
PipelineX

Release 0.4.9

Yusuke Minami

Mar 07, 2021

CONTENTS:

1	PipelineX	1
2	Introduction	3
3	Installation	5
4	Getting Started	7
4.1	Getting Started with Kedro 0.17.x	7
4.2	Example/Demo Projects tested with Kedro 0.16.x	7
5	HatchDict: YAML/JSON enhancement for Python developers	9
5.1	Import-less Python object (class and function)	9
5.2	Anchor-less aliasing (self-lookup)	12
5.3	Python expression	14
6	Kedro as the unified data interface framework	15
6.1	Why the unified data interface framework is needed	15
6.2	Kedro overview	18
7	Kedro context to use YAML files more conveniently	21
7.1	Options available in <code>parameters.yml</code>	21
7.1.1	Define Kedro pipelines	21
7.1.2	Configure Kedro run config using <code>RUN_CONFIG</code> key	22
7.1.3	HatchDict feature	22
7.2	Options available in <code>catalog.yml</code>	23
7.2.1	Enable caching	23
7.2.2	HatchDict feature	23
8	Integration of Kedro with MLflow as Kedro DataSet and Hooks (callbacks)	25
8.1	How to use MLflow from Kedro projects	25
8.2	Comparison with <code>kedro-mlflow</code> package	27
9	Integration of Kedro with the additional Python packages as Kedro DataSets, Hooks (callbacks), and wrappers.	29
9.1	Additional Kedro datasets (data interface sets)	29
9.2	Additional function decorators for benchmarking	30
9.3	Use with PyTorch	31
9.4	Use with PyTorch Ignite	33
9.5	Use with OpenCV	35
9.6	Use with PyTorch Lightning	35
9.7	Use with TensorFlow/Keras	35

10 Build Docker image	37
11 Why and how PipelineX was born	39
12 Author	41
13 Contributors are welcome!	43
13.1 How to contribute	43
13.2 Contributor list	43
14 pipelinux	45
14.1 pipelinux package	46
14.1.1 Subpackages	46
14.1.1.1 pipelinux.extras package	46
14.1.1.1.1 Subpackages	46
14.1.1.1.1.1 pipelinux.extras.datasets package	46
14.1.1.1.1.1.2 Subpackages	46
14.1.1.1.1.1.1.3 pipelinux.extras.datasets.htpx package	46
14.1.1.1.1.1.1.4 Submodules	46
14.1.1.1.1.1.1.5 pipelinux.extras.datasets.htpx.async_api_dataset module	46
14.1.1.1.1.1.1.6 pipelinux.extras.datasets.mlflow package	47
14.1.1.1.1.1.1.7 Submodules	47
14.1.1.1.1.1.1.8 pipelinux.extras.datasets.mlflow.mlflow_dataset module	47
14.1.1.1.1.1.1.9 pipelinux.extras.datasets.opencv package	48
14.1.1.1.1.1.1.10 Submodules	48
14.1.1.1.1.1.1.11 pipelinux.extras.datasets.opencv.images_dataset module	48
14.1.1.1.1.1.1.12 pipelinux.extras.datasets.pandas package	49
14.1.1.1.1.1.1.13 Submodules	49
14.1.1.1.1.1.1.14 pipelinux.extras.datasets.pandas.csv_local module	49
14.1.1.1.1.1.1.15 pipelinux.extras.datasets.pandas.efficient_csv_local module	50
14.1.1.1.1.1.1.16 pipelinux.extras.datasets.pandas.histogram module	50
14.1.1.1.1.1.1.17 pipelinux.extras.datasets.pandas.pandas_cat_matrix module	51
14.1.1.1.1.1.1.18 pipelinux.extras.datasets.pandas.pandas_describe module	51
14.1.1.1.1.1.1.19 pipelinux.extras.datasets.pandas.profilng package	52
14.1.1.1.1.1.1.20 Submodules	52
14.1.1.1.1.1.1.21 pipelinux.extras.datasets.pandas.profilng.pandas_profilng module	52
14.1.1.1.1.1.1.22 pipelinux.extras.datasets.pillow package	52
14.1.1.1.1.1.1.23 Submodules	52
14.1.1.1.1.1.1.24 pipelinux.extras.datasets.pillow.images_dataset module	52
14.1.1.1.1.1.1.25 pipelinux.extras.datasets.requests package	54
14.1.1.1.1.1.1.26 Submodules	54
14.1.1.1.1.1.1.27 pipelinux.extras.datasets.requests.api_dataset module	54
14.1.1.1.1.1.1.28 pipelinux.extras.datasets.seaborn package	56
14.1.1.1.1.1.1.29 Submodules	56
14.1.1.1.1.1.1.30 pipelinux.extras.datasets.seaborn.seaborn_pairplot module	56
14.1.1.1.1.1.1.31 pipelinux.extras.datasets.torchvision package	56
14.1.1.1.1.1.1.32 Submodules	56
14.1.1.1.1.1.1.33 pipelinux.extras.datasets.torchvision.iterable_images_dataset module	56
14.1.1.1.1.1.1.34 Submodules	57
14.1.1.1.1.1.1.35 pipelinux.extras.datasets.core module	57
14.1.1.1.1.1.1.36 pipelinux.extras.decorators package	57
14.1.1.1.1.1.1.37 Submodules	57
14.1.1.1.1.1.1.38 pipelinux.extras.decorators.decorators module	57
14.1.1.1.1.1.1.39 pipelinux.extras.decorators.memory_profiler module	57

14.1.1.1.1.40	pipelinex.extras.decorators.mlflow_logger module	57
14.1.1.1.1.41	pipelinex.extras.decorators.nvml_profiler module	58
14.1.1.1.1.42	pipelinex.extras.decorators.pandas_decorators module	58
14.1.1.1.1.43	pipelinex.extras.hooks package	58
14.1.1.1.1.44	Subpackages	58
14.1.1.1.1.45	pipelinex.extras.hooks.mlflow package	58
14.1.1.1.1.46	Submodules	58
14.1.1.1.1.47	pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger module . .	58
14.1.1.1.1.48	pipelinex.extras.hooks.mlflow.mlflow_basic_logger module	59
14.1.1.1.1.49	pipelinex.extras.hooks.mlflow.mlflow_catalog_logger module . . .	60
14.1.1.1.1.50	pipelinex.extras.hooks.mlflow.mlflow_datasets_logger module . .	61
14.1.1.1.1.51	pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger module . .	61
14.1.1.1.1.52	pipelinex.extras.hooks.mlflow.mlflow_time_logger module	62
14.1.1.1.1.53	pipelinex.extras.hooks.mlflow.mlflow_utils module	63
14.1.1.1.1.54	Submodules	63
14.1.1.1.1.55	pipelinex.extras.hooks.add_catalog_dict module	63
14.1.1.1.1.56	pipelinex.extras.hooks.add_transformers module	64
14.1.1.1.1.57	pipelinex.extras.ops package	65
14.1.1.1.1.58	Subpackages	65
14.1.1.1.1.59	pipelinex.extras.ops.ignite package	65
14.1.1.1.1.60	Subpackages	65
14.1.1.1.1.61	pipelinex.extras.ops.ignite.declaratives package	65
14.1.1.1.1.62	Submodules	65
14.1.1.1.1.63	pipelinex.extras.ops.ignite.declaratives.declarative_trainer module	65
14.1.1.1.1.64	pipelinex.extras.ops.ignite.handlers package	67
14.1.1.1.1.65	Submodules	67
14.1.1.1.1.66	pipelinex.extras.ops.ignite.handlers.flexible_checkpoint module .	67
14.1.1.1.1.67	pipelinex.extras.ops.ignite.handlers.time_limit module	68
14.1.1.1.1.68	pipelinex.extras.ops.ignite.metrics package	68
14.1.1.1.1.69	Submodules	68
14.1.1.1.1.70	pipelinex.extras.ops.ignite.metrics.cohen_kappa_score module . .	68
14.1.1.1.1.71	pipelinex.extras.ops.ignite.metrics.fbeta_score module	69
14.1.1.1.1.72	pipelinex.extras.ops.ignite.metrics.utils module	70
14.1.1.1.1.73	Submodules	70
14.1.1.1.1.74	pipelinex.extras.ops.allennlp_ops module	70
14.1.1.1.1.75	pipelinex.extras.ops.parse_args module	70
14.1.1.1.1.76	pipelinex.extras.ops.numpy_ops module	70
14.1.1.1.1.77	pipelinex.extras.ops.opencv_ops module	70
14.1.1.1.1.78	pipelinex.extras.ops.pandas_ops module	74
14.1.1.1.1.79	pipelinex.extras.ops.pytorch_ops module	80
14.1.1.1.1.80	pipelinex.extras.ops.shap_ops module	92
14.1.1.1.1.81	pipelinex.extras.ops.skimage_ops module	93
14.1.1.1.1.82	pipelinex.extras.ops.sklearn_ops module	93
14.1.1.1.1.83	pipelinex.extras.transformers package	95
14.1.1.1.1.84	Subpackages	95
14.1.1.1.1.85	pipelinex.extras.transformers.mlflow package	95
14.1.1.1.1.86	Submodules	95
14.1.1.1.1.87	pipelinex.extras.transformers.mlflow.mlflow_io_time_logger module	95
14.1.1.2	pipelinex.framework package	96
14.1.1.2.1	Subpackages	96
14.1.1.2.1.1	pipelinex.framework.context package	96
14.1.1.2.1.2	Submodules	96
14.1.1.2.1.3	pipelinex.framework.context.context module	96

14.1.1.2.1.4	pipelinex.framework.context.flexible_catalog_context module . .	96
14.1.1.2.1.5	pipelinex.framework.context.flexible_context module	96
14.1.1.2.1.6	pipelinex.framework.context.flexible_parameters_context module	96
14.1.1.2.1.7	pipelinex.framework.context.flexible_run_context module	98
14.1.1.2.1.8	pipelinex.framework.context.save_pipeline_json_context module .	100
14.1.1.2.2	Submodules	100
14.1.1.2.3	pipelinex.framework.configure module	100
14.1.1.3	pipelinex.hatch_dict package	100
14.1.1.3.1	Submodules	100
14.1.1.3.2	pipelinex.hatch_dict.hatch_dict module	100
14.1.1.4	pipelinex.pipeline package	101
14.1.1.4.1	Submodules	101
14.1.1.4.2	pipelinex.pipeline.pipeline module	101
14.1.1.4.3	pipelinex.pipeline.sub_pipeline module	102
14.1.2	Submodules	103
14.1.3	pipelinex.utils module	103
15	Indices and tables	105
	Python Module Index	107
	Index	109

PIPELINEX

PipelineX: Python package to build production-ready pipelines for experimentation with Kedro, MLflow, and more

INTRODUCTION

PipelineX is a Python package designed to make Machine Learning projects efficient with modular, reusable, and easy-to-use features for experimentation.

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

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

- `HatchDict` option which provides [enhancements for YAML/JSON](#) useful for parameter management summarized as follows.
 - Import-less Python object: Include (nested) Python classes and functions in a YAML/JSON file
 - Anchor-less aliasing: Look up another key in the same YAML/JSON file
 - Python expression in YAML/JSON files
- Kedro context to define Kedro pipelines in a YAML file with more options
- Integration of Kedro with [MLflow](#) as Kedro DataSets and Hooks. Note: You do not need to install MLflow if you do not use.
- Integration of Kedro with the additional Python packages as Kedro DataSets, Hooks, and wrappers.
 - [PyTorch](#)
 - [Ignite](#)
 - [Pandas](#)
 - [OpenCV](#)
 - [Memory Profiler](#)
 - [NVIDIA Management Library](#) Note: You do not need to install Python packages you do not use.

INSTALLATION

- [Option 1] pip install from the PyPI:

```
pip install pipelindex
```

- [Option 2] development install (will be updated as you modify the source code):

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


GETTING STARTED

4.1 Getting Started with Kedro 0.17.x

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: YAML/JSON ENHANCEMENT FOR PYTHON DEVELOPERS

5.1 Import-less Python object (class and function)

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?

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

```

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

```

(continues on next page)

(continued from previous page)

```
# 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)
```

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.

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:

```
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:
```

(continues on next page)

(continued from previous page)

```

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

```

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](#) and `parameters.yml` in `example/demo` projects .

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 (self-lookup)

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
"""

```

(continues on next page)

(continued from previous page)

```
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.

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 pipelinenx 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

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 = """
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}
```

KEDRO AS THE UNIFIED DATA INTERFACE FRAMEWORK

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 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

KEDRO CONTEXT TO USE YAML FILES MORE CONVENIENTLY

Using `pipeline.FlexibleContext`, you can use the YAML files more conveniently

7.1 Options available in `parameters.yml`

7.1.1 Define Kedro pipelines

You can define the inter-task dependency (DAG) for Kedro pipelines in `parameters.yml` using `PIPELINES` key.

- Optionally specify the default Python module (path of .py file) if you want to omit the module name
- Optionally specify the Python function decorator to apply to each node
- Specify inputs, func, and outputs for each node
 - 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`)

```
# parameters.yml

PIPELINES:
  __default__:
    =: pipeline.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
```

7.1.2 Configure Kedro run config using RUN_CONFIG key

- Optionally run nodes in parallel
- Optionally run only missing nodes (skip tasks which have already been run to resume pipeline using the intermediate data files or databases.) Note: You can use Kedro CLI to overwrite these run configs.

```
# 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
```

7.1.3 HatchDict feature

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 Options available in `catalog.yml`

7.2.1 Enable caching

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.)

7.2.2 `HatchDict` feature

You can use `HatchDict` feature in `catalog.yml`.

INTEGRATION OF KEDRO WITH MLFLOW AS KEDRO DATASET AND HOOKS (CALLBACKS)

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.

- Kedro DataSet

- `pipelinex.MLflowDataSet(https://github.com/Minyus/pipelinex/blob/master/src/pipelinex/extras/datasets/mlflow/mlflow_dataset.py)`

Kedro DataSet that saves data to or loads data from MLflow. Set `dataset` argument as follows.

- * If `dataset` is set to a Kedro DataSet object or a dictionary, it will be saved/loaded as an MLFlow artifact.
- * If `dataset` is set to a string either {"json", "csv", "xls", "parquet", "png", "jpg", "jpeg", "img", "pkl", "txt", "yaml", "yml"}, Kedro DataSet object will be created with the string as the file extension and will be saved/loaded as an MLflow artifact. Under the hood, the following Kedro DataSet classes will be used (inspired by [Kedro Wings](#)).

```
dataset_dicts = {
    "json": {"type": "json.JSONDataSet"},
    "csv": {"type": "pandas.CSVDataSet"},
    "xls": {"type": "pandas.ExcelDataSet"},
    "parquet": {"type": "pandas.ParquetDataSet"},
    "pkl": {"type": "pickle.PickleDataSet"},
    "png": {"type": "pillow.ImageDataSet"},
    "jpg": {"type": "pillow.ImageDataSet"},
    "jpeg": {"type": "pillow.ImageDataSet"},
    "img": {"type": "pillow.ImageDataSet"},
    "txt": {"type": "text.TextDataSet"},
    "yaml": {"type": "yaml.YAMLDataSet"},
    "yml": {"type": "yaml.YAMLDataSet"},
}
```

- * If `dataset` is set to a string “p”, the value will be saved/loaded as an MLflow parameter (string).
- * If `dataset` is set to a string “m”, the value will be saved/loaded as an MLflow metric (numeric).
- * If `dataset` is set to None (default), MLflow will not be used.

Regarding all the options, 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, 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(
        enable_mlflow=True, # Enable configuring and logging to MLflow
        uri="sqlite:///mlruns/sqlite.db",
        experiment_name="experiment_001",
        artifact_location="./mlruns/experiment_001",
        offset_hours=0, # Specify the offset hour (e.g. 0 for UTC/GMT +00:00) to
        log in MLflow
    )
)
```

(continues on next page)

(continued from previous page)

```

), # Configure and log duration time for the pipeline
pipelinex.MLflowArtifactsLoggerHook(
    enable_mlflow=True, # Enable logging to MLflow
    filepaths_before_pipeline_run=[
        "conf/base/parameters.yml"
    ], # Optionally specify the file paths to log before pipeline is run
    filepaths_after_pipeline_run=[
        "data/06_models/model.pkl"
    ], # Optionally specify the file paths to log after pipeline is run
), # Log artifacts of specified file paths and dataset names
pipelinex.MLflowDataSetsLoggerHook(
    enable_mlflow=True, # Enable logging to MLflow
), # Log output datasets of (list of) float, int, and str classes
pipelinex.MLflowTimeLoggerHook(
    enable_mlflow=True, # Enable logging to MLflow
), # Log duration time to run each node (task)
pipelinex.AddTransformersHook(
    transformers=[
        pipelinex.MLflowIOTimeLoggerTransformer(
            enable_mlflow=True
        ) # Log duration time to load and save each dataset
    ],
), # Add transformers
)

```

8.2 Comparison with kedro-mlflow package

Both [PipelineX](#) 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.
- 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 MemoryDataSet objects (each Kedro node Python func input/output not listed in catalog) as MLflow parameters or metrics, instead of manually specifying a Kedro DataSet in the catalog. If the data type is float or int, the MemoryDataSet object will be logged as an MLflow metric. If the data type is str or data structure (either {list, tuple, dict, set}), the MemoryDataSet object will be logged as an MLflow parameter (after being stringified).
 - [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 0.17.0 (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.

INTEGRATION OF KEDRO WITH THE ADDITIONAL PYTHON PACKAGES AS KEDRO DATASETS, HOOKS (CALLBACKS), AND WRAPPERS.

Supplements to `kedro.extras` `pipelineX.extras` provides features 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
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.

9.6 Use with PyTorch Lightning

(To-do: contributions are welcome.)

9.7 Use with TensorFlow/Keras

(To-do: contributions are welcome.)

BUILD DOCKER IMAGE

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


WHY AND HOW PIPELINEX WAS BORN

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
TWELVE

AUTHOR

Yusuke Minami

- [<GitHub>](#)
- [<Linkedin>](#)
- [<Twitter>](#)

CONTRIBUTORS ARE WELCOME!

13.1 How to contribute

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

13.2 Contributor list

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

PIPELINEX

14.1 pipelinux package

14.1.1 Subpackages

14.1.1.1 pipelinux.extras package

14.1.1.1.1 Subpackages

14.1.1.1.1.1 pipelinux.extras.datasets package

14.1.1.1.1.2 Subpackages

14.1.1.1.1.3 pipelinux.extras.datasets.httpx package

14.1.1.1.1.4 Submodules

14.1.1.1.1.5 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)
```

14.1.1.1.1.6 pipelinex.extras.datasets.mlflow package

14.1.1.1.1.7 Submodules

14.1.1.1.1.8 pipelinex.extras.datasets.mlflow.mlflow_dataset module

```
class pipelinex.extras.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,  
                                                                      caching=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, caching=True, copy_mode=None, file_caching=True)
```

Parameters

- **dataset** (Union[AbstractDataSet, Dict, str, None]) – A Kedro DataSet object or a dictionary used to save/load. If set to either {"json", "csv", "xls", "parquet", "png", "jpg", "jpeg", "img", "pkl", "txt", "yaml", "yml"}, dataset instance will be created

accordingly with filepath arg. If set to “p”, the value will be saved/loaded as a parameter (string). If set to “m”, the value will be saved/loaded as a metric (numeric). If None (default), MLflow will not be used.

- **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 recommended to overwrite by a Kedro hook.
- **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.

14.1.1.1.1.9 pipelinex.extras.datasets.opencv package

14.1.1.1.1.10 Submodules

14.1.1.1.1.11 pipelinex.extras.datasets.opencv.images_dataset module

```
class pipelinex.extras.datasets.opencv.images_dataset.OpenCVImagesLocalDataSet (filepath,  
load_args=None,  
save_args=None,  
channel_first=False,  
version=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.

14.1.1.1.12 `pipelineX.extras.datasets.pandas` package

14.1.1.1.13 Submodules

14.1.1.1.14 `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 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 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.

14.1.1.1.15 `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)
```

14.1.1.1.16 `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.

14.1.1.1.17 `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`

14.1.1.1.18 `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`

14.1.1.1.1.19 `pipelinex.extras.datasets.pandas_profiling` package

14.1.1.1.1.20 Submodules

14.1.1.1.1.21 `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 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.

14.1.1.1.1.22 `pipelinex.extras.datasets.pillow` package

14.1.1.1.1.23 Submodules

14.1.1.1.1.24 `pipelinex.extras.datasets.pillow.images_dataset` module

class `pipelinex.extras.datasets.pillow.images_dataset.ImagesLocalDataSet` (*path,*
load_args=None,
save_args={'suffix':
'jpg'},
chan-
nel_first=False,
re-
verse_color=False,
ver-
sion=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*)

14.1.1.1.1.25 `pipelinex.extras.datasets.requests` package

14.1.1.1.1.26 Submodules

14.1.1.1.1.27 `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.requests import APIDataSet

data_set = APIDataSet(
    url="https://quickstats.nass.usda.gov"
    params={
        "key": "SOME_TOKEN",
        "format": "JSON",
        "commodity_desc": "CORN",
```

(continues on next page)

(continued from previous page)

```

        "statisticcat_des": "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>

14.1.1.1.1.28 `pipelinex.extras.datasets.seaborn` package

14.1.1.1.1.29 Submodules

14.1.1.1.1.30 `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.

14.1.1.1.1.31 `pipelinex.extras.datasets.torchvision` package

14.1.1.1.1.32 Submodules

14.1.1.1.1.33 `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.

14.1.1.1.1.34 Submodules

14.1.1.1.1.35 `pipelineX.extras.datasets.core` module

14.1.1.1.1.36 `pipelineX.extras.decorators` package

14.1.1.1.1.37 Submodules

14.1.1.1.1.38 `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.

14.1.1.1.1.39 `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

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.

14.1.1.1.1.40 `pipelineX.extras.decorators.mlflow_logger` module

`pipelineX.extras.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.

14.1.1.1.1.41 `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

14.1.1.1.1.42 `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

14.1.1.1.1.43 `pipelinex.extras.hooks` package

14.1.1.1.1.44 Subpackages

14.1.1.1.1.45 `pipelinex.extras.hooks.mlflow` package

14.1.1.1.1.46 Submodules

14.1.1.1.1.47 `pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger` module

class `pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLoggerHook` (*filepath*, *filepath*, *dataset*, *enable_n*)

Bases: object

Logs artifacts of specified file paths and dataset names to MLflow

__init__ (*filepath_before_pipeline_run=None*, *filepath_after_pipeline_run=None*, *dataset_after_node_run=None*, *enable_mlflow=True*)

Parameters

- **filepath_before_pipeline_run** (Optional[List[str]]) – The file paths of artifacts to log before the pipeline is run.
- **filepath_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)

14.1.1.1.1.48 `pipelinex.extras.hooks.mlflow.mlflow_basic_logger` module

```
class pipelinex.extras.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook (uri=None,
ex-
per-
i-
ment_name=None,
ar-
ti-
fact_location=None,
run_name=None,
off-
set_hours=0,
en-
able_logging_time_begin=True,
en-
able_logging_time_end=True,
en-
able_logging_time=True,
log-
ging_kedro_run_params=[],
en-
able_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, off-
set_hours=0, enable_logging_time_begin=True, enable_logging_time_end=True, en-
able_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
- **experiment_name** (Optional[str]) – The experiment name. *name* arg fed to: 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
- **run_name** (Optional[str]) – Shown as ‘Run Name’ in MLflow UI.
- **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)

```
pipelinex.extras.hooks.mlflow.mlflow_basic_logger.get_timestamp(dt=None, off-  
set_hours=0,  
fmt='%Y-%m-  
%dT%H:%M:%S')
```

```
pipelinex.extras.hooks.mlflow.mlflow_basic_logger.get_timestamp_int(dt=None,  
off-  
set_hours=0)
```

```
pipelinex.extras.hooks.mlflow.mlflow_basic_logger.get_timestamps(dt=None, off-  
set_hours=0)
```

14.1.1.1.1.49 pipelinex.extras.hooks.mlflow.mlflow_catalog_logger module

```
class pipelinex.extras.hooks.mlflow.mlflow_catalog_logger.MLflowCatalogLoggerHook (auto=True,  
mlflow_catalog=  
en-  
able_mlflow=
```

Bases: object

Logs datasets to MLflow

__init__ (auto=True, mlflow_catalog={}, enable_mlflow=True)

Parameters

- **auto** (bool) – If True, for each dataset (Python func input/output) not listed in mlflow_catalog, log as a metric for float and int types, and log as a param for str, list, tuple, dict, and set types.
- **mlflow_catalog** (Dict[str, Union[str, AbstractDataSet]]) – Specify how to log each dataset (Python func input/output) in dict format. Specify “p” to log as a parameter, “m” to log as a metric, either a file extension, “json”, “csv”, “xls”, “parquet”, “png”, “jpg”, “jpeg”, “img”, “pkl”, “txt”, “yaml”, or “yml”, or Kedro DataSet instance to log as a corresponding file artifact.
- **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.extras.hooks.mlflow.mlflow_catalog_logger.get_kedro_runner()
pipelineX.extras.hooks.mlflow.mlflow_catalog_logger.mlflow_log_dataset(dataset,
                                                                    enable_mlflow=True)
pipelineX.extras.hooks.mlflow.mlflow_catalog_logger.running_parallel()

```

14.1.1.1.1.50 pipelineX.extras.hooks.mlflow.mlflow_datasets_logger module

class pipelineX.extras.hooks.mlflow.mlflow_datasets_logger.**MLflowDataSetsLoggerHook** (*enable_mlflow*)
 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.extras.hooks.mlflow.mlflow_datasets_logger.**MLflowOutputsLoggerHook** (*enable_mlflow*)
 Bases: [pipelineX.extras.hooks.mlflow.mlflow_datasets_logger.MLflowDataSetsLoggerHook](#)

Deprecated alias for *MLflowOutputsLoggerHook*

14.1.1.1.1.51 pipelineX.extras.hooks.mlflow.mlflow_env_vars_logger module

class pipelineX.extras.hooks.mlflow.mlflow_env_vars_logger.**MLflowEnvVarsLoggerHook** (*param_env_vars, metric_env_vars, prefix, enable_mlflow*)
 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.extras.hooks.mlflow.mlflow_env_vars_logger.env_vars_to_dict(env_vars=[],
                                                                    prefix="")

```

```

pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger.log_metric_env_vars(env_vars=[],
                                                                           pre-
                                                                           fix="",
                                                                           en-
                                                                           able_mlflow=True)

pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger.log_param_env_vars(env_vars=[],
                                                                           pre-
                                                                           fix="",
                                                                           en-
                                                                           able_mlflow=True)

```

14.1.1.1.1.52 pipelinex.extras.hooks.mlflow.mlflow_time_logger module

```

class pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook(gantt_filepath=None,
                                                                              gantt_params={},
                                                                              met-
                                                                              ric_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) 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
- **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.extras.hooks.mlflow.mlflow_time_logger.dump_dict (filepath, d)`

`pipelineX.extras.hooks.mlflow.mlflow_time_logger.load_dict (filepath)`

14.1.1.1.1.53 pipelineX.extras.hooks.mlflow.mlflow_utils module

`pipelineX.extras.hooks.mlflow.mlflow_utils.mlflow_end_run (enable_mlflow=True)`

`pipelineX.extras.hooks.mlflow.mlflow_utils.mlflow_log_artifacts (paths, artifact_path=None, enable_mlflow=True)`

`pipelineX.extras.hooks.mlflow.mlflow_utils.mlflow_log_metrics (metrics, step=None, enable_mlflow=True)`

`pipelineX.extras.hooks.mlflow.mlflow_utils.mlflow_log_params (params, enable_mlflow=True)`

`pipelineX.extras.hooks.mlflow.mlflow_utils.mlflow_start_run (uri, experiment_name, artifact_location, run_name=None, enable_mlflow=True)`

14.1.1.1.1.54 Submodules

14.1.1.1.1.55 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`

14.1.1.1.1.56 `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`

14.1.1.1.1.57 pipelinex.extras.ops package

14.1.1.1.1.58 Subpackages

14.1.1.1.1.59 pipelinex.extras.ops.ignite package

14.1.1.1.1.60 Subpackages

14.1.1.1.1.61 pipelinex.extras.ops.ignite.declaratives package

14.1.1.1.1.62 Submodules

14.1.1.1.1.63 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
- **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.

14.1.1.1.1.64 `pipelinex.extras.ops.ignite.handlers` package

14.1.1.1.1.65 Submodules

14.1.1.1.1.66 `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_offset=True, *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.

14.1.1.1.1.67 `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.

14.1.1.1.1.68 `pipelineX.extras.ops.ignite.metrics` package

14.1.1.1.1.69 Submodules

14.1.1.1.1.70 `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

14.1.1.1.1.71 `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

14.1.1.1.1.72 `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.
```

14.1.1.1.1.73 Submodules

14.1.1.1.1.74 `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.
```

14.1.1.1.1.75 `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)
```

14.1.1.1.1.76 `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)
```

14.1.1.1.1.77 `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.cv2' from '/home/docs/checkouts/readthedocs.org/user_builds/pipelinex/checkout/pipelinex-0.4.9/pipelinex/extras/ops/opencv_ops.py'>

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)

```

14.1.1.1.1.78 `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 pipelinex.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 pipelinex.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 pipelinex.extras.ops.pandas_ops.DfDrop (**kwargs)
    Bases: pipelinex.extras.ops.pandas_ops.DfBaseMethod

    method = 'drop'

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

    method = 'drop_duplicates'

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

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

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

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

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

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

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

    method = 'ewm'

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

```

14.1.1.1.1.79 pipelineX.extras.ops.pytorch_ops module

```

class pipelineX.extras.ops.pytorch_ops.CrossEntropyLoss2d(weight=None,
                                                            size_average=None,
                                                            ignore_index=-100,
                                                            reduce=None, reduc-
                                                            tion='mean')

```

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`

```

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.

training: `bool`

```

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.

```
training: bool
```

```
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.

```
training: bool
```

```
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.

```
training: bool
```

```
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
```

```
training: bool
```

```
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.

```
training: bool
```

```
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.

```
training: bool
```

```
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.

```
training: bool
```

```
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`.

```
training: bool
```

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

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

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

class pipelinetx.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 pipelinetx.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 pipelinetx.extras.ops.pytorch_ops.TensorAvgPool1d(batchnorm=None, activation=None, *args,
                                                         **kwargs)
    Bases: pipelinetx.extras.ops.pytorch_ops.ModuleConvWrap
    core
        alias of torch.nn.modules.pooling.AvgPool1d
    training: bool

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

```

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,      ac-
                                                         tivation=None,      *args,
                                                         **kwargs)

Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap

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

training:  bool

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

Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap

core
    alias of torch.nn.modules.pooling.MaxPool2d

training:  bool

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

Bases: pipelinex.extras.ops.pytorch_ops.ModuleConvWrap

core
    alias of torch.nn.modules.pooling.MaxPool3d

training:  bool

class pipelinex.extras.ops.pytorch_ops.TensorMean (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.TensorMin (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.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 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)

```

14.1.1.1.180 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.
```

14.1.1.1.181 `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'
```

14.1.1.1.182 `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*)

14.1.1.1.1.83 `pipelineX.extras.transformers` package

14.1.1.1.1.84 Subpackages

14.1.1.1.1.85 `pipelineX.extras.transformers.mlflow` package

14.1.1.1.1.86 Submodules

14.1.1.1.1.87 `pipelineX.extras.transformers.mlflow.mlflow_io_time_logger` module

class `pipelineX.extras.transformers.mlflow.mlflow_io_time_logger.MLflowIOTimeLoggerTransformer`

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`

14.1.1.2 `pipelinex.framework` package

14.1.1.2.1 Subpackages

14.1.1.2.1.1 `pipelinex.framework.context` package

14.1.1.2.1.2 Submodules

14.1.1.2.1.3 `pipelinex.framework.context.context` module

14.1.1.2.1.4 `pipelinex.framework.context.flexible_catalog_context` module

```
class pipelinex.framework.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.

14.1.1.2.1.5 `pipelinex.framework.context.flexible_context` module

```
class pipelinex.framework.context.flexible_context.FlexibleContext (*args, **kwargs)

    Bases: pipelinex.framework.context.flexible_parameters_context.FlexibleParametersContext,
            pipelinex.framework.context.flexible_catalog_context.FlexibleCatalogContext,
            pipelinex.framework.context.flexible_run_context.FlexibleRunContext
```

`project_name = 'KedroProject'`

`project_version = '0.17.1'`

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

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

Deprecated alias for `FlexibleContext` for backward-compatibility

14.1.1.2.1.6 `pipelinex.framework.context.flexible_parameters_context` module

```
class pipelinex.framework.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.framework.context.flexible_parameters_context.import_modules(modules=None)`

14.1.1.2.1.7 `pipelinex.framework.context.flexible_run_context` module

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

Bases: `pipelinex.framework.context.save_pipeline_json_context.
SavePipelineJsonContext, pipelinex.framework.context.flexible_run_context.
StringRunnerOptionContext, pipelinex.framework.context.
flexible_run_context.OnlyMissingOptionContext`

```
class pipelinex.framework.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.framework.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.

14.1.1.2.1.8 `pipelinex.framework.context.save_pipeline_json_context` module

```
class pipelinex.framework.context.save_pipeline_json_context.SavePipelineJsonContext (package
project_
env=No
ex-
tra_par

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

14.1.1.2.2 Submodules

14.1.1.2.3 `pipelinex.framework.configure` module

```
pipelinex.framework.configure.configure_source (project_path, source_dir='src')
pipelinex.framework.configure.load_context (project_path, **kwargs)

Return type KedroContext
```

14.1.1.3 `pipelinex.hatch_dict` package

14.1.1.3.1 Submodules

14.1.1.3.2 `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)

```

14.1.1.4 pipelinex.pipeline package

14.1.1.4.1 Submodules

14.1.1.4.2 pipelinex.pipeline.pipeline module

```

class pipelinex.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.

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()
```

14.1.1.4.3 `pipelinex.pipeline.sub_pipeline` module

```
class pipelinex.pipeline.sub_pipeline.SubPipeline (inputs=None,      outputs=None,
                                                    func=None,      module="",  dec-
                                                    orator=None,      intermedi-
                                                    ate_node_name_fmt='{__}_{:03d}',
                                                    **kwargs)
```

Bases: `kedro.pipeline.pipeline.Pipeline`

```
__init__(inputs=None, outputs=None, func=None, module="", decorator=None, intermedi-
         ate_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
# 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()
```

14.1.2 Submodules

14.1.3 pipelinex.utils module

```
class pipelinex.utils.DictToDict (**kwargs)
    Bases: object
    __init__ (**kwargs)
        Initialize self. See help(type(self)) for accurate signature.
```

fn = None

module = None

```
class pipelinex.utils.ItemGetter (item)
    Bases: object
    __init__ (item)
        Initialize self. See help(type(self)) for accurate signature.
```

```
class pipelinex.utils.TransformCompose (transforms)
    Bases: object
    __init__ (transforms)
        Initialize self. See help(type(self)) for accurate signature.
```

`pipelinex.utils.dict_io (func)`

Return type Callable

`pipelinex.utils.dict_of_list_to_list_of_dict (dict_of_list)`

`pipelinex.utils.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.http`, 46
`pipeline.extras.datasets.http.async_api_dataset`, 46
`pipeline.extras.datasets.mlflow`, 47
`pipeline.extras.datasets.mlflow.mlflow_dataset`, 47
`pipeline.extras.datasets.opencv`, 48
`pipeline.extras.datasets.opencv.images_dataset`, 48
`pipeline.extras.datasets.pandas`, 49
`pipeline.extras.datasets.pandas.csv_local`, 49
`pipeline.extras.datasets.pandas.efficient_csv_local`, 50
`pipeline.extras.datasets.pandas.histogram`, 50
`pipeline.extras.datasets.pandas.pandas_cat_matrix`, 51
`pipeline.extras.datasets.pandas.pandas_describe`, 51
`pipeline.extras.datasets.pandas_profiling`, 52
`pipeline.extras.datasets.pandas_profiling.pandas_profiling`, 52
`pipeline.extras.datasets.pillow`, 52
`pipeline.extras.datasets.pillow.images_dataset`, 52
`pipeline.extras.datasets.requests`, 54
`pipeline.extras.datasets.requests.api_dataset`, 54
`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.mlflow_logger`, 57
`pipeline.extras.decorators.nvml_profiler`, 58
`pipeline.extras.decorators.pandas_decorators`, 58
`pipeline.extras.hooks`, 58
`pipeline.extras.hooks.add_catalog_dict`, 63
`pipeline.extras.hooks.add_transformers`, 64
`pipeline.extras.hooks.mlflow`, 58
`pipeline.extras.hooks.mlflow.mlflow_artifacts_logger`, 58
`pipeline.extras.hooks.mlflow.mlflow_basic_logger`, 59
`pipeline.extras.hooks.mlflow.mlflow_catalog_logger`, 60
`pipeline.extras.hooks.mlflow.mlflow_datasets_logger`, 61
`pipeline.extras.hooks.mlflow.mlflow_env_vars_logger`, 61
`pipeline.extras.hooks.mlflow.mlflow_time_logger`, 62
`pipeline.extras.hooks.mlflow.mlflow_utils`, 63
`pipeline.extras.ops`, 65
`pipeline.extras.ops.allennlp_ops`, 70
`pipeline.extras.ops.parse_args_ops`, 70
`pipeline.extras.ops.ignite`, 65
`pipeline.extras.ops.ignite.declaratives`, 65
`pipeline.extras.ops.ignite.declaratives.declarative_ops`, 65
`pipeline.extras.ops.ignite.handlers`, 67
`pipeline.extras.ops.ignite.handlers.flexible_checker`, 67

67
pipelinex.extras.ops.ignite.handlers.time_limit,
68
pipelinex.extras.ops.ignite.metrics, 68
pipelinex.extras.ops.ignite.metrics.cohen_kappa_score,
68
pipelinex.extras.ops.ignite.metrics.fbeta_score,
69
pipelinex.extras.ops.ignite.metrics.utils,
70
pipelinex.extras.ops.numpy_ops, 70
pipelinex.extras.ops.opencv_ops, 70
pipelinex.extras.ops.pandas_ops, 74
pipelinex.extras.ops.pytorch_ops, 80
pipelinex.extras.ops.shap_ops, 92
pipelinex.extras.ops.skimage_ops, 93
pipelinex.extras.ops.sklearn_ops, 93
pipelinex.extras.transformers, 95
pipelinex.extras.transformers.mlflow,
95
pipelinex.extras.transformers.mlflow.mlflow_io_time_logger,
95
pipelinex.framework, 96
pipelinex.framework.configure, 100
pipelinex.framework.context, 96
pipelinex.framework.context.context, 96
pipelinex.framework.context.flexible_catalog_context,
96
pipelinex.framework.context.flexible_context,
96
pipelinex.framework.context.flexible_parameters_context,
96
pipelinex.framework.context.flexible_run_context,
98
pipelinex.framework.context.save_pipeline_json_context,
100
pipelinex.hatch_dict, 100
pipelinex.hatch_dict.hatch_dict, 100
pipelinex.pipeline, 101
pipelinex.pipeline.pipeline, 101
pipelinex.pipeline.sub_pipeline, 102
pipelinex.utils, 103

INDEX

Symbols

method), 61

__init__ () (pipelinex.extras.datasets.mlflow.mlflow_dataset.MLflowDataset method), 47

__init__ () (pipelinex.extras.datasets.opencv.images_dataset.OpenCVImagesLocalDataset method), 48

__init__ () (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataset method), 49

__init__ () (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSVLocalDataset method), 50

__init__ () (pipelinex.extras.datasets.pandas.histogram.HistogramDataset method), 50

__init__ () (pipelinex.extras.datasets.pandas.pandas_cat_matrix.PandasCatMatrixDataset method), 51

__init__ () (pipelinex.extras.datasets.pandas.pandas_describe.PandasDescribeDataset method), 51

__init__ () (pipelinex.extras.datasets.pandas_profiling.pandas_profiling.PandasProfilingDataset method), 52

__init__ () (pipelinex.extras.datasets.pillow.images_dataset.ImagesLocalDataset method), 53

__init__ () (pipelinex.extras.datasets.pillow.images_dataset.NumpyDataset method), 53

__init__ () (pipelinex.extras.datasets.pillow.images_dataset.NumpyDatasetFromList method), 53

__init__ () (pipelinex.extras.datasets.requests.api_dataset.APIDataset method), 55

__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), 63

__init__ () (pipelinex.extras.hooks.add_transformers.AddTransformersHook method), 64

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLoggerHook method), 58

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook method), 59

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_catalog_logger.MLflowCatalogLoggerHook method), 60

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_datasets_logger.MLflowDatasetsLoggerHook method), 61

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger.MLflowEnvVarsLoggerHook method), 61

__init__ () (pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook method), 62

__init__ () (pipelinex.extras.ops.allennlp_ops.AllennlpReaderToDict method), 70

__init__ () (pipelinex.extras.ops.argmax_ops.FeedArgsDict method), 70

__init__ () (pipelinex.extras.ops.ignite.declaratives.declarative_trainer.DeclarativeTrainer method), 67

__init__ () (pipelinex.extras.ops.ignite.handlers.flexible_checkpoint.FlexibleCheckpoint method), 67

__init__ () (pipelinex.extras.ops.ignite.handlers.time_limit.TimeLimit method), 68

__init__ () (pipelinex.extras.ops.ignite.metrics.cohen_kappa_score.CohenKappaScore method), 68

__init__ () (pipelinex.extras.ops.ignite.metrics.fbeta_score.FbetaScore method), 69

__init__ () (pipelinex.extras.ops.ignite.metrics.utils.ClassificationOutput method), 70

__init__ () (pipelinex.extras.ops.numpy_ops.ReverseChannel method), 70

__init__ () (pipelinex.extras.ops.opencv_ops.CvBGR2Gray method), 70

__init__ () (pipelinex.extras.ops.opencv_ops.CvBGR2HSV method), 70

__init__ () (pipelinex.extras.ops.opencv_ops.CvDiagonalEdgeFilter2d method), 71

__init__ () (pipelinex.extras.ops.opencv_ops.CvModuleListMerge method), 72

__init__ () (pipelinex.extras.ops.opencv_ops.CvScale method), 72

__init__ () (pipelinex.extras.ops.opencv_ops.CvSobel method), 72

__init__ () (pipelinex.extras.ops.opencv_ops.CvThreshold method), 72

__init__ () (pipelinex.extras.ops.pandas_ops.DfAddRowStat method), 74

__init__ () (pipelinex.extras.ops.pandas_ops.DfAssignColumns method), 74

__init__ () (pipelinex.extras.ops.pandas_ops.DfBaseMethod method), 74

__init__ () (pipelinex.extras.ops.pandas_ops.DfBaseTask method), 74

<code>method</code>), 74	<code>method</code>), 81
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfColApply <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.ModuleSumSkip <code>method</code>), 74	<code>method</code>), 82
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfConcat <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.Pool1dMixIn <code>method</code>), 75	<code>method</code>), 83
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfCondReplace <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.Pool2dMixIn <code>method</code>), 75	<code>method</code>), 83
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfDropFilter <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.Pool3dMixIn <code>method</code>), 75	<code>method</code>), 83
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfDtypesApply <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.StatModule <code>method</code>), 75	<code>method</code>), 83
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfDuplicate <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.StepBinary <code>method</code>), 75	<code>method</code>), 83
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfEval <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorClamp <code>method</code>), 75	<code>method</code>), 84
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfFocusTransform <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorClampMax <code>method</code>), 76	<code>method</code>), 84
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfGetColIndexes <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorClampMin <code>method</code>), 76	<code>method</code>), 84
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfGetDummies <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorConstantLinear <code>method</code>), 76	<code>method</code>), 85
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfMap <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorCumsum <code>method</code>), 76	<code>method</code>), 85
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfMerge <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorForward <code>method</code>), 76	<code>method</code>), 86
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfRelative <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorNearestPad <code>method</code>), 77	<code>method</code>), 90
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfRename <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorProba <code>method</code>), 77	<code>method</code>), 90
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfRowApply <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorSlice <code>method</code>), 77	<code>method</code>), 91
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfSample <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorSqueeze <code>method</code>), 77	<code>method</code>), 91
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfSlice <code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.TensorUnsqueeze <code>method</code>), 78	<code>method</code>), 92
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfSpatialFeatures <code>__init__</code> () (pipelinex.extras.ops.shap_ops.ExplainModel <code>method</code>), 78	<code>method</code>), 92
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfStrftime <code>__init__</code> () (pipelinex.extras.ops.shap_ops.Scale <code>method</code>), 78	<code>method</code>), 92
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfToDatetime <code>__init__</code> () (pipelinex.extras.ops.sklearn_ops.DfBaseTransformer <code>method</code>), 79	<code>method</code>), 93
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfToTimedelta <code>__init__</code> () (pipelinex.extras.ops.sklearn_ops.DfTrainTestSplit <code>method</code>), 79	<code>method</code>), 94
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.DfTotalSeconds <code>__init__</code> () (pipelinex.extras.ops.sklearn_ops.ZeroToZeroTransformer <code>method</code>), 79	<code>method</code>), 94
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.NestedDictToDf <code>__init__</code> () (pipelinex.extras.transformers.mlflow.mlflow_io_time_logge <code>method</code>), 79	<code>method</code>), 95
<code>__init__</code> () (pipelinex.extras.ops.pandas_ops.SrMap <code>__init__</code> () (pipelinex.framework.context.flexible_parameters_context.F <code>method</code>), 79	<code>method</code>), 96
<code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.ModuleBottleneck2d <code>__init__</code> () (pipelinex.hatch_dict.hatch_dict.Construct <code>method</code>), 80	<code>method</code>), 100
<code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.ModuleConcatSkip <code>__init__</code> () (pipelinex.hatch_dict.hatch_dict.HatchDict <code>method</code>), 81	<code>method</code>), 100
<code>__init__</code> () (pipelinex.extras.ops.pytorch_ops.ModuleConvWrap <code>__init__</code> () (pipelinex.hatch_dict.hatch_dict.Method	

A

```

__init__() (method), 101
__init__() (pipelinex.hatch_dict.hatch_dict.ToPipeline as_tuple() (in module
method), 101 pipelinex.extras.ops.pytorch_ops), 92
__init__() (pipelinex.pipeline.pipeline.FlexiblePipelineAsyncAPIDataSet (class in
method), 101 pipelinex.extras.datasets.httpx.async_api_dataset),
__init__() (pipelinex.pipeline.sub_pipeline.SubPipeline
method), 102 46
__init__() (pipelinex.utils.DictToDict method), 103 asyncio_run() (in module
__init__() (pipelinex.utils.ItemGetter method), 103 pipelinex.extras.datasets.httpx.async_api_dataset),
__init__() (pipelinex.utils.TransformCompose
method), 103 47

B
before_node_run()
(pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook
method), 62
before_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLoggerHook
method), 59
before_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook
method), 60
before_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_catalog_logger.MLflowCatalogLoggerHook
method), 60
after_catalog_created()
(pipelinex.extras.hooks.add_catalog_dict.AddCatalogDictHook
method), 63
after_catalog_created()
(pipelinex.extras.hooks.add_transformers.AddTransformersHook
method), 64
after_catalog_created()
(pipelinex.extras.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook
method), 60
after_node_run() (pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLoggerHook
method), 59
after_node_run() (pipelinex.extras.hooks.mlflow.mlflow_catalog_logger.MLflowCatalogLoggerHook
method), 60
after_node_run() (pipelinex.extras.hooks.mlflow.mlflow_datasets_logger.MLflowDataSetsLoggerHook
method), 61
after_node_run() (pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook
method), 62
after_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger.MLflowArtifactsLoggerHook
method), 59
after_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_basic_logger.MLflowBasicLoggerHook
method), 60
after_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger.MLflowEnvVarsLoggerHook
method), 61
after_pipeline_run()
(pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook
method), 62

AllennlpReaderToDict (class in
pipelinex.extras.ops.allennlp_ops), 70
APIDataSet (class in
pipelinex.extras.datasets.requests.api_dataset),

```

C

```

AbstractInputTransform (class in
pipelinex.extras.ops.ignite.metrics.utils), 70
MLflowArtifactsLoggerHook (class in
pipelinex.extras.ops.ignite.metrics.cohen_kappa_score),
MLflowCatalogLoggerHook (class in
pipelinex.extras.ops.ignite.metrics.cohen_kappa_score.CohenKappaScore
compute()), 60
MLflowDataSetsLoggerHook (class in
pipelinex.extras.ops.ignite.metrics.fbeta_score.FbetaScore
compute()), 60
MLflowTimeLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.ModuleConvWrap
configure_source() (in module
pipelinex.framework.configure), 100
MLflowArtifactsLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.TensorAvgPool1d
attribute), 83
MLflowBasicLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.TensorAvgPool2d
attribute), 83
MLflowEnvVarsLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.TensorAvgPool3d
attribute), 83
MLflowTimeLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.TensorConv1d
attribute), 85
MLflowArtifactsLoggerHook (class in
pipelinex.extras.ops.pytorch_ops.TensorConv2d
attribute), 85

```

core (pipelinex.extras.ops.pytorch_ops.TensorConv3d attribute), 85	CvModuleSum (class in pipelinex.extras.ops.opencv_ops), 72
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool1d attribute), 89	CvResize (class in pipelinex.extras.ops.opencv_ops), 72
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool2d attribute), 89	CvScale (class in pipelinex.extras.ops.opencv_ops), 72
core (pipelinex.extras.ops.pytorch_ops.TensorMaxPool3d attribute), 89	CvSobel (class in pipelinex.extras.ops.opencv_ops), 72
	CvThreshold (class in pipelinex.extras.ops.opencv_ops), 72
CrossEntropyLoss2d (class in pipelinex.extras.ops.pytorch_ops), 80	D
CSVLocalDataSet (class in pipelinex.extras.datasets.pandas.csv_local), 49	DEFAULT_LOAD_ARGS (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet attribute), 49
CvBGR2Gray (class in pipelinex.extras.ops.opencv_ops), 70	DEFAULT_LOAD_ARGS (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSV attribute), 50
CvBGR2HSV (class in pipelinex.extras.ops.opencv_ops), 70	DEFAULT_PREVIEW_ARGS (pipelinex.extras.datasets.pandas.efficient_csv_local.EfficientCSV attribute), 50
CvBilateralFilter (class in pipelinex.extras.ops.opencv_ops), 71	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.pandas.csv_local.CSVLocalDataSet attribute), 49
CvBlur (class in pipelinex.extras.ops.opencv_ops), 71	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.pandas_profiling.pandas_profiling.Pandas attribute), 52
CvBoxFilter (class in pipelinex.extras.ops.opencv_ops), 71	DEFAULT_SAVE_ARGS (pipelinex.extras.datasets.seaborn.seaborn_pairplot.SeabornPair attribute), 56
CvCanny (class in pipelinex.extras.ops.opencv_ops), 71	degree_matrix() (in module pipelinex.extras.ops.pandas_ops), 79
CvCvtColor (class in pipelinex.extras.ops.opencv_ops), 71	df_set_index() (in module pipelinex.extras.decorators.pandas_decorators), 58
CvDiagonalEdgeFilter2d (class in pipelinex.extras.ops.opencv_ops), 71	DfAddRowStat (class in pipelinex.extras.ops.pandas_ops), 74
CvDictToDict (class in pipelinex.extras.ops.opencv_ops), 71	DfAgg (class in pipelinex.extras.ops.pandas_ops), 74
CvDilate (class in pipelinex.extras.ops.opencv_ops), 71	DfAggregate (class in pipelinex.extras.ops.pandas_ops), 74
CvEqualizeHist (class in pipelinex.extras.ops.opencv_ops), 71	DfApply (class in pipelinex.extras.ops.pandas_ops), 74
CvErode (class in pipelinex.extras.ops.opencv_ops), 71	DfApplymap (class in pipelinex.extras.ops.pandas_ops), 74
CvFilter2d (class in pipelinex.extras.ops.opencv_ops), 71	DfAssignColumns (class in pipelinex.extras.ops.pandas_ops), 74
CvGaussianBlur (class in pipelinex.extras.ops.opencv_ops), 71	DfBaseMethod (class in pipelinex.extras.ops.pandas_ops), 74
CvHoughLinesP (class in pipelinex.extras.ops.opencv_ops), 71	DfBaseTask (class in pipelinex.extras.ops.pandas_ops), 74
CvLine (class in pipelinex.extras.ops.opencv_ops), 72	DfBaseTransformer (class in pipelinex.extras.ops.sklearn_ops), 93
CvMedianBlur (class in pipelinex.extras.ops.opencv_ops), 72	DfColApply (class in pipelinex.extras.ops.pandas_ops), 74
CvModuleConcat (class in pipelinex.extras.ops.opencv_ops), 72	DfConcat (class in pipelinex.extras.ops.pandas_ops), 75
CvModuleL1 (class in pipelinex.extras.ops.opencv_ops), 72	
CvModuleL2 (class in pipelinex.extras.ops.opencv_ops), 72	
CvModuleListMerge (class in pipelinex.extras.ops.opencv_ops), 72	
CvModuleMean (class in pipelinex.extras.ops.opencv_ops), 72	
CvModuleStack (class in pipelinex.extras.ops.opencv_ops), 72	

DfCondReplace	(class	in	DfSample	(class in <i>pipelinex.extras.ops.pandas_ops</i>),
<i>pipelinex.extras.ops.pandas_ops</i>),	75		77	
DfDrop	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	75	DfSelectDtypes	(class in
DfDropDuplicates	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	77
<i>pipelinex.extras.ops.pandas_ops</i>),	75		DfSelectDtypesCols	(class in
DfDropFilter	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	75		DfSetIndex	(class in
DfDtypesApply	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	75		DfShift	(class in <i>pipelinex.extras.ops.pandas_ops</i>),
DfDuplicate	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	75		DfSortValues	(class in
DfEval	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	75	<i>pipelinex.extras.ops.pandas_ops</i>),	78
DfEwm	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	75	DfSpatialFeatures	(class in
DfExpanding	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	75		DfStandardScaler	(class in
DfFillna	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		<i>pipelinex.extras.ops.sklearn_ops</i>),	94
76			DfStrftime	(class in
DfFilter	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		<i>pipelinex.extras.ops.pandas_ops</i>),	78
76			DfTail	(class in <i>pipelinex.extras.ops.pandas_ops</i>),
DfFilterCols	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	76		DfToDatetime	(class in
DfFocusTransform	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	78
<i>pipelinex.extras.ops.pandas_ops</i>),	76		DfTotalSeconds	(class in
DfGetColIndexes	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	79
<i>pipelinex.extras.ops.pandas_ops</i>),	76		DfToTimedelta	(class in
DfGetCols	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		<i>pipelinex.extras.ops.pandas_ops</i>),	79
76			DfTrainTestSplit	(class in
DfGetDummies	(class	in	<i>pipelinex.extras.ops.sklearn_ops</i>),	94
<i>pipelinex.extras.ops.pandas_ops</i>),	76		DfTransform	(class in
DfGroupby	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		<i>pipelinex.extras.ops.pandas_ops</i>),	79
76			dict_io()	(in module <i>pipelinex.utils</i>),
DfHead	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	76	dict_of_list_to_list_of_dict()	(in module
DfMap	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	76	<i>pipelinex.utils</i>),	103
DfMerge	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	76	dict_string_val_prefix()	(in module
DfMinMaxScaler	(class	in	<i>pipelinex.extras.datasets.pandas.efficient_csv_local</i>),	
<i>pipelinex.extras.ops.sklearn_ops</i>),	94		50	
DfNgroup	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		dict_val_replace_except()	(in module
77			<i>pipelinex.extras.datasets.pandas.efficient_csv_local</i>),	
DfPipe	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	77	50	
DfQuantileTransformer	(class	in	DictToDict	(class in <i>pipelinex.utils</i>),
<i>pipelinex.extras.ops.sklearn_ops</i>),	94		103	
DfQuery	(class in <i>pipelinex.extras.ops.pandas_ops</i>),	77	distance_matrix()	(in module
DfRelative	(class	in	<i>pipelinex.extras.ops.pandas_ops</i>),	79
<i>pipelinex.extras.ops.pandas_ops</i>),	77		distance_to_affinity()	(in module
DfRename	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		<i>pipelinex.extras.ops.pandas_ops</i>),	79
77			dot_flatten()	(in module
DfResample	(class	in	<i>pipelinex.hatch_dict.hatch_dict</i>),	101
<i>pipelinex.extras.ops.pandas_ops</i>),	77		dump_dict()	(in module
DfResetIndex	(class	in	<i>pipelinex.extras.hooks.mlflow.mlflow_time_logger</i>),	
<i>pipelinex.extras.ops.pandas_ops</i>),	77		63	
DfRolling	(class in <i>pipelinex.extras.ops.pandas_ops</i>),		E	
77			EfficientCSVLocalDataSet	(class in
DfRowApply	(class	in	<i>pipelinex.extras.datasets.pandas.efficient_csv_local</i>),	
<i>pipelinex.extras.ops.pandas_ops</i>),	77		50	

<code>eigen()</code> (in module <code>pipelinex.extras.ops.pandas_ops</code>), 79	98
<code>element_wise_average()</code> (in module <code>pipelinex.extras.ops.pytorch_ops</code>), 92	fn (<code>pipelinex.extras.ops.opencv_ops.CvBGR2Gray</code> attribute), 70
<code>element_wise_prod()</code> (in module <code>pipelinex.extras.ops.pytorch_ops</code>), 92	fn (<code>pipelinex.extras.ops.opencv_ops.CvBGR2HSV</code> attribute), 71
<code>element_wise_sum()</code> (in module <code>pipelinex.extras.ops.pytorch_ops</code>), 92	fn (<code>pipelinex.extras.ops.opencv_ops.CvBilateralFilter</code> attribute), 71
<code>env_vars_to_dict()</code> (in module <code>pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger</code>), 61	fn (<code>pipelinex.extras.ops.opencv_ops.CvBlur</code> attribute), 71
<code>EstimatorTransformer</code> (class in <code>pipelinex.extras.ops.sklearn_ops</code>), 94	fn (<code>pipelinex.extras.ops.opencv_ops.CvBoxFilter</code> attribute), 71
<code>expand_repeat()</code> (in module <code>pipelinex.extras.ops.opencv_ops</code>), 73	fn (<code>pipelinex.extras.ops.opencv_ops.CvCanny</code> attribute), 71
<code>ExplainModel</code> (class in <code>pipelinex.extras.ops.shap_ops</code>), 92	fn (<code>pipelinex.extras.ops.opencv_ops.CvCvtColor</code> attribute), 71
<code>extract_from_df()</code> (in module <code>pipelinex.extras.ops.sklearn_ops</code>), 94	fn (<code>pipelinex.extras.ops.opencv_ops.CvDilate</code> attribute), 71
F	fn (<code>pipelinex.extras.ops.opencv_ops.CvEqualizeHist</code> attribute), 71
<code>FbetaScore</code> (class in <code>pipelinex.extras.ops.ignite.metrics.fbeta_score</code>), 69	fn (<code>pipelinex.extras.ops.opencv_ops.CvErode</code> attribute), 71
<code>feed()</code> (in module <code>pipelinex.hatch_dict.hatch_dict</code>), 101	fn (<code>pipelinex.extras.ops.opencv_ops.CvFilter2d</code> attribute), 71
<code>FeedArgsDict</code> (class in <code>pipelinex.extras.ops argparse_ops</code>), 70	fn (<code>pipelinex.extras.ops.opencv_ops.CvGaussianBlur</code> attribute), 71
<code>fit()</code> (<code>pipelinex.extras.ops.sklearn_ops.DfBaseTransformer</code> method), 93	fn (<code>pipelinex.extras.ops.opencv_ops.CvHoughLinesP</code> attribute), 72
<code>fit_to_l()</code> (in module <code>pipelinex.extras.ops.opencv_ops</code>), 73	fn (<code>pipelinex.extras.ops.opencv_ops.CvLine</code> attribute), 72
<code>fit_to_uint8()</code> (in module <code>pipelinex.extras.ops.opencv_ops</code>), 73	fn (<code>pipelinex.extras.ops.opencv_ops.CvMedianBlur</code> attribute), 72
<code>fit_transform()</code> (<code>pipelinex.extras.ops.sklearn_ops.DfBaseTransformer</code> method), 93	fn (<code>pipelinex.extras.ops.opencv_ops.CvResize</code> attribute), 72
<code>fit_transform()</code> (<code>pipelinex.extras.ops.sklearn_ops.ZeroToZeroTransformer</code> method), 94	fn (<code>pipelinex.extras.ops.opencv_ops.CvSobel</code> attribute), 72
<code>FlexibleCatalogContext</code> (class in <code>pipelinex.framework.context.flexible_catalog_context</code>), 96	fn (<code>pipelinex.extras.ops.opencv_ops.CvThreshold</code> attribute), 72
<code>FlexibleContext</code> (class in <code>pipelinex.framework.context.flexible_context</code>), 96	fn (<code>pipelinex.extras.ops.opencv_ops.NpAbs</code> attribute), 73
<code>FlexibleModelCheckpoint</code> (class in <code>pipelinex.extras.ops.ignite.handlers.flexible_checkpoint</code>), 67	fn (<code>pipelinex.extras.ops.opencv_ops.NpConcat</code> attribute), 73
<code>FlexibleParametersContext</code> (class in <code>pipelinex.framework.context.flexible_parameters_context</code>), 96	fn (<code>pipelinex.extras.ops.opencv_ops.NpFullLike</code> attribute), 73
<code>FlexiblePipeline</code> (class in <code>pipelinex.pipeline.pipeline</code>), 101	fn (<code>pipelinex.extras.ops.opencv_ops.NpMean</code> attribute), 73
<code>FlexibleRunContext</code> (class in <code>pipelinex.framework.context.flexible_run_context</code>),	fn (<code>pipelinex.extras.ops.opencv_ops.NpOnesLike</code> attribute), 73
	fn (<code>pipelinex.extras.ops.opencv_ops.NpSqrt</code> attribute), 73
	fn (<code>pipelinex.extras.ops.opencv_ops.NpSquare</code> attribute), 73
	fn (<code>pipelinex.extras.ops.opencv_ops.NpStack</code> attribute), 73
	fn (<code>pipelinex.extras.ops.opencv_ops.NpSum</code> attribute), 73

fn (pipelinex.extras.ops.opencv_ops.NpZerosLike attribute), 73 method), 89
 forward() (pipelinex.extras.ops.pytorch_ops.TensorNearestPad method), 90
 fn (pipelinex.extras.ops.skimage_ops.SkimageMarkBoundaries attribute), 93 forward() (pipelinex.extras.ops.pytorch_ops.TensorProba method), 90
 fn (pipelinex.extras.ops.skimage_ops.SkimageSegmentationFelzenszwaldf attribute), 93 forward() (pipelinex.extras.ops.pytorch_ops.TensorRange method), 90
 fn (pipelinex.extras.ops.skimage_ops.SkimageSegmentationQuickshift attribute), 93 forward() (pipelinex.extras.ops.pytorch_ops.TensorSkip method), 90
 fn (pipelinex.extras.ops.skimage_ops.SkimageSegmentationSlic attribute), 93 forward() (pipelinex.extras.ops.pytorch_ops.TensorSlice method), 91
 fn (pipelinex.extras.ops.skimage_ops.SkimageSegmentationWatershed attribute), 93 forward() (pipelinex.extras.ops.pytorch_ops.TensorSqueeze method), 91
 fn (pipelinex.utils.DictToDict attribute), 103 method), 91
 forward() (pipelinex.extras.ops.pytorch_ops.CrossEntropyLoss2d method), 80 forward() (pipelinex.extras.ops.pytorch_ops.TensorSum method), 91
 forward() (pipelinex.extras.ops.pytorch_ops.ModuleAvg forward() (pipelinex.extras.ops.pytorch_ops.TensorUnsqueeze method), 80 method), 92
 forward() (pipelinex.extras.ops.pytorch_ops.ModuleConcat method), 81
 forward() (pipelinex.extras.ops.pytorch_ops.ModuleListMerge method), 81
 forward() (pipelinex.extras.ops.pytorch_ops.ModuleProd method), 82
 forward() (pipelinex.extras.ops.pytorch_ops.ModuleSum method), 82
 forward() (pipelinex.extras.ops.pytorch_ops.NLLoss method), 82
 forward() (pipelinex.extras.ops.pytorch_ops.StepBinary method), 83
 forward() (pipelinex.extras.ops.pytorch_ops.TensorClamp method), 84
 forward() (pipelinex.extras.ops.pytorch_ops.TensorClampMax method), 84
 forward() (pipelinex.extras.ops.pytorch_ops.TensorClampMin method), 84
 forward() (pipelinex.extras.ops.pytorch_ops.TensorConstantLinear method), 85
 forward() (pipelinex.extras.ops.pytorch_ops.TensorCumsum method), 85
 forward() (pipelinex.extras.ops.pytorch_ops.TensorExp method), 86
 forward() (pipelinex.extras.ops.pytorch_ops.TensorFlatten method), 86
 forward() (pipelinex.extras.ops.pytorch_ops.TensorForward method), 86
 forward() (pipelinex.extras.ops.pytorch_ops.TensorIdentity method), 88
 forward() (pipelinex.extras.ops.pytorch_ops.TensorLog method), 88
 forward() (pipelinex.extras.ops.pytorch_ops.TensorMax method), 88
 forward() (pipelinex.extras.ops.pytorch_ops.TensorMean method), 89
 forward() (pipelinex.extras.ops.pytorch_ops.TensorMin method), 89

G
 Merge (class in pipelinex.hatch_dict.hatch_dict), 100
 get() (pipelinex.hatch_dict.hatch_dict.HatchDict method), 100
 get_kedro_runner() (in module pipelinex.extras.hooks.mlflow.mlflow_catalog_logger), 60
 get_nv_info() (in module pipelinex.extras.decorators.nvml_profiler), 58
 get_params() (pipelinex.hatch_dict.hatch_dict.HatchDict method), 100
 get_timestamp() (in module pipelinex.extras.hooks.mlflow.mlflow_basic_logger), 60
 get_timestamp_int() (in module pipelinex.extras.hooks.mlflow.mlflow_basic_logger), 60
 get_timestamps() (in module pipelinex.extras.hooks.mlflow.mlflow_basic_logger), 60

H
 HatchDict (class in pipelinex.hatch_dict.hatch_dict), 100
 HistogramDataSet (class in pipelinex.extras.datasets.pandas.histogram), 50

I
 ignore_index (pipelinex.extras.ops.pytorch_ops.CrossEntropyLoss2d attribute), 80
 ignore_index (pipelinex.extras.ops.pytorch_ops.NLLoss attribute), 82

ImagesLocalDataSet	(class in <i>Method</i> (class in <i>pipelinex.hatch_dict.hatch_dict</i>), 100 <i>pipelinex.extras.datasets.pillow.images_dataset</i>), method (attribute), 74 52
import_modules()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfAggregate <i>pipelinex.framework.context.flexible_parameters_context</i>), attribute), 74 97
ItemGetter	(class in <i>pipelinex.utils</i>), 103
items()	(<i>pipelinex.hatch_dict.hatch_dict.HatchDict</i> method), 100 method (pipelinex.extras.ops.pandas_ops.DfApplymap attribute), 74
IterableImagesDataSet	(class in <i>method</i> (pipelinex.extras.ops.pandas_ops.DfBaseMethod <i>pipelinex.extras.datasets.torchvision.iterable_images_dataset</i>), attribute), 74 56
K	
keys()	(<i>pipelinex.hatch_dict.hatch_dict.HatchDict</i> method), 100 method (pipelinex.extras.ops.pandas_ops.DfDrop attribute), 75 method (pipelinex.extras.ops.pandas_ops.DfDropDuplicates attribute), 75
L	
laplacian_eigen()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfEwm <i>pipelinex.extras.ops.pandas_ops</i>), attribute), 75 79
laplacian_matrix()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfExpanding <i>pipelinex.extras.ops.pandas_ops</i>), attribute), 75 79
list_of_dict_to_dict_of_list()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfFillna <i>pipelinex.utils</i>), attribute), 76 103
load()	(<i>pipelinex.extras.transformers.mlflow.mlflow_io_time_logger.MlflowIOTimeLoggerTransformer</i> method), 95 method (pipelinex.extras.ops.pandas_ops.DfHead attribute), 76
load_context()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfNgroup <i>pipelinex.framework.configure</i>), attribute), 77 100
load_dict()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfPipe <i>pipelinex.extras.hooks.mlflow.mlflow_time_logger</i>), attribute), 77 63
load_image()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfQuery <i>pipelinex.extras.datasets.pillow.images_dataset</i>), attribute), 77 53
load_obj()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfResample <i>pipelinex.hatch_dict.hatch_dict</i>), attribute), 77 101
load_time_dict()	(<i>pipelinex.extras.hooks.mlflow.mlflow_time_logger.MlflowTimeLoggerHook</i> method), 63 method (pipelinex.extras.ops.pandas_ops.DfResetIndex attribute), 77
log_df_summary()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfRolling <i>pipelinex.extras.decorators.pandas_decorators</i>), attribute), 77 58
log_metric_env_vars()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfSelectDtypes <i>pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger</i>), attribute), 78 62
log_param_env_vars()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfSetIndex <i>pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger</i>), attribute), 78 62
log_time()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfShift <i>pipelinex.extras.decorators.decorators</i>), attribute), 78 57
M	
mem_profile()	(in module <i>method</i> (pipelinex.extras.ops.pandas_ops.DfSortValues <i>pipelinex.extras.decorators.memory_profiler</i>), attribute), 78 57
	method (pipelinex.hatch_dict.hatch_dict.Get attribute), 100 method (pipelinex.hatch_dict.hatch_dict.Method at-

<i>tribute</i>), 101	module
<code>mix_up()</code> (in module <i>pipelinex.extras.ops.opencv_ops</i>), 73	<i>pipelinex</i> , 46
<code>mlflow_end_run()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_utils</i>), 63	<i>pipelinex.extras</i> , 46
<code>mlflow_log_artifacts()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_utils</i>), 63	<i>pipelinex.extras.datasets</i> , 46
<code>mlflow_log_dataset()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_catalog_logger</i>), 61	<i>pipelinex.extras.datasets.core</i> , 57
<code>mlflow_log_metrics()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_utils</i>), 63	<i>pipelinex.extras.datasets.httpx</i> , 46
<code>mlflow_log_params()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_utils</i>), 63	<i>pipelinex.extras.datasets.httpx.async_api_datas</i> , 46
<code>mlflow_log_time()</code> (in module <i>pipelinex.extras.decorators.mlflow_logger</i>), 57	<i>pipelinex.extras.datasets.mlflow</i> , 47
<code>mlflow_start_run()</code> (in module <i>pipelinex.extras.hooks.mlflow.mlflow_utils</i>), 63	<i>pipelinex.extras.datasets.mlflow.mlflow_dataset</i> , 47
<i>MLflowArtifactsLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_artifacts_logger</i>), 58	<i>pipelinex.extras.datasets.opencv</i> , 48
<i>MLflowBasicLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_basic_logger</i>), 59	<i>pipelinex.extras.datasets.opencv.images_dataset</i> , 48
<i>MLflowCatalogLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_catalog_logger</i>), 60	<i>pipelinex.extras.datasets.pandas</i> , 49
<i>MLflowDataSet</i> (class in <i>pipelinex.extras.datasets.mlflow.mlflow_dataset</i>), 47	<i>pipelinex.extras.datasets.pandas.csv_local</i> , 49
<i>MLflowDataSetsLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_datasets_logger</i>), 61	<i>pipelinex.extras.datasets.pandas.efficient_csv</i> , 50
<i>MLflowEnvVarsLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_env_vars_logger</i>), 61	<i>pipelinex.extras.datasets.pandas.histogram</i> , 50
<i>MLflowFlexibleContext</i> (class in <i>pipelinex.framework.context.flexible_context</i>), 96	<i>pipelinex.extras.datasets.pandas.pandas_cat_mat</i> , 51
<i>MLflowIOTimeLoggerTransformer</i> (class in <i>pipelinex.extras.transformers.mlflow.mlflow_io_time_logger</i>), 95	<i>pipelinex.extras.datasets.pandas.pandas_describ</i> , 51
<i>MLflowOutputsLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_datasets_logger</i>), 61	<i>pipelinex.extras.datasets.pandas.pandas_profiling</i> , 52
<i>MLflowTimeLoggerHook</i> (class in <i>pipelinex.extras.hooks.mlflow.mlflow_time_logger</i>), 62	<i>pipelinex.extras.datasets.pandas.pandas_profiling.pand</i> , 52
	<i>pipelinex.extras.datasets.pillow</i> , 52
	<i>pipelinex.extras.datasets.pillow.images_dataset</i> , 52
	<i>pipelinex.extras.datasets.requests</i> , 54
	<i>pipelinex.extras.datasets.requests.api_dataset</i> , 54
	<i>pipelinex.extras.datasets.seaborn</i> , 56
	<i>pipelinex.extras.datasets.seaborn.seaborn_pairp</i> , 56
	<i>pipelinex.extras.datasets.torchvision</i> , 56
	<i>pipelinex.extras.datasets.torchvision.iterable</i> , 56
	<i>pipelinex.extras.decorators</i> , 57
	<i>pipelinex.extras.decorators.decorators</i> , 57
	<i>pipelinex.extras.decorators.memory_profiler</i> , 57
	<i>pipelinex.extras.decorators.mlflow_logger</i> , 57
	<i>pipelinex.extras.decorators.nvml_profiler</i> , 58
	<i>pipelinex.extras.decorators.pandas_decorators</i> ,

58
 pipelindex.extras.hooks, 58
 pipelindex.extras.hooks.add_catalog_dict, 58
 63
 pipelindex.extras.hooks.add_transformers, 96
 64
 pipelindex.extras.hooks.mlflow, 58
 pipelindex.extras.hooks.mlflow.mlflow_artifacts_logger, 96
 58
 pipelindex.extras.hooks.mlflow.mlflow_basic_logger, 96
 59
 pipelindex.extras.hooks.mlflow.mlflow_catalog_logger, 96
 60
 pipelindex.extras.hooks.mlflow.mlflow_dataset_logger, 96
 61
 pipelindex.extras.hooks.mlflow.mlflow_env_variables_logger, 96
 61
 pipelindex.extras.hooks.mlflow.mlflow_time_logger, 96
 62
 pipelindex.extras.hooks.mlflow.mlflow_utils, 96
 63
 pipelindex.extras.ops, 65
 pipelindex.extras.ops.allennlp_ops, 70
 pipelindex.extras.ops.argparse_ops, 70
 pipelindex.extras.ops.ignite, 65
 pipelindex.extras.ops.ignite.declaratives, 65
 pipelindex.extras.ops.ignite.declaratives.declarative_trainer, 65
 pipelindex.extras.ops.ignite.handlers, 67
 pipelindex.extras.ops.ignite.handlers.ModuleHotCheckpoint, 67
 pipelindex.extras.ops.ignite.handlers.ModuleMonitor, 68
 pipelindex.extras.ops.ignite.metrics, 68
 pipelindex.extras.ops.ignite.metrics.ModuleConcatSkip, 68
 pipelindex.extras.ops.ignite.metrics.ModuleCappedWrap, 68
 pipelindex.extras.ops.ignite.metrics.ModuleScoreMerge, 69
 pipelindex.extras.ops.ignite.metrics.ModuleProd, 70
 pipelindex.extras.ops.numpy_ops, 70
 pipelindex.extras.ops.opencv_ops, 70
 pipelindex.extras.ops.pandas_ops, 74
 pipelindex.extras.ops.pytorch_ops, 80
 pipelindex.extras.ops.shap_ops, 92
 pipelindex.extras.ops.skimage_ops, 93
 pipelindex.extras.ops.sklearn_ops, 93
 pipelindex.extras.transformers, 95
 pipelindex.extras.transformers.mlflow, 95
 pipelindex.extras.transformers.mlflow.mlflow_io, 95
 pipelindex.framework, 96
 pipelindex.framework.configure, 100
 pipelindex.framework.context, 96
 pipelindex.framework.context.context, 96
 pipelindex.framework.context.flexible_catalog_context, 96
 pipelindex.framework.context.flexible_context, 96
 pipelindex.framework.context.flexible_parameters, 96
 pipelindex.framework.context.flexible_run_context, 96
 pipelindex.framework.context.save_pipeline_json, 96
 pipelindex.hatch_dict, 100
 pipelindex.hatch_dict.hatch_dict, 100
 pipelindex.pipeline, 101
 pipelindex.pipeline.pipeline, 101
 pipelindex.pipeline.sub_pipeline, 102
 pipelindex.utils, 103
 module (pipelindex.extras.ops.opencv_ops.CvDictToDict attribute), 71
 module (pipelindex.extras.ops.opencv_ops.NpDictToDict attribute), 73
 module (pipelindex.extras.ops.skimage_ops.SkimageSegmentationDictToDict attribute), 93
 module (pipelindex.utils.DictToDict attribute), 103
 ModuleAvg (class in pipelindex.extras.ops.pytorch_ops), 80
 ModuleHotCheckpoint (class in pipelindex.extras.ops.pytorch_ops), 80
 ModuleMonitor (class in pipelindex.extras.ops.pytorch_ops), 81
 ModuleConcatSkip (class in pipelindex.extras.ops.pytorch_ops), 81
 ModuleCappedWrap (class in pipelindex.extras.ops.pytorch_ops), 81
 ModuleScoreMerge (class in pipelindex.extras.ops.pytorch_ops), 81
 ModuleProd (class in pipelindex.extras.ops.pytorch_ops), 81
 ModuleSum (class in pipelindex.extras.ops.pytorch_ops), 82
 ModuleSumSkip (class in pipelindex.extras.ops.pytorch_ops), 82
 N
 namespace () (in module pipelindex.extras.ops.argparse_ops), 70

[nested_dict_to_df\(\)](#) (in module [PandasProfilingDataSet](#) (class in [pipelinex.extras.ops.pandas_ops](#)), 80
[pipelinex.extras.datasets.pandas_profiling.pandas_profiling](#)),
[NestedDictToDf](#) (class in [pipelinex.extras.ops.pandas_ops](#)), 79
[52](#)
[params\(\)](#) ([pipelinex.framework.context.flexible_parameters_context.FlexibleParametersContext](#)), (in module [pipelinex.hatch_dict.hatch_dict](#)),
[NetworkTrain](#) (class in [pipelinex.extras.ops.ignite.declaratives.declarative_pipeline](#)), 65
[101](#)
[nl_loss\(\)](#) (in module [pipelinex.extras.ops.pytorch_ops](#)), 92
[pass_through\(\)](#) (in module [pipelinex.hatch_dict.hatch_dict](#)), 101
[NLLoss](#) (class in [pipelinex.extras.ops.pytorch_ops](#)), 82
[pipelinex](#)
[Np3DArrDataset](#) (class in [pipelinex.extras.datasets.pillow.images_dataset](#)), 53
[module, 46](#)
[pipelinex.extras](#)
[module, 46](#)
[Np3DArrDatasetFromList](#) (class in [pipelinex.extras.datasets.pillow.images_dataset](#)), 53
[pipelinex.extras.datasets](#)
[module, 46](#)
[pipelinex.extras.datasets.core](#)
[module, 57](#)
[NpAbs](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 72
[pipelinex.extras.datasets.httpx](#)
[module, 46](#)
[NpConcat](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.httpx.async_api_dataset](#)
[module, 46](#)
[NpDictToDict](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.mlflow](#)
[module, 47](#)
[NpFullLike](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.mlflow.mlflow_dataset](#)
[module, 47](#)
[NpMean](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.opencv](#)
[module, 48](#)
[NpOnesLike](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.opencv.images_dataset](#)
[module, 48](#)
[NpSqrt](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pandas](#)
[module, 49](#)
[NpSquare](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pandas.csv_local](#)
[module, 49](#)
[NpStack](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pandas.efficient_csv_local](#)
[module, 50](#)
[NpSum](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pandas.histogram](#)
[module, 50](#)
[NpZerosLike](#) (class in [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pandas.pandas_cat_matrix](#)
[module, 51](#)
[nvml_profile\(\)](#) (in module [pipelinex.extras.decorators.nvml_profiler](#)), 58
[pipelinex.extras.datasets.pandas.pandas_describe](#)
[module, 51](#)
O
[OnlyMissingOptionContext](#) (class in [pipelinex.framework.context.flexible_run_context](#)), 98
[pipelinex.extras.datasets.pandas_profiling](#)
[module, 52](#)
[OpenCVImagesLocalDataSet](#) (class in [pipelinex.extras.datasets.opencv.images_dataset](#)), 48
[pipelinex.extras.datasets.pandas_profiling.pandas_profiling](#)
[module, 52](#)
[overlay\(\)](#) (in module [pipelinex.extras.ops.opencv_ops](#)), 73
[pipelinex.extras.datasets.pillow](#)
[module, 52](#)
P
[pipelinex.extras.datasets.pillow.images_dataset](#)
[module, 52](#)
[PandasCatMatrixDataSet](#) (class in [pipelinex.extras.datasets.pandas.pandas_cat_matrix](#)), 51
[pipelinex.extras.datasets.requests](#)
[module, 54](#)
[PandasDescribeDataSet](#) (class in [pipelinex.extras.datasets.pandas.pandas_describe](#)), 51
[pipelinex.extras.datasets.requests.api_dataset](#)
[module, 54](#)
[pipelinex.extras.datasets.seaborn](#)

module, 56	module, 67
pipelineX.extras.datasets.seaborn.seaborn	pipelineX.extras.ops.ignite.handlers.flexible_check
module, 56	module, 67
pipelineX.extras.datasets.torchvision	pipelineX.extras.ops.ignite.handlers.time_limit
module, 56	module, 68
pipelineX.extras.datasets.torchvision.it	pipelineX.extras.ops.ignite.metrics
module, 56	module, 68
pipelineX.extras.decorators	pipelineX.extras.ops.ignite.metrics.cohen_kappa_sc
module, 57	module, 68
pipelineX.extras.decorators.decorators	pipelineX.extras.ops.ignite.metrics.fbeta_score
module, 57	module, 69
pipelineX.extras.decorators.memory_profile	pipelineX.extras.ops.ignite.metrics.utils
module, 57	module, 70
pipelineX.extras.decorators.mlflow_logger	pipelineX.extras.ops.numpy_ops
module, 57	module, 70
pipelineX.extras.decorators.nvml_profile	pipelineX.extras.ops.opencv_ops
module, 58	module, 70
pipelineX.extras.decorators.pandas_decorator	pipelineX.extras.ops.pandas_ops
module, 58	module, 74
pipelineX.extras.hooks	pipelineX.extras.ops.pytorch_ops
module, 58	module, 80
pipelineX.extras.hooks.add_catalog_dict	pipelineX.extras.ops.shap_ops
module, 63	module, 92
pipelineX.extras.hooks.add_transformers	pipelineX.extras.ops.skimage_ops
module, 64	module, 93
pipelineX.extras.hooks.mlflow	pipelineX.extras.ops.sklearn_ops
module, 58	module, 93
pipelineX.extras.hooks.mlflow.mlflow_art	pipelineX.extras.transformers
module, 58	module, 95
pipelineX.extras.hooks.mlflow.mlflow_base	pipelineX.extras.transformers.mlflow
module, 59	module, 95
pipelineX.extras.hooks.mlflow.mlflow_cat	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 60	module, 95
pipelineX.extras.hooks.mlflow.mlflow_data	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 61	module, 96
pipelineX.extras.hooks.mlflow.mlflow_env	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 61	module, 100
pipelineX.extras.hooks.mlflow.mlflow_time	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 62	module, 96
pipelineX.extras.hooks.mlflow.mlflow_util	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 63	module, 96
pipelineX.extras.ops	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 65	module, 96
pipelineX.extras.ops.allennlp_ops	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 70	module, 96
pipelineX.extras.ops argparse_ops	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 70	module, 96
pipelineX.extras.ops.ignite	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 65	module, 98
pipelineX.extras.ops.ignite.declaratives	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 65	module, 100
pipelineX.extras.ops.ignite.declarativesp	pipelineX.extras.transformers.mlflow.mlflow_io_time
module, 65	module, 100
pipelineX.extras.ops.ignite.handlers	pipelineX.extras.transformers.mlflow.mlflow_io_time
	pipelineX.extras.transformers.mlflow.mlflow_io_time

module, 100
 pipeline.pipeline
 module, 101
 pipeline.pipeline.pipeline
 module, 101
 pipeline.pipeline.sub_pipeline
 module, 102
 pipeline.utils
 module, 103
 Pool1dMixIn (class in *pipeline.extras.ops.pytorch_ops*), 82
 Pool2dMixIn (class in *pipeline.extras.ops.pytorch_ops*), 83
 Pool3dMixIn (class in *pipeline.extras.ops.pytorch_ops*), 83
 project_name (*pipeline.framework.context.flexible_context.FlexibleContext* attribute), 96
 project_version (*pipeline.framework.context.flexible_context.FlexibleContext* attribute), 96
R
 request_coroutine() (in module *pipeline.extras.datasets.http.async_api_dataset*), 47
 requests_coroutine() (in module *pipeline.extras.datasets.http.async_api_dataset*), 47
 reset() (*pipeline.extras.ops.ignite.metrics.cohen_kappa_score.CohenKappaScore* method), 68
 reset() (*pipeline.extras.ops.ignite.metrics.fbeta_score.FbetaScore* method), 69
 reverse_channel() (in module *pipeline.extras.ops.numpy_ops*), 70
 ReverseChannel (class in *pipeline.extras.ops.numpy_ops*), 70
 row_vector_to_square_matrix() (in module *pipeline.extras.ops.pandas_ops*), 80
 run() (*pipeline.framework.context.flexible_parameters_context.FlexibleParametersContext* method), 97
 run() (*pipeline.framework.context.flexible_run_context.OnlyMissingOptionContext* method), 98
 run() (*pipeline.framework.context.flexible_run_context.StringRunOptionContext* method), 99
 running_parallel() (in module *pipeline.extras.hooks.mlflow.mlflow_catalog_logger*), 61
S
 save() (*pipeline.extras.transformers.mlflow.mlflow_io_time_logger.MlflowTimeLogger* method), 95
 SavePipelineJsonContext (class in *pipeline.framework.context.save_pipeline_json_context*), 100
 Scale (class in *pipeline.extras.ops.shap_ops*), 92
 scale() (in module *pipeline.extras.datasets.pillow.images_dataset*), 53
 SeabornPairPlotDataSet (class in *pipeline.extras.datasets.seaborn.seaborn_pairplot*), 56
 setup_conv_params() (in module *pipeline.extras.ops.pytorch_ops*), 92
 SkimageMarkBoundaries (class in *pipeline.extras.ops.skimage_ops*), 93
 SkimageSegmentationDictToDict (class in *pipeline.extras.ops.skimage_ops*), 93
 SkimageSegmentationFelzenszwalb (class in *pipeline.extras.ops.skimage_ops*), 93
 SkimageSegmentationQuickshift (class in *pipeline.extras.ops.skimage_ops*), 93
 SkimageSegmentationSlic (class in *pipeline.extras.ops.skimage_ops*), 93
 SkimageSegmentationWatershed (class in *pipeline.extras.ops.skimage_ops*), 93
 SrMap (class in *pipeline.extras.ops.pandas_ops*), 79
 StatModule (class in *pipeline.extras.ops.pytorch_ops*), 83
 step_binary() (in module *pipeline.extras.ops.pytorch_ops*), 92
 StepBinary (class in *pipeline.extras.ops.pytorch_ops*), 83
 StringRunOptionContext (class in *pipeline.framework.context.flexible_run_context*), 99
 SubPipeline (class in *pipeline.pipeline.sub_pipeline*), 102
 sum_up() (in module *pipeline.extras.ops.opencv_ops*), 73
T
 tensor_max() (in module *pipeline.extras.ops.pytorch_ops*), 92
 tensor_min() (in module *pipeline.extras.ops.pytorch_ops*), 92
 OnlyMissingOptionContext (class in *pipeline.framework.context.flexible_run_context*), 98
 TensorAvgPool1d (class in *pipeline.extras.ops.pytorch_ops*), 83
 TensorAvgPool2d (class in *pipeline.extras.ops.pytorch_ops*), 83
 TensorAvgPool3d (class in *pipeline.extras.ops.pytorch_ops*), 84
 TensorClamp (class in *pipeline.extras.ops.pytorch_ops*), 84
 TensorClampMin (class in *pipeline.extras.ops.pytorch_ops*), 84
 TensorConstantLinear (class in *pipeline.extras.ops.pytorch_ops*), 85

TensorConv1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 85	in	TensorMaxPool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 89	in
TensorConv2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 85	in	TensorMean	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 89	in
TensorConv3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 85	in	TensorMin	(class in <i>pipelinex.extras.ops.pytorch_ops</i>), 89	
TensorCumsum	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 85	in	TensorNearestPad	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 90	in
TensorExp	(class in <i>pipelinex.extras.ops.pytorch_ops</i>), 86		TensorProba	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 90	in
TensorFlatten	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 86	in	TensorRange	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 90	in
TensorForward	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 86	in	TensorSkip	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 90	in
TensorGlobalAvgPool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 86	in	TensorSlice	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 91	in
TensorGlobalAvgPool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 86	in	TensorSqueeze	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 91	in
TensorGlobalAvgPool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	TensorSum	(class in <i>pipelinex.extras.ops.pytorch_ops</i>), 91	
TensorGlobalMaxPool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	TensorUnsqueeze	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 92	in
TensorGlobalMaxPool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	TimeLimit	(class <i>pipelinex.extras.ops.ignite.handlers.time_limit</i>), 68	in
TensorGlobalMaxPool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	to_array()	(in <i>pipelinex.extras.ops.pytorch_ops</i>), 92	module
TensorGlobalMinPool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	to_channel_first_arr()	(in <i>pipelinex.extras.ops.numpy_ops</i>), 70	module
TensorGlobalMinPool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	to_channel_first_tensor()	(in <i>pipelinex.extras.ops.pytorch_ops</i>), 92	module
TensorGlobalMinPool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	to_channel_last_arr()	(in <i>pipelinex.extras.ops.numpy_ops</i>), 70	module
TensorGlobalRangePool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	to_channel_last_tensor()	(in <i>pipelinex.extras.ops.pytorch_ops</i>), 92	module
TensorGlobalRangePool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	ToPipeline	(class in <i>pipelinex.hatch_dict.hatch_dict</i>), 101	
TensorGlobalRangePool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 87	in	total_seconds_to_datetime()	(in <i>pipelinex.extras.decorators.pandas_decorators</i>), 58	module
TensorGlobalSumPool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 88	in	training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleAvg</i> attribute), 80	
TensorGlobalSumPool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 88	in	training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleBottleneck2d</i> attribute), 81	
TensorGlobalSumPool3d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 88	in	training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleConcat</i> attribute), 81	
TensorIdentity	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 88	in	training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleConcatSkip</i> attribute), 81	
TensorLog	(class in <i>pipelinex.extras.ops.pytorch_ops</i>), 88		training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleConvWrap</i> attribute), 81	
TensorMax	(class in <i>pipelinex.extras.ops.pytorch_ops</i>), 88		training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleListMerge</i> attribute), 81	
TensorMaxPool1d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 89	in	training	(<i>pipelinex.extras.ops.pytorch_ops.ModuleProd</i> attribute), 82	
TensorMaxPool2d	(class <i>pipelinex.extras.ops.pytorch_ops</i>), 89	in			


```

training(pipeline.extras.ops.pytorch_ops.TensorGlobalMinPool2d
attribute), 87
update() (pipeline.extras.ops.ignite.metrics.cohen_kappa_score.Cohen
training(pipeline.extras.ops.pytorch_ops.TensorGlobalMinPool3d
method), 68

```

`update()` (*pipelinex.extras.ops.ignite.metrics.fbeta_score.FbetaScore*
method), [69](#)
`update_time_dict()`
(*pipelinex.extras.hooks.mlflow.mlflow_time_logger.MLflowTimeLoggerHook*
method), [63](#)

Z

`ZeroToZeroTransformer` (*class* *in*
pipelinex.extras.ops.sklearn_ops), [94](#)