `**kwargs`)

Bases: `kedro.pipeline.pipeline.Pipeline`

`__init__` (`nodes`, \*, `parameters_in_inputs=False`, `module=""`, `decorator=[]`, `**kwargs`)

Initialise Pipeline with a list of Node instances.

#### Parameters

- **nodes** – The iterable of nodes the Pipeline will be made of. If you provide pipelines among the list of nodes, those pipelines will be expanded and all their nodes will become part of this new pipeline.
- **tags** – Optional set of tags to be applied to all the pipeline nodes.

#### Raises

- **ValueError** – When an empty list of nodes is provided, or when not all nodes have unique names.
- **CircularDependencyError** – When visiting all the nodes is not possible due to the existence of a circular dependency.
- **OutputNotUniqueError** – When multiple Node instances produce the same output.
- **ConfirmNotUniqueError** – When multiple Node instances attempt to confirm the same dataset.

Example:

```
from kedro.pipeline import Pipeline
from kedro.pipeline import node

# In the following scenario first_ds and second_ds
# are data sets provided by io. Pipeline will pass these
# data sets to first_node function and provides the result
# to the second_node as input.
```

(continues on next page)

(continued from previous page)

```

def first_node(first_ds, second_ds):
    return dict(third_ds=first_ds+second_ds)

def second_node(third_ds):
    return third_ds

pipeline = Pipeline([
    node(first_node, ['first_ds', 'second_ds'], ['third_ds']),
    node(second_node, dict(third_ds='third_ds'), ['fourth_ds'])])

pipeline.describe()

```

### 13.1.1.2.1.12 `pipeline.flex_kedro.pipeline.sub_pipeline` module

```

class pipeline.flex_kedro.pipeline.sub_pipeline.SubPipeline (inputs=None,
                                                             outputs=None,
                                                             func=None,
                                                             module="", decorator=None,
                                                             intermediate_node_name_fmt='{__}_{:03d}',
                                                             **kwargs)

```

Bases: `kedro.pipeline.pipeline.Pipeline`

**\_\_init\_\_** (*inputs=None, outputs=None, func=None, module="", decorator=None, intermediate\_node\_name\_fmt='{\_\_}\_{:03d}', \*\*kwargs*)  
 Initialise Pipeline with a list of Node instances.

#### Parameters

- **nodes** – The iterable of nodes the Pipeline will be made of. If you provide pipelines among the list of nodes, those pipelines will be expanded and all their nodes will become part of this new pipeline.
- **tags** – Optional set of tags to be applied to all the pipeline nodes.

#### Raises

- **ValueError** – When an empty list of nodes is provided, or when not all nodes have unique names.
- **CircularDependencyError** – When visiting all the nodes is not possible due to the existence of a circular dependency.
- **OutputNotUniqueError** – When multiple Node instances produce the same output.
- **ConfirmNotUniqueError** – When multiple Node instances attempt to confirm the same dataset.

Example:

```

from kedro.pipeline import Pipeline
from kedro.pipeline import node

# In the following scenario first_ds and second_ds
# are data sets provided by io. Pipeline will pass these

```

(continues on next page)

(continued from previous page)

```

# data sets to first_node function and provides the result
# to the second_node as input.

def first_node(first_ds, second_ds):
    return dict(third_ds=first_ds+second_ds)

def second_node(third_ds):
    return third_ds

pipeline = Pipeline([
    node(first_node, ['first_ds', 'second_ds'], ['third_ds']),
    node(second_node, dict(third_ds='third_ds'), ['fourth_ds'])])

pipeline.describe()

```

### 13.1.1.2.2 Submodules

#### 13.1.1.2.3 `pipelinex.flex_kedro.configure` module

`pipelinex.flex_kedro.configure.configure_source` (*project\_path*, *source\_dir*='src')

`pipelinex.flex_kedro.configure.load_context` (*project\_path*, *\*\*kwargs*)

**Return type** `KedroContext`

### 13.1.1.3 `pipelinex.hatch_dict` package

#### 13.1.1.3.1 Submodules

#### 13.1.1.3.2 `pipelinex.hatch_dict.hatch_dict` module

**class** `pipelinex.hatch_dict.hatch_dict.Construct` (*obj*)

Bases: `object`

**\_\_init\_\_** (*obj*)

Initialize self. See help(type(self)) for accurate signature.

**class** `pipelinex.hatch_dict.hatch_dict.Get` (*\*args*, *\*\*kwargs*)

Bases: `pipelinex.hatch_dict.hatch_dict.Method`

**method** = 'get'

**class** `pipelinex.hatch_dict.hatch_dict.HatchDict` (*egg*, *lookup*=`{}`, *support\_nested\_keys*=`True`, *self\_lookup\_key*='\$', *support\_import*=`True`, *additional\_import\_modules*=['`pipelinex`'], *obj\_key*='=', *eval\_parentheses*=`True`)

Bases: `object`

**\_\_init\_\_** (*egg*, *lookup*=`{}`, *support\_nested\_keys*=`True`, *self\_lookup\_key*='\$', *support\_import*=`True`, *additional\_import\_modules*=['`pipelinex`'], *obj\_key*='=', *eval\_parentheses*=`True`)

Initialize self. See help(type(self)) for accurate signature.

**get** (*key*=`None`, *default*=`None`, *lookup*=`{}`)

**Return type** Any

**get\_params()**

**items()**

**keys()**

**class** `pipelinex.hatch_dict.hatch_dict.Method(*args, **kwargs)`  
Bases: `object`

**\_\_init\_\_**(\*args, \*\*kwargs)  
Initialize self. See `help(type(self))` for accurate signature.

**method** = None

**class** `pipelinex.hatch_dict.hatch_dict.ToPipeline(*args)`  
Bases: `object`

**\_\_init\_\_**(\*args)  
Initialize self. See `help(type(self))` for accurate signature.

`pipelinex.hatch_dict.hatch_dict.dot_flatten(d)`

`pipelinex.hatch_dict.hatch_dict.feed(func, args)`

`pipelinex.hatch_dict.hatch_dict.load_obj(obj_path, default_obj_path="")`

Extract an object from a given path. :type `obj_path`: `str` :param `obj_path`: Path to an object to be extracted, including the object name. :type `default_obj_path`: `str` :param `default_obj_path`: Default object path.

**Return type** Any

**Returns** Extracted object.

**Raises** **AttributeError** – When the object does not have the given named attribute.

`pipelinex.hatch_dict.hatch_dict.pass_(*argsignore, **kwargsignore)`

`pipelinex.hatch_dict.hatch_dict.pass_through(*args, **kwargs)`

#### 13.1.1.4 pipelinux.mlflow\_on\_kedro package

##### 13.1.1.4.1 Subpackages

##### 13.1.1.4.1.1 pipelinux.mlflow\_on\_kedro.datasets package

##### 13.1.1.4.1.2 Subpackages

##### 13.1.1.4.1.3 pipelinux.mlflow\_on\_kedro.datasets.mlflow package

##### 13.1.1.4.1.4 Submodules

##### 13.1.1.4.1.5 pipelinux.mlflow\_on\_kedro.datasets.mlflow.mlflow\_dataset module

```
class pipelinux.mlflow_on_kedro.datasets.mlflow.mlflow_dataset.MLflowDataSet (dataset=None,  
filepath=None,  
dataset_name=None,  
sav-  
ing_tracking_uri=None,  
sav-  
ing_experiment_name=None,  
sav-  
ing_run_id=None,  
load-  
ing_tracking_uri=None,  
load-  
ing_run_id=None,  
cached=True,  
copy_mode=None,  
file_caching=True)
```

Bases: kedro.io.core.AbstractDataSet

MLflowDataSet saves data to, and loads data from MLflow.

You can also specify a MLflowDataSet in catalog.yml

Example:

```
test_ds:  
  type: MLflowDataSet  
  dataset: pkl
```

```
__init__ (dataset=None, filepath=None, dataset_name=None, saving_tracking_uri=None, sav-  
ing_experiment_name=None, saving_run_id=None, loading_tracking_uri=None, load-  
ing_run_id=None, cached=True, copy_mode=None, file_caching=True)
```

##### Parameters

- **dataset** (Union[AbstractDataSet, Dict, str, None]) – Specify how to treat the dataset as an MLflow metric, parameter, or artifact.
  - If set to “p”, the value will be saved/loaded as an MLflow parameter (string).
  - If set to “m”, the value will be saved/loaded as an MLflow metric (numeric).
  - If set to “a”, the value will be saved/loaded based on the data type.

- \* If the data type is either {float, int}, the value will be saved/loaded as an MLflow metric.
- \* If the data type is either {str, list, tuple, set}, the value will be saved/load as an MLflow parameter.
- \* If the data type is dict, the value will be flattened with dot (“.”) as the separator and then saved/loaded as either an MLflow metric or parameter based on each data type as explained above.
- If set to either {“json”, “csv”, “xls”, “parquet”, “png”, “jpg”, “jpeg”, “img”, “pkl”, “txt”, “yaml”, “yml”}, the backend dataset instance will be created accordingly to save/load as an MLflow artifact.
- If set to a Kedro DataSet object or a dictionary, it will be used as the backend dataset to save/load as an MLflow artifact.
- If set to None (default), MLflow logging will be skipped.
- **filepath** (Optional[str]) – File path, usually in local file system, to save to and load from. Used only if the dataset arg is a string. If None (default), <temp directory>/<dataset\_name arg>.<dataset arg> is used.
- **dataset\_name** (Optional[str]) – Used only if the dataset arg is a string and filepath arg is None. If None (default), Python object ID is used, but will be overwritten by MLflowCatalogLoggerHook.
- **saving\_tracking\_uri** (Optional[str]) – MLflow Tracking URI to save to. If None (default), MLFLOW\_TRACKING\_URI environment variable is used.
- **saving\_experiment\_name** (Optional[str]) – MLflow experiment name to save to. If None (default), new experiment will not be created or started. Ignored if saving\_run\_id is set.
- **saving\_run\_id** (Optional[str]) – An existing MLflow experiment run ID to save to. If None (default), no existing experiment run will be resumed.
- **loading\_tracking\_uri** (Optional[str]) – MLflow Tracking URI to load from. If None (default), MLFLOW\_TRACKING\_URI environment variable is used.
- **loading\_run\_id** (Optional[str]) – MLflow experiment run ID to load from. If None (default), current active run ID will be used if available.
- **caching** (bool) – Enable caching if parallel runner is not used. True in default.
- **copy\_mode** (Optional[str]) – The copy mode used to copy the data. Possible values are: “deepcopy”, “copy” and “assign”. If not provided, it is inferred based on the data type. Ignored if caching arg is False.
- **file\_caching** (bool) – Attempt to use the file at filepath when loading if no cache found in memory. True in default.



#### 13.1.1.4.1.6 `pipelinex.mlflow_on_kedro.decorators` package

#### 13.1.1.4.1.7 Submodules

#### 13.1.1.4.1.8 `pipelinex.mlflow_on_kedro.decorators.mlflow_logger` module

`pipelinex.mlflow_on_kedro.decorators.mlflow_logger.mlflow_log_time` (*func*)

A function decorator which logs the time taken for executing a function.

**Parameters** `func` (Callable) – The function to be logged.

**Return type** Callable

**Returns** A wrapped function, which will execute the provided function and log the running time.

#### 13.1.1.4.1.9 `pipelinex.mlflow_on_kedro.hooks` package

#### 13.1.1.4.1.10 Subpackages

#### 13.1.1.4.1.11 `pipelinex.mlflow_on_kedro.hooks.mlflow` package

#### 13.1.1.4.1.12 Submodules

#### 13.1.1.4.1.13 `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_artifacts_logger` module

**class** `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLogger`

Bases: `object`

Logs artifacts of specified file paths and dataset names to MLflow

**\_\_init\_\_** (*filepaths\_before\_pipeline\_run=None*, *filepaths\_after\_pipeline\_run=None*, *datasets\_after\_node\_run=None*, *enable\_mlflow=True*)

**Parameters**

- **filepaths\_before\_pipeline\_run** (Optional[List[str]]) – The file paths of artifacts to log before the pipeline is run.
- **filepaths\_after\_pipeline\_run** (Optional[List[str]]) – The file paths of artifacts to log after the pipeline is run.
- **datasets\_after\_node\_run** (Optional[List[str]]) – The dataset names to log after the node is run.
- **enable\_mlflow** (bool) – Enable logging to MLflow.

**after\_node\_run** (*node*, *catalog*, *inputs*, *outputs*)

**after\_pipeline\_run** (*run\_params*, *pipeline*, *catalog*)

**before\_pipeline\_run** (*run\_params*, *pipeline*, *catalog*)

#### 13.1.1.4.1.14 `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger` module

```
class pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook (uri=None, experiment_name=None, artifact_location=None, run_name=None, run_id=None, nested=False, tags=None, offset_hours=0, enable_logging_time_begin=True, enable_logging_time_end=True, enable_logging_time=True, logging_kedro_run_params=[], enable_mlflow=True)
```

Bases: `object`

Configures and logs duration time for the pipeline to MLflow

```
__init__ (uri=None, experiment_name=None, artifact_location=None, run_name=None, run_id=None, nested=False, tags=None, offset_hours=0, enable_logging_time_begin=True, enable_logging_time_end=True, enable_logging_time=True, logging_kedro_run_params=[], enable_mlflow=True)
```

##### Parameters

- **uri** (Optional[str]) – The MLflow tracking server URI. *uri* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.set\\_tracking\\_uri](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.set_tracking_uri)
- **experiment\_name** (Optional[str]) – The experiment name. *name* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.create\\_experiment](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.create_experiment)
- **artifact\_location** (Optional[str]) – *artifact\_location* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.create\\_experiment](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.create_experiment)
- **run\_name** (Optional[str]) – Shown as ‘Run Name’ in MLflow UI.
- **run\_id** (Optional[str]) – An existing MLflow experiment run UUID instead of letting MLflow create a new run under the *experiment\_name*. *run\_id* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.start\\_run](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.start_run)
- **nested** (bool) – *nested* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.start\\_run](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.start_run)
- **tags** (Optional[Dict[str, Any]]) – *tags* arg fed to: [https://www.mlflow.org/docs/latest/python\\_api/mlflow.html#mlflow.start\\_run](https://www.mlflow.org/docs/latest/python_api/mlflow.html#mlflow.start_run)

- **offset\_hours** (float) – The offset hour (e.g. 0 for UTC+00:00) to log in MLflow. 0 in default.
- **enable\_logging\_time\_begin** (bool) – Enable logging the time the Kedro pipeline began. True in default.
- **enable\_logging\_time\_end** (bool) – Enable logging the time the Kedro pipeline ended. True in default.
- **enable\_logging\_time** (bool) – Enable logging the time duration the Kedro pipeline ran. True in default.
- **logging\_kedro\_run\_params** (Union[List[str], str]) – List of Kedro Run Params to log to MLflow or “\_\_ALL\_\_” to log all. [] (Empty) in default.
- **enable\_mlflow** (bool) – Enable configuring and logging to MLflow.

**after\_catalog\_created**()

**after\_pipeline\_run**(run\_params, pipeline, catalog)

**before\_pipeline\_run**(run\_params, pipeline, catalog)

```

pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger.get_timestamp(dt=None,
                                off-
                                set_hours=0,
                                fmt='%Y-
                                %m-
                                %dT%H:%M:%S')

pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger.get_timestamp_int(dt=None,
                                off-
                                set_hours=0)

pipeline.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger.get_timestamps(dt=None,
                                off-
                                set_hours=0)

```

#### 13.1.1.4.1.15 pipeline.mlflow\_on\_kedro.hooks.mlflow.mlflow\_catalog\_logger module

**class** pipeline.mlflow\_on\_kedro.hooks.mlflow.mlflow\_catalog\_logger.MLflowCatalogLoggerHook

Bases: object

Logs datasets to MLflow

**\_\_init\_\_**(auto=True, mlflow\_catalog={}, enable\_mlflow=True)

##### Parameters

- **auto** (bool) – If True, each dataset (Python func input/output) not listed in the catalog
- **be logged following the same rule as "a" option below.**  
(will) –
- **mlflow\_catalog** (Dict[str, Union[str, AbstractDataSet]]) – [Deprecated in favor of MLflowDataSet] Specify how to log each dataset
- **func input/output** ((Python) –

- If set to “p”, the value will be saved/loaded as an MLflow parameter (string).
- If set to “m”, the value will be saved/loaded as an MLflow metric (numeric).
- If set to “a”, the value will be saved/loaded based on the data type.
  - \* If the data type is either {float, int}, the value will be saved/loaded as an MLflow metric.
  - \* If the data type is either {str, list, tuple, set}, the value will be saved/load as an MLflow parameter.
  - \* If the data type is dict, the value will be flattened with dot (“.”) as the separator and then saved/loaded as either an MLflow metric or parameter based on each data type as explained above.
- If set to either {“json”, “csv”, “xls”, “parquet”, “png”, “jpg”, “jpeg”, “img”, “pkl”, “txt”, “yaml”, “yml”}, the backend dataset instance will be created accordingly to save/load as an MLflow artifact.
- If set to a Kedro DataSet object or a dictionary, it will be used as the backend dataset to save/load as an MLflow artifact.
- If set to None (default), MLflow logging will be skipped.
- **enable\_mlflow** (bool) – Enable logging to MLflow.

**after\_node\_run** (node, catalog, inputs, outputs)

**before\_pipeline\_run** (run\_params, pipeline, catalog)

**Return type** None

```
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger.get_kedro_runner()
```

```
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger.mlflow_log_dataset(dataset,
en-
able_mlflow=True)
```

```
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger.running_parallel()
```

#### 13.1.1.4.1.16 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_datasets\_logger module

```
class pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger.MLflowDataSetsLoggerHook
```

Bases: object

Logs datasets of (list of) float/int and str classes to MLflow

**\_\_init\_\_** (enable\_mlflow=True)

**Parameters** **enable\_mlflow** (bool) – Enable logging to MLflow.

**after\_node\_run** (node, catalog, inputs, outputs)

```
class pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger.MLflowOutputsLoggerHook
```

Bases: `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger.MLflowDataSetsLoggerHook`

Deprecated alias for `MLflowOutputsLoggerHook`

### 13.1.1.4.1.17 `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger` module

**class** `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger.MLflowEnvVarsLoggerHook`

Bases: `object`

Logs environment variables to MLflow

**\_\_init\_\_** (*param\_env\_vars=None, metric\_env\_vars=None, prefix=None, enable\_mlflow=True*)

#### Parameters

- **param\_env\_vars** (`Optional[List[str]]`) – Environment variables to log to MLflow as parameters
- **metric\_env\_vars** (`Optional[List[str]]`) – Environment variables to log to MLflow as metrics
- **prefix** (`Optional[str]`) – Prefix to add to each name of MLflow parameters and metrics (“env.” in default)
- **enable\_mlflow** (`bool`) – Enable logging to MLflow.

**after\_pipeline\_run** ()

**before\_pipeline\_run** ()

`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger.env_vars_to_dict` (*env\_vars=[],*  
*pre-*  
*fix=""*)

`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger.log_metric_env_vars` (*env\_vars=[],*  
*pre-*  
*fix="",*  
*en-*  
*able\_mlflow=*

`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger.log_param_env_vars` (*env\_vars=[],*  
*pre-*  
*fix="",*  
*en-*  
*able\_mlflow=*

13.1.1.4.1.18 `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger` module

```
class pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook (gantt_filepath=None, gantt_params={}, metric_name_prefix='_time_to_run', task_name_func=<function _get_task_name>, time_log_filepath=None, enable_plotly=True, enable_mlflow=True)
```

Bases: `object`

Logs duration time to run each node (task) to MLflow. Optionally, the execution logs can be visualized as a Gantt chart by `plotly.figure_factory.create_gantt` ([https://plotly.github.io/plotly.py-docs/generated/plotly.figure\\_factory.create\\_gantt.html](https://plotly.github.io/plotly.py-docs/generated/plotly.figure_factory.create_gantt.html)) if `plotly` is installed.

```
__init__ (gantt_filepath=None, gantt_params={}, metric_name_prefix='_time_to_run',
          task_name_func=<function _get_task_name>, time_log_filepath=None,
          enable_plotly=True, enable_mlflow=True)
```

**Parameters**

- **gantt\_filepath** (Optional[str]) – File path to save the generated gantt chart.
- **gantt\_params** (Dict[str, Any]) – Args fed to: [https://plotly.github.io/plotly.py-docs/generated/plotly.figure\\_factory.create\\_gantt.html](https://plotly.github.io/plotly.py-docs/generated/plotly.figure_factory.create_gantt.html)
- **metric\_name\_prefix** (str) – Prefix for the metric names. The metric names are `metric_name_prefix` concatenated with the string returned by `task_name_func`.
- **task\_name\_func** (Callable[[Node], str]) – Callable to return the task name using `kedro.pipeline.node.Node` object.
- **time\_log\_filepath** (Optional[str]) – File path to save the time log in JSON format.
- **enable\_plotly** (bool) – Enable visualization of logged time as a gantt chart.
- **enable\_mlflow** (bool) – Enable logging to MLflow.

```
after_node_run (node, catalog, inputs, outputs)
```

```
after_pipeline_run (run_params, pipeline, catalog)
```

```
before_node_run (node, catalog, inputs)
```

```
load_time_dict (key)
```

```
update_time_dict (key, d)
```

```
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger.dump_dict (filepath, d)
```

```
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger.load_dict (filepath)
```

#### 13.1.1.4.1.19 `pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils` module

```

pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_end_run(enable_mlflow=True)
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_log_artifacts(paths,
                                                                    ar-
                                                                    ti-
                                                                    fact_path=None,
                                                                    en-
                                                                    able_mlflow=True)
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_log_metrics(metrics,
                                                                    step=None,
                                                                    en-
                                                                    able_mlflow=True)
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_log_params(params,
                                                                    en-
                                                                    able_mlflow=True)
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_log_values(d, en-
                                                                    able_mlflow=True)
pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils.mlflow_start_run(uri=None,
                                                                    run_id=None,
                                                                    experi-
                                                                    ment_name=None,
                                                                    arti-
                                                                    fact_location=None,
                                                                    run_name=None,
                                                                    nested=False,
                                                                    tags=None,
                                                                    en-
                                                                    able_mlflow=True)

```

#### 13.1.1.4.1.20 `pipelinex.mlflow_on_kedro.transformers` package

##### 13.1.1.4.1.21 Subpackages

##### 13.1.1.4.1.22 `pipelinex.mlflow_on_kedro.transformers.mlflow` package

##### 13.1.1.4.1.23 Submodules

#### 13.1.1.4.1.24 `pipelinex.mlflow_on_kedro.transformers.mlflow.mlflow_io_time_logger` module

```
class pipelinex.mlflow_on_kedro.transformers.mlflow.mlflow_io_time_logger.MLflowIOTimeLogger
```

Bases: `kedro.io.transformers.AbstractTransformer`

Log duration time to load and save each dataset.

```
__init__(enable_mlflow=True, metric_name_prefix='_time_to_')
```

##### Parameters

- **enable\_mlflow** (bool) – Enable logging to MLflow.

- **metric\_name\_prefix** (str) – Prefix for the metric names. The metric names are *metric\_name\_prefix* concatenated with ‘load <data\_set\_name>’ or ‘save <data\_set\_name>’

**load** (data\_set\_name, load)

This method will be deprecated in Kedro 0.18.0 in favour of the Dataset Hooks *before\_dataset\_loaded* and *after\_dataset\_loaded*.

Wrap the loading of a dataset. Call `load` to get the data from the data set / next transformer.

#### Parameters

- **data\_set\_name** (str) – The name of the data set being loaded.
- **load** (Callable[[], Any]) – A callback to retrieve the data being loaded from the data set / next transformer.

**Return type** Any

**Returns** The loaded data.

**save** (data\_set\_name, save, data)

This method will be deprecated in Kedro 0.18.0 in favour of the Dataset Hooks *before\_dataset\_saved* and *after\_dataset\_saved*.

Wrap the saving of a dataset. Call `save` to pass the data to the data set / next transformer.

#### Parameters

- **data\_set\_name** (str) – The name of the data set being saved.
- **save** (Callable[[Any], None]) – A callback to pass the data being saved on to the data set / next transformer.
- **data** (Any) – The data being saved

**Return type** None

## 13.1.2 Submodules

### 13.1.3 `pipelineutils` module

```
class pipelineutils.DictToDict (**kwargs)
```

Bases: object

```
__init__ (**kwargs)
```

Initialize self. See help(type(self)) for accurate signature.

```
fn = None
```

```
module = None
```

```
class pipelineutils.ItemGetter (item)
```

Bases: object

```
__init__ (item)
```

Initialize self. See help(type(self)) for accurate signature.

```
class pipelineutils.TransformCompose (transforms)
```

Bases: object

```
__init__ (transforms)
```

Initialize self. See help(type(self)) for accurate signature.



`pipelineutils.dict_io` (*func*)

**Return type** Callable

`pipelineutils.dict_of_list_to_list_of_dict` (*dict\_of\_list*)

`pipelineutils.list_of_dict_to_dict_of_list` (*list\_of\_dict*)



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### p

[pipeline](#), 46  
[pipeline.extras](#), 46  
[pipeline.extras.datasets](#), 46  
[pipeline.extras.datasets.core](#), 57  
[pipeline.extras.datasets.httpx](#), 46  
[pipeline.extras.datasets.httpx.async\\_api\\_dataset](#), 46  
[pipeline.extras.datasets.opencv](#), 47  
[pipeline.extras.datasets.opencv.images\\_dataset](#), 47  
[pipeline.extras.datasets.pandas](#), 47  
[pipeline.extras.datasets.pandas.csv\\_local](#), 47  
[pipeline.extras.datasets.pandas.efficient\\_csv\\_local](#), 49  
[pipeline.extras.datasets.pandas.fixed\\_width\\_csv\\_dataset](#), 49  
[pipeline.extras.datasets.pandas.histogram](#), 50  
[pipeline.extras.datasets.pandas.pandas\\_cat\\_matrix](#), 50  
[pipeline.extras.datasets.pandas.pandas\\_describe](#), 51  
[pipeline.extras.datasets.pandas.pandas\\_profiling](#), 51  
[pipeline.extras.datasets.pandas.pandas\\_profiling.pandas\\_profiling](#), 51  
[pipeline.extras.datasets.pillow](#), 52  
[pipeline.extras.datasets.pillow.images\\_dataset](#), 52  
[pipeline.extras.datasets.requests](#), 53  
[pipeline.extras.datasets.requests.api\\_dataset](#), 53  
[pipeline.extras.datasets.seaborn](#), 56  
[pipeline.extras.datasets.seaborn.seaborn\\_pairplot](#), 56  
[pipeline.extras.datasets.torchvision](#), 56  
[pipeline.extras.datasets.torchvision.iterable\\_images\\_dataset](#), 56  
[pipeline.extras.decorators](#), 57  
[pipeline.extras.decorators.decorators](#), 57  
[pipeline.extras.decorators.memory\\_profiler](#), 57  
[pipeline.extras.decorators.nvml\\_profiler](#), 57  
[pipeline.extras.decorators.pandas\\_decorators](#), 58  
[pipeline.extras.hooks](#), 58  
[pipeline.extras.hooks.add\\_catalog\\_dict](#), 58  
[pipeline.extras.hooks.add\\_transformers](#), 58  
[pipeline.extras.ops](#), 59  
[pipeline.extras.ops.allennlp\\_ops](#), 64  
[pipeline.extras.ops.parse\\_args\\_ops](#), 64  
[pipeline.extras.ops.ignite](#), 59  
[pipeline.extras.ops.ignite.declaratives](#), 59  
[pipeline.extras.ops.ignite.declaratives.declarative](#), 59  
[pipeline.extras.ops.ignite.handlers](#), 61  
[pipeline.extras.ops.ignite.handlers.flexible\\_checks](#), 61  
[pipeline.extras.ops.ignite.handlers.time\\_limit](#), 62  
[pipeline.extras.ops.ignite.metrics](#), 62  
[pipeline.extras.ops.ignite.metrics.cohen\\_kappa\\_score](#), 62  
[pipeline.extras.ops.ignite.metrics.fbeta\\_score](#), 63  
[pipeline.extras.ops.ignite.metrics.utils](#), 64  
[pipeline.extras.ops.numpy\\_ops](#), 64  
[pipeline.extras.ops.opencv\\_ops](#), 64  
[pipeline.extras.ops.pandas\\_ops](#), 68  
[pipeline.extras.ops.pytorch\\_ops](#), 74  
[pipeline.extras.ops.shap\\_ops](#), 86  
[pipeline.extras.ops.skimage\\_ops](#), 86  
[pipeline.extras.ops.sklearn\\_ops](#), 87  
[pipeline.extras.transformers](#), 89

[pipelinex.flex\\_kedro, 89](#) [105](#)  
[pipelinex.flex\\_kedro.configure, 95](#) [pipelinex.utils, 106](#)  
[pipelinex.flex\\_kedro.context, 89](#)  
[pipelinex.flex\\_kedro.context.context,](#)  
[89](#)  
[pipelinex.flex\\_kedro.context.flexible\\_catalog\\_context,](#)  
[89](#)  
[pipelinex.flex\\_kedro.context.flexible\\_context,](#)  
[89](#)  
[pipelinex.flex\\_kedro.context.flexible\\_parameters\\_context,](#)  
[89](#)  
[pipelinex.flex\\_kedro.context.flexible\\_run\\_context,](#)  
[91](#)  
[pipelinex.flex\\_kedro.context.save\\_pipeline\\_json\\_context,](#)  
[93](#)  
[pipelinex.flex\\_kedro.pipeline, 93](#)  
[pipelinex.flex\\_kedro.pipeline.pipeline,](#)  
[93](#)  
[pipelinex.flex\\_kedro.pipeline.sub\\_pipeline,](#)  
[94](#)  
[pipelinex.hatch\\_dict, 95](#)  
[pipelinex.hatch\\_dict.hatch\\_dict, 95](#)  
[pipelinex.mlflow\\_on\\_kedro, 97](#)  
[pipelinex.mlflow\\_on\\_kedro.datasets, 97](#)  
[pipelinex.mlflow\\_on\\_kedro.datasets.mlflow,](#)  
[97](#)  
[pipelinex.mlflow\\_on\\_kedro.datasets.mlflow.mlflow\\_dataset,](#)  
[97](#)  
[pipelinex.mlflow\\_on\\_kedro.decorators,](#)  
[99](#)  
[pipelinex.mlflow\\_on\\_kedro.decorators.mlflow\\_logger,](#)  
[99](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks, 99](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow,](#)  
[99](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_artifacts\\_logger,](#)  
[99](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_basic\\_logger,](#)  
[100](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_catalog\\_logger,](#)  
[101](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_datasets\\_logger,](#)  
[102](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_env\\_vars\\_logger,](#)  
[103](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_time\\_logger,](#)  
[104](#)  
[pipelinex.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_utils,](#)  
[105](#)  
[pipelinex.mlflow\\_on\\_kedro.transformers,](#)  
[105](#)  
[pipelinex.mlflow\\_on\\_kedro.transformers.mlflow,](#)  
[105](#)  
[pipelinex.mlflow\\_on\\_kedro.transformers.mlflow.mlflow\\_io\\_time\\_logger,](#)

## INDEX

## Symbols

# Symbols

```
__init__ () (pipelinex.extras.datasets.opencv.images_dataset.OpenCVImagesLocalDataSet
method), 47
__init__ () (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet
method), 48
__init__ () (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSVLocalDataSet
method), 49
__init__ () (pipelinex.extras.datasets.pandas.fixed_width_csv_dataset.FixedWidthCSVDataSet
method), 49
__init__ () (pipelinex.extras.datasets.pandas.histogram.HistogramDataSet
method), 50
__init__ () (pipelinex.extras.datasets.pandas.pandas_cat_matrix.PandasCatMatrixDataSet
method), 50
__init__ () (pipelinex.extras.datasets.pandas.pandas_describe.PandasDescribeDataSet
method), 51
__init__ () (pipelinex.extras.datasets.pandas_profiling.pandas_profiling.PandasProfilingDataSet
method), 51
__init__ () (pipelinex.extras.datasets.pillow.images_dataset.ImagesLocalDataSet
method), 52
__init__ () (pipelinex.extras.datasets.pillow.images_dataset.Np3DArrayDataSet
method), 53
__init__ () (pipelinex.extras.datasets.pillow.images_dataset.Np3DArrayDataSetFromList
method), 53
__init__ () (pipelinex.extras.datasets.requests.api_dataset.APIDataSet
method), 54
__init__ () (pipelinex.extras.datasets.seaborn.seaborn_pairplot.SeabornPairPlotDataSet
method), 56
__init__ () (pipelinex.extras.datasets.torchvision.iterable_images_dataset.IterableImagesDataSet
method), 56
__init__ () (pipelinex.extras.hooks.add_catalog_dict.AddCatalogDictHook
method), 58
__init__ () (pipelinex.extras.hooks.add_transformers.AddTransformersHook
method), 58
__init__ () (pipelinex.extras.ops.allennlp_ops.AllennlpReaderToDict
method), 64
__init__ () (pipelinex.extras.ops.argmaxparse_ops.FeedArgsDict
method), 64
__init__ () (pipelinex.extras.ops.ignite.declaratives.declarative_trainer.NetworkTrainer
method), 61
__init__ () (pipelinex.extras.ops.ignite.handlers.flexible_checkpoint.FlexibleModelCheckpoint
method), 61
__init__ () (pipelinex.extras.ops.ignite.handlers.time_limit.TimeLimit
method), 62
__init__ () (pipelinex.extras.ops.ignite.metrics.cohen_kappa_score.CohenKappaScore
method), 62
__init__ () (pipelinex.extras.ops.ignite.metrics.fbeta_score.FbetaScore
method), 63
__init__ () (pipelinex.extras.ops.ignite.metrics.utils.ClassificationOutput
method), 64
__init__ () (pipelinex.extras.ops.pumpy_ops.ReverseChannel
method), 64
__init__ () (pipelinex.extras.ops.opencv_ops.CvBGR2Gray
method), 64
__init__ () (pipelinex.extras.ops.opencv_ops.CvBGR2HSV
method), 64
__init__ () (pipelinex.extras.ops.opencv_ops.CvDiagonalEdgeFilter2d
method), 65
__init__ () (pipelinex.extras.ops.opencv_ops.CvModuleListMerge
method), 66
__init__ () (pipelinex.extras.ops.opencv_ops.CvScale
method), 66
__init__ () (pipelinex.extras.ops.opencv_ops.CvSobel
method), 66
__init__ () (pipelinex.extras.ops.opencv_ops.CvThreshold
method), 66
__init__ () (pipelinex.extras.ops.pandas_ops.DfAddRowStat
method), 68
__init__ () (pipelinex.extras.ops.pandas_ops.DfAssignColumns
method), 68
__init__ () (pipelinex.extras.ops.pandas_ops.DfBaseMethod
method), 68
__init__ () (pipelinex.extras.ops.pandas_ops.DfBaseTask
method), 68
__init__ () (pipelinex.extras.ops.pandas_ops.DfColApply
method), 68
__init__ () (pipelinex.extras.ops.pandas_ops.DfConcat
method), 69
__init__ () (pipelinex.extras.ops.pandas_ops.DfCondReplace
method), 69
__init__ () (pipelinex.extras.ops.pandas_ops.DfDropFilter
method), 69
__init__ () (pipelinex.extras.ops.pandas_ops.DfDtypesApply
method), 69
__init__ () (pipelinex.extras.ops.pandas_ops.DfDuplicate
method), 69
```

`method), 69`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfEval` `method), 69`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfFocusTransform` `method), 70`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfGetColumnInfo` `method), 70`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfGetDummies` `method), 70`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfMap` `method), 70`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfMerge` `method), 70`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfRelative` `method), 71`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfRename` `method), 71`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfRowApply` `method), 71`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfSample` `method), 71`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfSlice` `method), 72`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfSpatialFeatures` `method), 72`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfStrftime` `method), 72`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfToDatetime` `method), 73`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfToTimedelta` `method), 73`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.DfTotalSeconds` `method), 73`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.NestedDictToDf` `method), 73`  
`__init__()` (`pipelinex.extras.ops.pandas_ops.SrMap` `method), 73`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.ModuleBottleneck2d` `method), 75`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.ModuleConcatSkip` `method), 75`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.ModuleConvWrap` `method), 75`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.ModuleSumSkip` `method), 76`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.Pool1dMixIn` `method), 76`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.Pool2dMixIn` `method), 76`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.Pool3dMixIn` `method), 76`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.StatModule` `method), 77`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.StepBinary` `method), 77`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorClamp` `method), 77`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorClampMax` `method), 78`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorClampMin` `method), 78`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorConstantLinear` `method), 78`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorCumsum` `method), 79`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorForward` `method), 80`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorNearestPad` `method), 83`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorProba` `method), 84`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorSlice` `method), 85`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorSqueeze` `method), 85`  
`__init__()` (`pipelinex.extras.ops.pytorch_ops.TensorUnsqueeze` `method), 85`  
`__init__()` (`pipelinex.extras.ops.shap_ops.ExplainModel` `method), 86`  
`__init__()` (`pipelinex.extras.ops.shap_ops.Scale` `method), 86`  
`__init__()` (`pipelinex.extras.ops.sklearn_ops.DfBaseTransformer` `method), 87`  
`__init__()` (`pipelinex.extras.ops.sklearn_ops.DfTrainTestSplit` `method), 88`  
`__init__()` (`pipelinex.extras.ops.sklearn_ops.ZeroToZeroTransformer` `method), 88`  
`__init__()` (`pipelinex.flex_kedro.context.flexible_parameters_context.Fl` `method), 89`  
`__init__()` (`pipelinex.flex_kedro.pipeline.pipeline.FlexiblePipeline` `method), 93`  
`__init__()` (`pipelinex.flex_kedro.pipeline.sub_pipeline.SubPipeline` `method), 94`  
`__init__()` (`pipelinex.hatch_dict.hatch_dict.Construct` `method), 95`  
`__init__()` (`pipelinex.hatch_dict.hatch_dict.HatchDict` `method), 95`  
`__init__()` (`pipelinex.hatch_dict.hatch_dict.Method` `method), 96`  
`__init__()` (`pipelinex.hatch_dict.hatch_dict.ToPipeline` `method), 96`  
`__init__()` (`pipelinex.mlflow_on_kedro.datasets.mlflow.mlflow_dataset.` `method), 97`  
`__init__()` (`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_artifacts_L` `method), 99`  
`__init__()` (`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logs` `method), 100`  
`__init__()` (`pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_lo`





core (pipelinex.extras.ops.pytorch_ops.TensorConv2d attribute), 79	CvModuleStack (class in pipelinex.extras.ops.opencv_ops), 66
core (pipelinex.extras.ops.pytorch_ops.TensorConv3d attribute), 79	CvModuleSum (class in pipelinex.extras.ops.opencv_ops), 66
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool1d attribute), 82	CvResize (class in pipelinex.extras.ops.opencv_ops), 66
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool2d attribute), 83	CvScale (class in pipelinex.extras.ops.opencv_ops), 66
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool3d attribute), 83	CvSobel (class in pipelinex.extras.ops.opencv_ops), 66
CrossEntropyLoss2d (class in pipelinex.extras.ops.pytorch_ops), 74	CvThreshold (class in pipelinex.extras.ops.opencv_ops), 66
CSVLocalDataSet (class in pipelinex.extras.datasets.pandas.csv_local), 47	<b>D</b>
CvBGR2Gray (class in pipelinex.extras.ops.opencv_ops), 64	DEFAULT_LOAD_ARGS (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet attribute), 48
CvBGR2HSV (class in pipelinex.extras.ops.opencv_ops), 64	DEFAULT_LOAD_ARGS (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSV attribute), 49
CvBilateralFilter (class in pipelinex.extras.ops.opencv_ops), 65	DEFAULT_PREVIEW_ARGS (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSV attribute), 49
CvBlur (class in pipelinex.extras.ops.opencv_ops), 65	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet attribute), 48
CvBoxFilter (class in pipelinex.extras.ops.opencv_ops), 65	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.pandas_profiling.pandas_profiling.PandasProfiling attribute), 51
CvCanny (class in pipelinex.extras.ops.opencv_ops), 65	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.seaborn.seaborn_pairplot.SeabornPairPlot attribute), 56
CvCvtColor (class in pipelinex.extras.ops.opencv_ops), 65	degree_matrix() (in module pipelinex.extras.ops.pandas_ops), 73
CvDiagonalEdgeFilter2d (class in pipelinex.extras.ops.opencv_ops), 65	df_set_index() (in module pipelinex.extras.decorators.pandas_decorators), 58
CvDictToDict (class in pipelinex.extras.ops.opencv_ops), 65	DfAddRowStat (class in pipelinex.extras.ops.pandas_ops), 68
CvDilate (class in pipelinex.extras.ops.opencv_ops), 65	DfAgg (class in pipelinex.extras.ops.pandas_ops), 68
CvEqualizeHist (class in pipelinex.extras.ops.opencv_ops), 65	DfAggregate (class in pipelinex.extras.ops.pandas_ops), 68
CvErode (class in pipelinex.extras.ops.opencv_ops), 65	DfApply (class in pipelinex.extras.ops.pandas_ops), 68
CvFilter2d (class in pipelinex.extras.ops.opencv_ops), 65	DfApplymap (class in pipelinex.extras.ops.pandas_ops), 68
CvGaussianBlur (class in pipelinex.extras.ops.opencv_ops), 65	DfAssignColumns (class in pipelinex.extras.ops.pandas_ops), 68
CvHoughLinesP (class in pipelinex.extras.ops.opencv_ops), 65	DfBaseMethod (class in pipelinex.extras.ops.pandas_ops), 68
CvLine (class in pipelinex.extras.ops.opencv_ops), 66	DfBaseTask (class in pipelinex.extras.ops.pandas_ops), 68
CvMedianBlur (class in pipelinex.extras.ops.opencv_ops), 66	DfBaseTransformer (class in pipelinex.extras.ops.sklearn_ops), 87
CvModuleConcat (class in pipelinex.extras.ops.opencv_ops), 66	DfColApply (class in pipelinex.extras.ops.pandas_ops), 68
CvModuleL1 (class in pipelinex.extras.ops.opencv_ops), 66	
CvModuleL2 (class in pipelinex.extras.ops.opencv_ops), 66	
CvModuleListMerge (class in pipelinex.extras.ops.opencv_ops), 66	
CvModuleMean (class in pipelinex.extras.ops.opencv_ops), 66	

DfConcat (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfRowApply (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71
DfCondReplace (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSample (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71
DfDrop (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSelectDtypes (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71
DfDropDuplicates (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSelectDtypesCols (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfDropFilter (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSetIndex (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfDtypesApply (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfShift (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfDuplicate (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSlice (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfEval (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSortValues (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfEwm (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfSpatialFeatures (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfExpanding (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 69	DfStandardScaler (class in <i>pipelinex.extras.ops.sklearn_ops</i> ), 87
DfFillna (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfStrftime (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfFilter (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfTail (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfFilterCols (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfToDatetime (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 72
DfFocusTransform (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfTotalSeconds (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 73
DfGetColIndexes (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfToTimedelta (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 73
DfGetCols (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfTrainTestSplit (class in <i>pipelinex.extras.ops.sklearn_ops</i> ), 88
DfGetDummies (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	DfTransform (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 73
DfGroupby (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	dict_io() (in module <i>pipelinex.utils</i> ), 106
DfHead (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	dict_of_list_to_list_of_dict() (in module <i>pipelinex.utils</i> ), 107
DfMap (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	dict_string_val_prefix() (in module <i>pipelinex.extras.datasets.pandas.efficient_csv_local</i> ), 49
DfMerge (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 70	dict_val_replace_except() (in module <i>pipelinex.extras.datasets.pandas.efficient_csv_local</i> ), 49
DfMinMaxScaler (class in <i>pipelinex.extras.ops.sklearn_ops</i> ), 87	DictToDict (class in <i>pipelinex.utils</i> ), 106
DfNgroup (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	distance_matrix() (in module <i>pipelinex.extras.ops.pandas_ops</i> ), 73
DfPipe (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	distance_to_affinity() (in module <i>pipelinex.extras.ops.pandas_ops</i> ), 73
DfQuantileTransformer (class in <i>pipelinex.extras.ops.sklearn_ops</i> ), 87	dot_flatten() (in module <i>pipelinex.hatch_dict.hatch_dict</i> ), 96
DfQuery (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	dump_dict() (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger</i> ), 104
DfRelative (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	<b>E</b>
DfRename (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	EfficientCSVLocalDataSet (class in <i>pipelinex.extras.datasets.pandas.efficient_csv_local</i> ),
DfResample (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	
DfResetIndex (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	
DfRolling (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 71	

49  
eigen() (in module *pipelinex.extras.ops.pandas\_ops*),  
73  
element\_wise\_average() (in module  
*pipelinex.extras.ops.pytorch\_ops*), 86  
element\_wise\_prod() (in module  
*pipelinex.extras.ops.pytorch\_ops*), 86  
element\_wise\_sum() (in module  
*pipelinex.extras.ops.pytorch\_ops*), 86  
env\_vars\_to\_dict() (in module  
*pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_loggable*), 65  
103  
EstimatorTransformer (class in  
*pipelinex.extras.ops.sklearn\_ops*), 88  
expand\_repeat() (in module  
*pipelinex.extras.ops.opencv\_ops*), 67  
ExplainModel (class in  
*pipelinex.extras.ops.shap\_ops*), 86  
extract\_from\_df() (in module  
*pipelinex.extras.ops.sklearn\_ops*), 88

**F**

FbetaScore (class in  
*pipelinex.extras.ops.ignite.metrics.fbeta\_score*),  
63  
feed() (in module *pipelinex.hatch\_dict.hatch\_dict*), 96  
FeedArgsDict (class in  
*pipelinex.extras.ops argparse\_ops*), 64  
fit() (*pipelinex.extras.ops.sklearn\_ops.DfBaseTransformer*  
method), 87  
fit\_to\_l() (in module  
*pipelinex.extras.ops.opencv\_ops*), 67  
fit\_to\_uint8() (in module  
*pipelinex.extras.ops.opencv\_ops*), 67  
fit\_transform() (*pipelinex.extras.ops.sklearn\_ops.DfBaseTransformer*  
method), 87  
fit\_transform() (*pipelinex.extras.ops.sklearn\_ops.ZeroToZeroThresholdTransformer*  
method), 88  
fix\_width() (in module  
*pipelinex.extras.datasets.pandas.fixed\_width\_csv\_dataset*),  
50  
FixedWidthCSVDataset (class in  
*pipelinex.extras.datasets.pandas.fixed\_width\_csv\_dataset*),  
49  
FlexibleCatalogContext (class in  
*pipelinex.flex\_kedro.context.flexible\_catalog\_context*),  
89  
FlexibleContext (class in  
*pipelinex.flex\_kedro.context.flexible\_context*),  
89  
FlexibleModelCheckpoint (class in  
*pipelinex.extras.ops.ignite.handlers.flexible\_checkpoint*),  
61  
FlexibleParametersContext (class in  
*pipelinex.flex\_kedro.context.flexible\_parameters\_context*),  
89  
FlexiblePipeline (class in  
*pipelinex.flex\_kedro.pipeline.pipeline*), 93  
FlexibleRunContext (class in  
*pipelinex.flex\_kedro.context.flexible\_run\_context*),  
91  
fn (*pipelinex.extras.ops.opencv\_ops.CvBGR2Gray* at-  
tribute), 64  
fn (*pipelinex.extras.ops.opencv\_ops.CvBGR2HSV* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvBilateralFilter*  
attribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvBlur* attribute),  
65  
fn (*pipelinex.extras.ops.opencv\_ops.CvBoxFilter* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvCanny* attribute),  
65  
fn (*pipelinex.extras.ops.opencv\_ops.CvCvtColor* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvDilate* attribute),  
65  
fn (*pipelinex.extras.ops.opencv\_ops.CvEqualizeHist* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvErode* attribute),  
65  
fn (*pipelinex.extras.ops.opencv\_ops.CvFilter2d* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvGaussianBlur* at-  
tribute), 65  
fn (*pipelinex.extras.ops.opencv\_ops.CvHoughLinesP* at-  
tribute), 66  
fn (*pipelinex.extras.ops.opencv\_ops.CvLine* attribute),  
66  
fn (*pipelinex.extras.ops.opencv\_ops.CvMedianBlur* at-  
tribute), 66  
fn (*pipelinex.extras.ops.opencv\_ops.CvResize* attribute),  
66  
fn (*pipelinex.extras.ops.opencv\_ops.CvSobel* attribute),  
66  
fn (*pipelinex.extras.ops.opencv\_ops.CvThreshold*  
attribute), 66  
fn (*pipelinex.extras.ops.opencv\_ops.NpAbs* attribute), 67  
fn (*pipelinex.extras.ops.opencv\_ops.NpConcat* at-  
tribute), 67  
fn (*pipelinex.extras.ops.opencv\_ops.NpFullLike* at-  
tribute), 67  
fn (*pipelinex.extras.ops.opencv\_ops.NpMean* attribute),  
67  
fn (*pipelinex.extras.ops.opencv\_ops.NpOnesLike* at-  
tribute), 67  
fn (*pipelinex.extras.ops.opencv\_ops.NpSqrt* attribute),  
67

fn (pipelinex.extras.ops.opencv\_ops.NpSquare attribute), 67  
 fn (pipelinex.extras.ops.opencv\_ops.NpStack attribute), 67  
 fn (pipelinex.extras.ops.opencv\_ops.NpSum attribute), 67  
 fn (pipelinex.extras.ops.opencv\_ops.NpZerosLike attribute), 67  
 fn (pipelinex.extras.ops.skimage\_ops.SkimageMarkBoundaries attribute), 86  
 fn (pipelinex.extras.ops.skimage\_ops.SkimageSegmentationFelzenszwaldf attribute), 84  
 fn (pipelinex.extras.ops.skimage\_ops.SkimageSegmentationQuickshift attribute), 84  
 fn (pipelinex.extras.ops.skimage\_ops.SkimageSegmentationSlic attribute), 84  
 fn (pipelinex.extras.ops.skimage\_ops.SkimageSegmentationWatershed attribute), 85  
 fn (pipelinex.utils.DictToDict attribute), 106  
 forward () (pipelinex.extras.ops.pytorch\_ops.CrossEntropyLoss), 74  
 forward () (pipelinex.extras.ops.pytorch\_ops.ModuleAvg), 74  
 forward () (pipelinex.extras.ops.pytorch\_ops.ModuleConcat), 75  
 forward () (pipelinex.extras.ops.pytorch\_ops.ModuleListMerge), 75  
 forward () (pipelinex.extras.ops.pytorch\_ops.ModuleProd), 75  
 forward () (pipelinex.extras.ops.pytorch\_ops.ModuleSum), 76  
 forward () (pipelinex.extras.ops.pytorch\_ops.NLLoss), 76  
 forward () (pipelinex.extras.ops.pytorch\_ops.StepBinary), 77  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorClamp), 78  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorClampMax), 78  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorClampMin), 78  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorConstantLinear), 78  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorCumsum), 79  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorExp), 79  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorFlatten), 80  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorForwardHistogramDataSet), 80  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorIdentity), 82  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorLog), 82  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorMax), 82  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorMean), 83  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorMin), 83  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorNearestPad), 83  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorProba), 84  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorRange), 84  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorSkip), 84  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorSlice), 84  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorSqueeze), 85  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorSum), 85  
 forward () (pipelinex.extras.ops.pytorch\_ops.TensorUnsqueeze), 85  
 Merge (class in pipelinex.hatch\_dict.hatch\_dict), 95  
 get () (pipelinex.hatch\_dict.hatch\_dict.HatchDict), 95  
 get\_kedro\_runner () (in module pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_catalog\_logger), 102  
 get\_nv\_info () (in module pipelinex.extras.decorators.nvml\_profiler), 57  
 get\_params () (pipelinex.hatch\_dict.hatch\_dict.HatchDict), 96  
 get\_timestamp () (in module pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_basic\_logger), 101  
 get\_timestamp\_int () (in module pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_basic\_logger), 101  
 get\_timestamps () (in module pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_basic\_logger), 101  
 HatchDict (class in pipelinex.hatch\_dict.hatch\_dict), 95  
 HistogramDataSet (class in pipelinex.extras.datasets.pandas.histogram), 50



**I**  
 ignore\_index (pipelinex.extras.ops.pytorch\_ops.CrossEntropyLoss (in module pipelinex.extras.decorators.decorators), 57  
 attribute), 74  
 ignore\_index (pipelinex.extras.ops.pytorch\_ops.NLLoss (in module pipelinex.extras.decorators.decorators), 57  
 attribute), 76  
 ImagesLocalDataSet (class in pipelinex.extras.decorators.memory\_profiler),  
 pipelinex.extras.datasets.pillow.images\_dataset), 57  
 52  
 import\_modules () (in module method (pipelinex.extras.ops.pandas\_ops.DfAgg at-  
 pipelinex.flex\_kedro.context.flexible\_parameters\_context), tribute), 68  
 90  
 method (pipelinex.extras.ops.pandas\_ops.DfAggregate  
 attribute), 68  
 ItemGetter (class in pipelinex.utils), 106  
 items () (pipelinex.hatch\_dict.hatch\_dict.HatchDict method (pipelinex.extras.ops.pandas\_ops.DfApply at-  
 method), 96 tribute), 68  
 IterableImagesDataSet (class in method (pipelinex.extras.ops.pandas\_ops.DfApplymap  
 pipelinex.extras.datasets.torchvision.iterable\_images\_dataset), 68  
 56  
 method (pipelinex.extras.ops.pandas\_ops.DfBaseMethod  
 attribute), 68  
**K**  
 method (pipelinex.extras.ops.pandas\_ops.DfBaseTask  
 attribute), 68  
 keys () (pipelinex.hatch\_dict.hatch\_dict.HatchDict method (pipelinex.extras.ops.pandas\_ops.DfDrop at-  
 method), 96 tribute), 69  
**L**  
 method (pipelinex.extras.ops.pandas\_ops.DfDropDuplicates  
 attribute), 69  
 label\_smoothing (pipelinex.extras.ops.pytorch\_ops.CrossEntropyLoss (in module pipelinex.extras.decorators.decorators), 57  
 attribute), 74  
 laplacian\_eigen () (in module method (pipelinex.extras.ops.pandas\_ops.DfEwm at-  
 pipelinex.extras.ops.pandas\_ops), 73 tribute), 69  
 laplacian\_matrix () (in module method (pipelinex.extras.ops.pandas\_ops.DfExpanding  
 pipelinex.extras.ops.pandas\_ops), 73 attribute), 69  
 list\_of\_dict\_to\_dict\_of\_list () (in module method (pipelinex.extras.ops.pandas\_ops.DfFillna at-  
 pipelinex.utils), 107 tribute), 70  
 load () (pipelinex.mlflow\_on\_kedro.transformers.mlflow.mlflow\_io\_time\_logger.MLflowIOTimeLoggerTransformer  
 method), 106 method (pipelinex.extras.ops.pandas\_ops.DfHead at-  
 attribute), 70  
 load\_context () (in module method (pipelinex.extras.ops.pandas\_ops.DfNgroup at-  
 pipelinex.flex\_kedro.configure), 95 tribute), 71  
 load\_dict () (in module method (pipelinex.extras.ops.pandas\_ops.DfPipe  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_time\_logger), (pipelinex.extras.ops.pandas\_ops.DfPipe  
 104 attribute), 71  
 load\_image () (in module method (pipelinex.extras.ops.pandas\_ops.DfQuery at-  
 pipelinex.extras.datasets.pillow.images\_dataset), tribute), 71  
 53  
 load\_obj () (in module method (pipelinex.extras.ops.pandas\_ops.DfResample  
 pipelinex.hatch\_dict.hatch\_dict), 96 attribute), 71  
 load\_time\_dict () (pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_time\_logger.MLflowTimeLoggerHook  
 method), 104 method (pipelinex.extras.ops.pandas\_ops.DfResetIndex  
 attribute), 71  
 log\_df\_summary () (in module method (pipelinex.extras.ops.pandas\_ops.DfRolling at-  
 pipelinex.extras.decorators.pandas\_decorators), method (pipelinex.extras.ops.pandas\_ops.DfSelectDtypes  
 58 attribute), 72  
 log\_metric\_env\_vars () (in module method (pipelinex.extras.ops.pandas\_ops.DfSetIndex  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_env\_vars\_logger), (pipelinex.extras.ops.pandas\_ops.DfSetIndex  
 103 attribute), 72  
 method (pipelinex.extras.ops.pandas\_ops.DfShift  
 attribute), 72  
 log\_param\_env\_vars () (in module method (pipelinex.extras.ops.pandas\_ops.DfSortValues  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_env\_vars\_logger), (pipelinex.extras.ops.pandas\_ops.DfSortValues  
 103 attribute), 72

method ( <i>pipelinex.extras.ops.pandas_ops.DfTail</i> attribute), 72	89
method ( <i>pipelinex.extras.ops.pandas_ops.DfTransform</i> attribute), 73	MLflowIOTimeLoggerTransformer (class in <i>pipelinex.mlflow_on_kedro.transformers.mlflow.mlflow_io_time_logger</i> ), 105
method ( <i>pipelinex.hatch_dict.hatch_dict.Get</i> attribute), 95	MLflowOutputsLoggerHook (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger</i> ), 102
method ( <i>pipelinex.hatch_dict.hatch_dict.Method</i> attribute), 96	MLflowTimeLoggerHook (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_time_logger</i> ), 104
<i>mix_up()</i> (in module <i>pipelinex.extras.ops.opencv_ops</i> ), 67	
<i>mlflow_end_run()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils</i> ), 105	<i>pipelinex</i> , 46
<i>mlflow_log_artifacts()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils</i> ), 105	<i>pipelinex.extras</i> , 46
<i>mlflow_log_dataset()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger</i> ), 102	<i>pipelinex.extras.datasets</i> , 46
<i>mlflow_log_metrics()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils</i> ), 105	<i>pipelinex.extras.datasets.core</i> , 57
<i>mlflow_log_params()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils</i> ), 105	<i>pipelinex.extras.datasets.httpx</i> , 46
<i>mlflow_log_time()</i> (in module <i>pipelinex.mlflow_on_kedro.decorators.mlflow_logger</i> ), 99	<i>pipelinex.extras.datasets.httpx.async_api_datasets</i> , 46
<i>mlflow_log_values()</i> (in module <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_utils</i> ), 105	<i>pipelinex.extras.datasets.opencv</i> , 47
<i>MLflowArtifactsLoggerHook</i> (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_artifacts_logger</i> ), 99	<i>pipelinex.extras.datasets.opencv.images_dataset</i> , 47
<i>MLflowBasicLoggerHook</i> (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_basic_logger</i> ), 100	<i>pipelinex.extras.datasets.pandas</i> , 47
<i>MLflowCatalogLoggerHook</i> (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_catalog_logger</i> ), 101	<i>pipelinex.extras.datasets.pandas.csv_local</i> , 47
<i>MLflowDataSet</i> (class in <i>pipelinex.mlflow_on_kedro.datasets.mlflow.mlflow_dataset</i> ), 97	<i>pipelinex.extras.datasets.pandas.efficient_csv</i> , 49
<i>MLflowDataSetsLoggerHook</i> (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_datasets_logger</i> ), 102	<i>pipelinex.extras.datasets.pandas.fixed_width_csv</i> , 49
<i>MLflowEnvVarsLoggerHook</i> (class in <i>pipelinex.mlflow_on_kedro.hooks.mlflow.mlflow_env_vars_logger</i> ), 103	<i>pipelinex.extras.datasets.pandas.histogram</i> , 50
<i>MLflowFlexibleContext</i> (class in <i>pipelinex.flex_kedro.context.flexible_context</i> ), 56	<i>pipelinex.extras.datasets.pandas.pandas_cat_mat</i> , 51
	<i>pipelinex.extras.datasets.pandas.pandas_describe</i> , 51
	<i>pipelinex.extras.datasets.pandas_profiling</i> , 51
	<i>pipelinex.extras.datasets.pandas_profiling.pandas_profiling</i> , 51
	<i>pipelinex.extras.datasets.pillow</i> , 52
	<i>pipelinex.extras.datasets.pillow.images_dataset</i> , 52
	<i>pipelinex.extras.datasets.requests</i> , 53
	<i>pipelinex.extras.datasets.requests.api_dataset</i> , 53
	<i>pipelinex.extras.datasets.seaborn</i> , 56
	<i>pipelinex.extras.datasets.seaborn.seaborn_pairplot</i> , 56
	<i>pipelinex.extras.datasets.torchvision</i> , 56
	<i>pipelinex.extras.datasets.torchvision.iterable_dataset</i> , 56
	<i>pipelinex.extras.decorators</i> , 57

[pipelineX.extras.decorators.decorators](#), [pipelineX.flex\\_kedro.context.flexible\\_parameter](#)  
[57](#) [89](#)  
[pipelineX.extras.decorators.memory\\_profile](#), [pipelineX.flex\\_kedro.context.flexible\\_run\\_conte](#)  
[57](#) [91](#)  
[pipelineX.extras.decorators.nvml\\_profile](#), [pipelineX.flex\\_kedro.context.save\\_pipeline\\_json](#)  
[57](#) [93](#)  
[pipelineX.extras.decorators.pandas\\_decorator](#), [pipelineX.flex\\_kedro.pipeline](#), [93](#)  
[58](#) [pipelineX.flex\\_kedro.pipeline.pipeline](#),  
[pipelineX.extras.hooks](#), [58](#) [93](#)  
[pipelineX.extras.hooks.add\\_catalog\\_dict](#), [pipelineX.flex\\_kedro.pipeline.sub\\_pipeline](#),  
[58](#) [94](#)  
[pipelineX.extras.hooks.add\\_transformers](#), [pipelineX.hatch\\_dict](#), [95](#)  
[58](#) [pipelineX.hatch\\_dict.hatch\\_dict](#), [95](#)  
[pipelineX.extras.ops](#), [59](#) [pipelineX.mlflow\\_on\\_kedro](#), [97](#)  
[pipelineX.extras.ops.allennlp\\_ops](#), [pipelineX.mlflow\\_on\\_kedro.datasets](#),  
[64](#) [97](#)  
[pipelineX.extras.ops.argparse\\_ops](#), [pipelineX.mlflow\\_on\\_kedro.datasets.mlflow](#),  
[64](#) [97](#)  
[pipelineX.extras.ops.ignite](#), [59](#) [pipelineX.mlflow\\_on\\_kedro.datasets.mlflow.mlflow](#)  
[pipelineX.extras.ops.ignite.declaratives](#), [97](#)  
[59](#) [pipelineX.mlflow\\_on\\_kedro.decorators](#),  
[pipelineX.extras.ops.ignite.declaratives.declarative\\_trainer](#),  
[59](#) [99](#)  
[pipelineX.extras.ops.ignite.declaratives.declarative\\_trainer](#),  
[59](#) [pipelineX.mlflow\\_on\\_kedro.decorators.mlflow\\_log](#)  
[pipelineX.extras.ops.ignite.handlers](#), [99](#)  
[61](#) [pipelineX.mlflow\\_on\\_kedro.hooks](#), [99](#)  
[pipelineX.extras.ops.ignite.handlers.flexible\\_checkpointing](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow](#),  
[61](#) [99](#)  
[pipelineX.extras.ops.ignite.handlers.time\\_pipeline](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_a](#)  
[62](#) [99](#)  
[pipelineX.extras.ops.ignite.metrics](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_b](#)  
[62](#) [100](#)  
[pipelineX.extras.ops.ignite.metrics.coherence](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_c](#)  
[62](#) [101](#)  
[pipelineX.extras.ops.ignite.metrics.fbeta](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_d](#)  
[63](#) [102](#)  
[pipelineX.extras.ops.ignite.metrics.util](#), [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_e](#)  
[64](#) [103](#)  
[pipelineX.extras.ops.numpy\\_ops](#), [64](#) [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_t](#)  
[pipelineX.extras.ops.opencv\\_ops](#), [64](#) [104](#)  
[pipelineX.extras.ops.pandas\\_ops](#), [68](#) [pipelineX.mlflow\\_on\\_kedro.hooks.mlflow.mlflow\\_u](#)  
[pipelineX.extras.ops.pytorch\\_ops](#), [74](#) [105](#)  
[pipelineX.extras.ops.shap\\_ops](#), [86](#) [pipelineX.mlflow\\_on\\_kedro.transformers](#),  
[pipelineX.extras.ops.skimage\\_ops](#), [86](#) [105](#)  
[pipelineX.extras.ops.sklearn\\_ops](#), [87](#) [pipelineX.mlflow\\_on\\_kedro.transformers.mlflow](#),  
[pipelineX.extras.transformers](#), [89](#) [105](#)  
[pipelineX.flex\\_kedro](#), [89](#) [pipelineX.mlflow\\_on\\_kedro.transformers.mlflow.m](#)  
[pipelineX.flex\\_kedro.configure](#), [95](#) [105](#)  
[pipelineX.flex\\_kedro.context](#), [89](#) [pipelineX.utils](#), [106](#)  
[pipelineX.flex\\_kedro.context.context.module](#) ([pipelineX.extras.ops.opencv\\_ops.CvDictToDict](#)  
[89](#) [attribute](#)), [65](#)  
[pipelineX.flex\\_kedro.context.flexible.module](#) ([pipelineX.extras.ops.opencv\\_ops.NpDictToDict](#)  
[89](#) [attribute](#)), [67](#)  
[pipelineX.flex\\_kedro.context.flexible.module](#) ([pipelineX.extras.ops.skimage\\_ops.SkimageSegmentationDictToDi](#)  
[89](#) [attribute](#)), [86](#)



module (*pipelinex.utils.DictToDict attribute*), 106  
 ModuleAvg (*class in pipelinex.extras.ops.pytorch\_ops*), 74  
 ModuleBottleneck2d (*class in pipelinex.extras.ops.pytorch\_ops*), 74  
 ModuleConcat (*class in pipelinex.extras.ops.pytorch\_ops*), 75  
 ModuleConcatSkip (*class in pipelinex.extras.ops.pytorch\_ops*), 75  
 ModuleConvWrap (*class in pipelinex.extras.ops.pytorch\_ops*), 75  
 ModuleListMerge (*class in pipelinex.extras.ops.pytorch\_ops*), 75  
 ModuleProd (*class in pipelinex.extras.ops.pytorch\_ops*), 75  
 ModuleSum (*class in pipelinex.extras.ops.pytorch\_ops*), 76  
 ModuleSumSkip (*class in pipelinex.extras.ops.pytorch\_ops*), 76  
**N**  
 namespace () (*in module pipelinex.extras.ops argparse\_ops*), 64  
 nested\_dict\_to\_df () (*in module pipelinex.extras.ops pandas\_ops*), 74  
 NestedDictToDf (*class in pipelinex.extras.ops pandas\_ops*), 73  
 NetworkTrain (*class in pipelinex.extras.ops ignite.declaratives.declarative\_trainer*), 59  
 nl\_loss () (*in module pipelinex.extras.ops.pytorch\_ops*), 86  
 NLLoss (*class in pipelinex.extras.ops.pytorch\_ops*), 76  
 Np3DArrDataset (*class in pipelinex.extras.datasets.pillow.images\_dataset*), 53  
 Np3DArrDatasetFromList (*class in pipelinex.extras.datasets.pillow.images\_dataset*), 53  
 NpAbs (*class in pipelinex.extras.ops.opencv\_ops*), 66  
 NpConcat (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpDictToDict (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpFullLike (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpMean (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpOnesLike (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpSqrt (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpSquare (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpStack (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpSum (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 NpZerosLike (*class in pipelinex.extras.ops.opencv\_ops*), 67  
 nvml\_profile () (*in module pipelinex.extras.decorators.nvml\_profiler*), 57  
**O**  
 OnlyMissingOptionContext (*class in pipelinex.flex\_kedro.context.flexible\_run\_context*), 91  
 OpenCVImagesLocalDataSet (*class in pipelinex.extras.datasets.opencv.images\_dataset*), 47  
 overlay () (*in module pipelinex.extras.ops.opencv\_ops*), 67  
**P**  
 PandasCatMatrixDataSet (*class in pipelinex.extras.datasets.pandas.pandas\_cat\_matrix*), 50  
 PandasDescribeDataSet (*class in pipelinex.extras.datasets.pandas.pandas\_describe*), 51  
 PandasProfilingDataSet (*class in pipelinex.extras.datasets.pandas\_profiling.pandas\_profiling*), 51  
 params () (*pipelinex.flex\_kedro.context.flexible\_parameters\_context.FlexibleParametersContext property*), 90  
 pass\_through () (*in module pipelinex.hatch\_dict.hatch\_dict*), 96  
 pipelinex  
   module, 46  
   pipelinex.extras  
     module, 46  
     pipelinex.extras.datasets  
       module, 46  
       pipelinex.extras.datasets.core  
         module, 57  
       pipelinex.extras.datasets.httpx  
         module, 46  
         pipelinex.extras.datasets.httpx.async\_api\_dataset  
           module, 46  
         pipelinex.extras.datasets.opencv  
           module, 47  
         pipelinex.extras.datasets.opencv.images\_dataset  
           module, 47  
         pipelinex.extras.datasets.pandas  
           module, 47  
         pipelinex.extras.datasets.pandas.csv\_local  
           module, 47  
         pipelinex.extras.datasets.pandas.efficient\_csv\_local  
           module, 49

pipelinex.extras.datasets.pandas.fixed_width_pipeline_extras	pipelinex.extras.ops.ignite.declaratives.declarative
module, 49	module, 59
pipelinex.extras.datasets.pandas.histogram_pipeline_extras	pipelinex.extras.ops.ignite.handlers
module, 50	module, 61
pipelinex.extras.datasets.pandas.pandas_pipeline_extras	pipelinex.extras.ops.ignite.handlers.flexible_check
module, 50	module, 61
pipelinex.extras.datasets.pandas.pandas_describer_extras	pipelinex.extras.ops.ignite.handlers.time_limit
module, 51	module, 62
pipelinex.extras.datasets.pandas_profiling_pipeline_extras	pipelinex.extras.ops.ignite.metrics
module, 51	module, 62
pipelinex.extras.datasets.pandas_profiling_pipeline_extras	pipelinex.extras.ops.ignite.metrics.cohen_kappa_score
module, 51	module, 62
pipelinex.extras.datasets.pillow	pipelinex.extras.ops.ignite.metrics.fbeta_score
module, 52	module, 63
pipelinex.extras.datasets.pillow.images_pipeline_extras	pipelinex.extras.ops.ignite.metrics.utils
module, 52	module, 64
pipelinex.extras.datasets.requests	pipelinex.extras.ops.numpy_ops
module, 53	module, 64
pipelinex.extras.datasets.requests.api_dispatcher_extras	pipelinex.extras.ops.opencv_ops
module, 53	module, 64
pipelinex.extras.datasets.seaborn	pipelinex.extras.ops.pandas_ops
module, 56	module, 68
pipelinex.extras.datasets.seaborn.seaborn_pipeline_extras	pipelinex.extras.ops.pytorch_ops
module, 56	module, 74
pipelinex.extras.datasets.torchvision	pipelinex.extras.ops.shap_ops
module, 56	module, 86
pipelinex.extras.datasets.torchvision.itertools_image_dataset	pipelinex.extras.ops.skimage_ops
module, 56	module, 86
pipelinex.extras.decorators	pipelinex.extras.ops.sklearn_ops
module, 57	module, 87
pipelinex.extras.decorators.decorators	pipelinex.extras.transformers
module, 57	module, 89
pipelinex.extras.decorators.memory_profiler_extras	pipelinex.flex_kedro
module, 57	module, 89
pipelinex.extras.decorators.nvml_profiler_extras	pipelinex.flex_kedro.configure
module, 57	module, 95
pipelinex.extras.decorators.pandas_decorator_extras	pipelinex.flex_kedro.context
module, 58	module, 89
pipelinex.extras.hooks	pipelinex.flex_kedro.context.context
module, 58	module, 89
pipelinex.extras.hooks.add_catalog_dict_extras	pipelinex.flex_kedro.context.flexible_catalog_context
module, 58	module, 89
pipelinex.extras.hooks.add_transformers_extras	pipelinex.flex_kedro.context.flexible_context
module, 58	module, 89
pipelinex.extras.ops	pipelinex.flex_kedro.context.flexible_parameters_context
module, 59	module, 89
pipelinex.extras.ops.allennlp_ops	pipelinex.flex_kedro.context.flexible_run_context
module, 64	module, 91
pipelinex.extras.ops.parseparse_ops	pipelinex.flex_kedro.context.save_pipeline_json_context
module, 64	module, 93
pipelinex.extras.ops.ignite	pipelinex.flex_kedro.pipeline
module, 59	module, 93
pipelinex.extras.ops.ignite.declaratives	pipelinex.flex_kedro.pipeline.pipeline
module, 59	module, 93

pipelinex.flex\_kedro.pipeline.sub\_pipeline  
     module, 94  
 pipelinex.hatch\_dict  
     module, 95  
 pipelinex.hatch\_dict.hatch\_dict  
     module, 95  
 pipelinex.mlflow\_on\_kedro  
     module, 97  
 pipelinex.mlflow\_on\_kedro.datasets  
     module, 97  
 pipelinex.mlflow\_on\_kedro.datasets.mlflow  
     module, 97  
 pipelinex.mlflow\_on\_kedro.datasets.mlflow.mlflow\_dataset  
     module, 97  
 pipelinex.mlflow\_on\_kedro.decorators  
     module, 99  
 pipelinex.mlflow\_on\_kedro.decorators.mlflow\_logger  
     module, 99  
 pipelinex.mlflow\_on\_kedro.hooks  
     module, 99  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow  
     module, 99  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_artifact\_logger  
     module, 99  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_basic\_logger  
     module, 100  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_catalog\_logger  
     module, 101  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_datasets\_logger  
     module, 102  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_env\_vars\_logger  
     module, 103  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_time\_logger  
     module, 104  
 pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_utils  
     module, 105  
 pipelinex.mlflow\_on\_kedro.transformers  
     module, 105  
 pipelinex.mlflow\_on\_kedro.transformers.mlflow  
     module, 105  
 pipelinex.mlflow\_on\_kedro.transformers.mlflow.mlflow\_io\_time\_logger  
     module, 105  
 pipelinex.utils  
     module, 106  
 Pool1dMixIn (class in *pipelinex.extras.ops.pytorch\_ops*), 76  
 Pool2dMixIn (class in *pipelinex.extras.ops.pytorch\_ops*), 76  
 Pool3dMixIn (class in *pipelinex.extras.ops.pytorch\_ops*), 76  
 project\_name (*pipelinex.flex\_kedro.context.flexible\_context.FlexibleContext*  
     attribute), 89  
 project\_version (*pipelinex.flex\_kedro.context.flexible\_context.FlexibleContext*  
     attribute), 89  
 request\_coroutine() (in module  
     *pipelinex.extras.datasets.httpx.async\_api\_dataset*),  
     47  
 requests\_coroutine() (in module  
     *pipelinex.extras.datasets.httpx.async\_api\_dataset*),  
     47  
 reset() (*pipelinex.extras.ops.ignite.metrics.cohen\_kappa\_score.CohenK*  
     method), 62  
 reset() (*pipelinex.extras.ops.ignite.metrics.fbeta\_score.FbetaScore*  
     method), 63  
 reverse\_channel() (in module  
     *pipelinex.extras.ops.numpy\_ops*), 64  
 ReverseChannel (class in  
     *pipelinex.extras.ops.numpy\_ops*), 64  
 row\_vector\_to\_square\_matrix() (in module  
     *pipelinex.extras.ops.pandas\_ops*), 74  
 run() (*pipelinex.flex\_kedro.context.flexible\_parameters\_context.FlexibleP*  
     method), 90  
 run() (*pipelinex.flex\_kedro.context.flexible\_run\_context.OnlyMissingOpti*  
     method), 91  
 run() (*pipelinex.flex\_kedro.context.flexible\_run\_context.StringRunnerOpti*  
     method), 92  
 running\_parallel() (in module  
     *pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_catalog\_logger*),  
     102  
**S**  
 save() (*pipelinex.mlflow\_on\_kedro.transformers.mlflow.mlflow\_io\_time\_l*  
     method), 106  
 SavePipelineToJsonContext (class in  
     *pipelinex.flex\_kedro.context.save\_pipeline\_json\_context*),  
     93  
 Scale (class in *pipelinex.extras.ops.shap\_ops*), 86  
 scale() (in module  
     *pipelinex.extras.datasets.pillow.images\_dataset*),  
     53  
 SeabornPairPlotDataSet (class in  
     *pipelinex.extras.datasets.seaborn.seaborn\_pairplot*),  
     56  
 setup\_conv\_params() (in module  
     *pipelinex.extras.ops.pytorch\_ops*), 86  
 SkimageMarkBoundaries (class in  
     *pipelinex.extras.ops.skimage\_ops*), 86  
 SkimageSegmentationDictToDict (class in  
     *pipelinex.extras.ops.skimage\_ops*), 86  
 SkimageSegmentationFelzenszwalb (class in  
     *pipelinex.extras.ops.skimage\_ops*), 86  
 SkimageSegmentationQuickshift (class in  
     *pipelinex.extras.ops.skimage\_ops*), 87  
 SkimageSegmentationSlic (class in  
     *pipelinex.extras.ops.skimage\_ops*), 87  
 SkimageSegmentationWatershed (class in  
     *pipelinex.extras.ops.skimage\_ops*), 87

SrMap (class in <i>pipelinex.extras.ops.pandas_ops</i> ), 73	TensorGlobalAvgPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80
StatModule (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalMaxPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80
step_binary() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	TensorGlobalMaxPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
StepBinary (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalMaxPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
StringRunnerOptionContext (class in <i>pipelinex.flex_kedro.context.flexible_run_context</i> ), 92	TensorGlobalMinPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
SubPipeline (class in <i>pipelinex.flex_kedro.pipeline.sub_pipeline</i> ), 94	TensorGlobalMinPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
sum_up() (in module <i>pipelinex.extras.ops.opencv_ops</i> ), 67	TensorGlobalMinPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
<b>T</b>	
tensor_max() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	TensorGlobalRangePool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
tensor_min() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	TensorGlobalRangePool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
TensorAvgPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalRangePool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
TensorAvgPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalSumPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
TensorAvgPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalSumPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 81
TensorClamp (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 77	TensorGlobalSumPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 82
TensorClampMax (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 78	TensorIdentity (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 82
TensorClampMin (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 78	TensorLog (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 82
TensorConstantLinear (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 78	TensorMax (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 82
TensorConv1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 79	TensorMaxPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 82
TensorConv2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 79	TensorMaxPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 83
TensorConv3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 79	TensorMaxPool3d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 83
TensorCumsum (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 79	TensorMean (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 83
TensorExp (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 79	TensorMin (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 83
TensorFlatten (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80	TensorNearestPad (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 83
TensorForward (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80	TensorProba (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 84
TensorGlobalAvgPool1d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80	TensorRange (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 84
TensorGlobalAvgPool2d (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 80	TensorSkip (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 84
	TensorSlice (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 84
	TensorSqueeze (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 85

TensorSum (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 85	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool1d</i> attribute), 80
TensorUnsqueeze (class in <i>pipelinex.extras.ops.pytorch_ops</i> ), 85	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool2d</i> attribute), 80
TimeLimit (class in <i>pipelinex.extras.ops.ignite.handlers.time_limit</i> ), 62	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalAvgPool3d</i> attribute), 80
to_array() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool1d</i> attribute), 81
to_channel_first_arr() (in module <i>pipelinex.extras.ops.numpy_ops</i> ), 64	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool2d</i> attribute), 81
to_channel_first_tensor() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMaxPool3d</i> attribute), 81
to_channel_last_arr() (in module <i>pipelinex.extras.ops.numpy_ops</i> ), 64	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool1d</i> attribute), 81
to_channel_last_tensor() (in module <i>pipelinex.extras.ops.pytorch_ops</i> ), 86	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool2d</i> attribute), 81
ToPipeline (class in <i>pipelinex.hatch_dict.hatch_dict</i> ), 96	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalMinPool3d</i> attribute), 81
total_seconds_to_datetime() (in module <i>pipelinex.extras.decorators.pandas_decorators</i> ), 58	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool1d</i> attribute), 81
training ( <i>pipelinex.extras.ops.pytorch_ops.StatModule</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool2d</i> attribute), 81
training ( <i>pipelinex.extras.ops.pytorch_ops.StepBinary</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalRangePool3d</i> attribute), 81
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorAvgPool1d</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool1d</i> attribute), 81
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorAvgPool2d</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool2d</i> attribute), 81
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorAvgPool3d</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorGlobalSumPool3d</i> attribute), 82
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorAvgPool1d</i> attribute), 77	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorIdentity</i> attribute), 82
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorClamp</i> attribute), 78	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorLog</i> attribute), 82
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorClampMax</i> attribute), 78	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMax</i> attribute), 82
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorClampMin</i> attribute), 78	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMaxPool1d</i> attribute), 83
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorConstantLinear</i> attribute), 79	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMaxPool2d</i> attribute), 83
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorConv1d</i> attribute), 79	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMaxPool3d</i> attribute), 83
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorConv2d</i> attribute), 79	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMean</i> attribute), 83
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorConv3d</i> attribute), 79	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorMin</i> attribute), 83
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorCumsum</i> attribute), 79	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorNearestPad</i> attribute), 84
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorExp</i> attribute), 80	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorProba</i> attribute), 84
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorFlatten</i> attribute), 80	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorRange</i> attribute), 84
training ( <i>pipelinex.extras.ops.pytorch_ops.TensorForward</i> attribute), 80	training ( <i>pipelinex.extras.ops.pytorch_ops.TensorSkip</i> attribute), 84

`training` (*pipelinex.extras.ops.pytorch\_ops.TensorSlice*  
*attribute*), 85

`training` (*pipelinex.extras.ops.pytorch\_ops.TensorSqueeze*  
*attribute*), 85

`training` (*pipelinex.extras.ops.pytorch\_ops.TensorSum*  
*attribute*), 85

`training` (*pipelinex.extras.ops.pytorch\_ops.TensorUnsqueeze*  
*attribute*), 86

`transform()` (*pipelinex.extras.ops.sklearn\_ops.DfBaseTransformer*  
*method*), 87

`transform()` (*pipelinex.extras.ops.sklearn\_ops.ZeroToZeroTransformer*  
*method*), 88

`TransformCompose` (*class in pipelinex.utils*), 106

## U

`update()` (*pipelinex.extras.ops.ignite.metrics.cohen\_kappa\_score.CohenKappaScore*  
*method*), 62

`update()` (*pipelinex.extras.ops.ignite.metrics.fbeta\_score.FbetaScore*  
*method*), 63

`update_time_dict()`  
(*pipelinex.mlflow\_on\_kedro.hooks.mlflow.mlflow\_time\_logger.MLflowTimeLoggerHook*  
*method*), 104

## Z

`ZeroToZeroTransformer` (*class in*  
*pipelinex.extras.ops.sklearn\_ops*), 88