Kaishi

Release 0.0.1

KUNGFU.AI

Mar 29, 2020

CONTENTS

Kaishi is a toolkit from KUNGFU.AI used to accelerate the initial phases of exploratory data analysis, as well as to enable rapid dataset preparation for downstream tasks.

More simply, Kaishi helps you automate steps to get from a raw dataset (in the form of a directory of files) to something that's usable for machine learning (or any other task you may need a clean dataset for).

Examples of common operations include:

- Filtering duplicate files
- Standardizing image sizes
- Detecting similar data that may not add value to machine learning tasks
- Deduplication of table rows
- Many more...

CHAPTER

ONE

API REFERENCE

This page contains auto-generated API reference documentation[1].

1.1 kaishi

1.1.1 Subpackages

kaishi.core

Core module for Kaishi datasets.

This module contains classes, functions, etc. that are useful elsewhere in the project, as well as dataset definitions that are agnostic to the data type. In particular, the *kaishi.core.dataset.FileDataset* class can be used to operate on any directory of files, regardless of the dataset. Specific data type modules (e.g. *kaishi.image*) also inherit functionality from the core module.

Subpackages

kaishi.core.filters

Pipeline components that filter data points based on user-defined criteria.

Submodules

kaishi.core.filters.by_label

Class definition for filter by label.

[1] Created with sphinx-autoapi

Module Contents

class kaishi.core.filters.by_label.**FilterByLabel**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter each element of a dataset by a specified label.

__call__(*self*, *dataset*)

configure(*self*, *label_to_filter=None*)
Specify the label to filter.

Parameters **label_to_filter** (*str*) – data elements with this label will be filtered

kaishi.core.filters.by_regex

Class definition for filter by regex.

Module Contents

class kaishi.core.filters.by_regex.**FilterByRegex**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter data elements with a filename matching a specified regex.

__call__(*self*, *dataset*)

configure(*self*, *pattern='/(?=a)b/'*)
Configure the regex pattern to match (default does not filter).

Parameters **pattern** (*str*) – pattern to filter by

kaishi.core.filters.duplicate_files

Class definition for filtering duplicate files.

Module Contents

class kaishi.core.filters.duplicate_files.**FilterDuplicateFiles**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter duplicate files, detected via hashing.

__call__(*self*, *dataset*)

kaishi.core.filters.subsample

Class definition for subsampling filter.

Module Contents

class kaishi.core.filters.subsample.**FilterSubsample**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter by subsampling.

__call__(*self*, *dataset*)

configure(*self*, *N=None*, *seed=None*)
Configuration options for subsample filter.

Parameters

- **N** (*int*) – number of data points to keep
- **seed** (*int*) – random seed for reproducibility

kaishi.core.labelers

Pipeline components that label data points based on user-defined criteria.

Submodules

kaishi.core.labelers.validation_and_test

Class definition for validation and test labeler.

Module Contents

class kaishi.core.labelers.validation_and_test.**LabelerValidationAndTest**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Assign validation and/or test data labels.

__call__(*self*, *dataset*)

configure(*self*, *val_frac: float = 0.2*, *test_frac: float = 0.0*, *seed=None*)
Configure the labeler (note: any nonlabeled data point is automatically assigned a "TRAIN" label).

Parameters

- **val_frac** (*float*) – fraction of data points to assign "VALIDATE" label
- **test_frac** (*float*) – fraction of data points to assign "TEST" label
- **seed** (*int*) – random seed for reproducibility

Submodules

kaishi.core.dataset

Primary interface to the image tool kit.

Module Contents

class kaishi.core.dataset.**FileDataset**
Factory for generic file dataset objects.

kaishi.core.file

Class definition for reading/writing files of various types.

Module Contents

class kaishi.core.file.**File**(*basedir: str*, *relpath: str*, *filename: str*)
Class with common methods and members to work with files.

__repr__(*self*)

__str__(*self*)

compute_hash(*self*)
Compute the hash of the file.

Returns hash value

has_label(*self*, *label_to_check: str*)
Check if file has a specific label.

Parameters **label_to_check** (*str*) – label to look for

Returns flag indicating if label is present in the file

Return type bool

add_label(*self*, *label_to_add: str*)
Add a label to a file object.

Parameters **label_to_add** (*str*) – label to append to the file's labels

remove_label(*self*, *label_to_remove: str*)
Remove a label from a file object. If the label is not found, this method does nothing.

Parameters **label_to_remove** (*str*) – label to remove from the file

kaishi.core.file_group

Class definition for reading/writing files of various types.

Module Contents

class kaishi.core.file_group.**FileGroup**(*recursive: bool*)
Class for reading and performing general operations on groups of files.

__getitem__(*self*, *key*)
Get a specific file object.

load_dir(*self*, *source: str*, *file_initializer*, *recursive: bool*)
Read file names in a directory

Parameters

- **source** (*str*) – Directory to load from
- **file_initializer** (kaishi file initializer class (e.g. *kaishi.core.file.File*)) – Data file calss to initialize each file with

get_pipeline_options(*self*)
Returns available pipeline options for this dataset.

Returns list of uninitialized pipeline component objects

Return type list

configure_pipeline(*self*, *choices: list = None*, *verbose: bool = False*)
Configures the sequence of components in the data processing pipeline.

Parameters

- **choices** (*list*) – list of pipeline choices
- **verbose** (*bool*) – flag to indicate verbosity

file_report(*self*, *max_file_entries=16*, *max_filter_entries=10*)
Show a report of valid and invalid data.

Parameters

- **max_file_entries** (*int*) – max number of entries to print of file list
- **max_filter_entries** (*int*) – max number of entries to print per filter category (e.g. duplicates, similar, etc.)

run_pipeline(*self*, *verbose: bool = False*)
Run the pipeline as configured.

Parameters **verbose** (*bool*) – flag to indicate verbosity

kaishi.core.labels

Enumeration definition for labels.

Module Contents

class kaishi.core.labels.**Labels**
Bases: enum.Enum

Enumeration of possible data labels.

RECTIFIED = 0

ROTATED_RIGHT = 1

ROTATED_LEFT = 2

UPSIDE_DOWN = 3

DOCUMENT = 4

STRETCHED = 5

GRAYSCALE = 6

TRAIN = 7

VALIDATE = 8

TEST = 9

kaishi.core.misc

Miscellaneous helper functions.

Module Contents

kaishi.core.misc.**load_files_by_walk**(*dir_name_raw: str*, *file_initializer*, *recursive: bool = False*)
Load files from a directory with an option to recurse.

Parameters

- **dir_name_raw** (*str*) – Directory to load file structure from
- **file_initializer** (kaishi file initializer class (e.g. *kaishi.core.file.File*)) – Data file class to initialize each file with
- **recursive** (*bool*) – Option to load recursively, defaults to False

Returns canonical directory name, list of subdirectories, and list of initialized files

Return type str, list, and list

kaishi.core.misc.**trim_list_by_inds**(*list_to_trim: list*, *indices: list*)
Trim a list given an unordered list of indices.

Parameters

- **list_to_trim** (*list*) – list to remove elements from

- **indices** (*list*) – indices of list items to remove

Returns new list, trimmed items

Return type list, list

kaishi.core.misc.**find_duplicate_inds**(*list_with_duplicates: list*)
Find indices of duplicates in a list.

Parameters **list_with_duplicates** (*list*) – list containing duplicate items

Returns list of duplicate indices, list of unique items (parents of duplicates)

Return type list and list

kaishi.core.misc.**find_similar_by_value**(*list_of_values: list*, *difference_threshold*)
Find near duplicates based on similar reference value.

Parameters

- **list_of_values** (*list*) – list of values to compare
- **difference_threshold** – differences above this threshold will be identified for removal

Returns list of similar indices, list of unique items (parents of similar items)

Return type list and list

kaishi.core.misc.**md5sum**(*filename: str*)
Compute the md5sum of a file.

Parameters **filename** (*str*) – name of file to compute hash of

Returns hash value

class kaishi.core.misc.**CollapseChildren**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Restructure potentially multi-layer file tree into a single parent/child layer.

__call__(*self*, *dataset*)

kaishi.core.misc.**is_valid_label**(*label_str: str*, *label_enum*)
Check if a label is contained in an enum.

Parameters **label_str** (*str*) – string defining the label

Returns flag indicating if label is valid

Return type bool

kaishi.core.pipeline

Class definition for a pipeline object.

Module Contents

class kaishi.core.pipeline.**Pipeline**
Base class for a generic pipeline object.

__call__(*self*, *dataset*, *verbose: bool = False*)
Run the full pipeline as configured.

Parameters

- **dataset** (initiailized kaishi dataset class (e.g. :class *kaishi.image.dataset.Dataset*)) – dataset to perform pipeline operations on
- **verbose** (*bool*) – flag to indicate verbosity

__repr__(*self*)
Print pipeline overview.

__str__(*self*)

_get_configs_for_component(*self*, *initialized_component*)
Get args and values of them from an initialized component.

Parameters **initialized_component** (initialized pipeline component (has to inherit from *kaishi.core.pipeline_component.PipelineComponent*)) – pipeline to get configurable arguments for

Returns dictionary with argument name keys and their values as contents

Return type dict

add_component(*self*, *component*)
Add a method to be called as a pipeline step.

Parameters **component** – component to add to the pipeline

remove_component(*self*, *index*)
Remove a pipeline method by index.

Parameters **index** (*int*) – index to remove

reset(*self*)
Reset the pipeline by removing all components.

kaishi.core.pipeline_component

Class definition for pipeline components.

Module Contents

class kaishi.core.pipeline_component.**PipelineComponent**
Base class for pipeline components.

__str__(*self*)

__repr__(*self*)

configure(*self*)
Method to configure via named arguments. Defaults to no configurations, unless inherited and overridden.

applies_to(*self*, *target_criteria*)
Limit data files that the component applies to via regex.

Parameters **target_criteria** – list or string containing a regex to denote files that this component applies to

get_target_indexes(*self*, *dataset*)
Get target indexes of a dataset based on criteria set using the *applies_to()* method

Parameters **dataset** (initialized kaishi dataset object (e.g. `kaishi.core.dataset.FileDatset`)) – dataset to inspect

Returns list of indexes

Return type list of int

_is_valid_target_int(*self*, *target*)
Check if an index is a valid integer type.

Parameters **target** (*int (or similar, e.g. np.int32)*) – target to verify

Returns flag indicating if the target is a valid integer type

Return type bool

_is_valid_target_str(*self*, *target*)
Check if an index is a valid string type.

Parameters **target** (*int (or similar)*) – target to verify

Returns flag indicating if the target is a valid string type

Return type bool

kaishi.core.printing

Definitions for print helper utilities.

Module Contents

kaishi.core.printing.**should_print_row**(*i: int*, *max_entries: int*, *num_entries: int*)
Make decision to print row or not based on max_rows.

Parameters

- **i** (*int*) – index of row
- **max_entries** (*int*) – max number of entries for the table
- **num_entries** (*int*) – number of possible entries (the full list)

Returns 0 if should not print, 1 if should print, 2 if should print ellipsis ("...")

Return type int

kaishi.image

Kaishi framework for image datasets.

Subpackages

kaishi.image.filters

Pipeline components that filter data points based on user-defined criteria, specifically for image datasets.

Submodules

kaishi.image.filters.invalid_file_extensions

Class definition for filtering by invalid image file extensions.

Module Contents

kaishi.image.filters.invalid_file_extensions.**VALID_EXT = ['.bmp', '.dib', '.jpeg', '.jpg',**

class kaishi.image.filters.invalid_file_extensions.**FilterInvalidFileExtensions**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter files without a valid image file extension, where valid extensions are defined in the *configure* method.

__call__(*self*, *dataset*)
Perform the filter operation on *dataset*.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – dataset to perform filter operation on

configure(*self*, *valid_extensions=VALID_EXT*)
Configure filter with the valid extensions defined.

Parameters **valid_extensions** (*list[str]*) – list of valid extensions (each should start with ".")

kaishi.image.filters.invalid_image_headers

Class definition for filtering files with invalid image headers.

Module Contents

class kaishi.image.filters.invalid_image_headers.**FilterInvalidImageHeaders**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter image files that have invalid or nonexistent headers.

__call__(*self*, *dataset*)
Perform filter operation on a kaishi image dataset.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – image dataset to perform filter operation on

kaishi.image.filters.similar

Class definition for filtering similar images in a dataset.

Module Contents

class kaishi.image.filters.similar.**FilterSimilar**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter near duplicate files, detected via perceptual hashing (using the *imagehash* library).

__call__(*self*, *dataset*)
Perform filter operation on a specified dataset.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – dataset to perform operation on

configure(*self*, *perceptual_hash_threshold=3*)
Configure the filter with a perceptual hash threshold.

Parameters **perceptual_hash_threshold** (*int or float*) – threshold for determining whether or not images are similar (> are deemed not similar)

kaishi.image.labelers

Pipeline components that label data points based on user-defined criteria, specifically for image datasets.

Submodules

kaishi.image.labelers.generic_convnet

Class definition for generic convnet labeler.

Module Contents

class kaishi.image.labelers.generic_convnet.**LabelerGenericConvnet**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Use pre-trained ConvNet to predict image labels (e.g. stretched, rotated, etc.).

This labeler uses a default configured *kaishi.image.model.Model* object where the output layer is assumed to have 6 values ranging 0 to 1, where the labels are [DOCUMENT, RECTIFIED, ROTATED_RIGHT, ROTATED_LEFT, UPSIDE_DOWN, STRETCHED].

__call__(*self*, *dataset*)
Perform the labeling operation on an image dataset.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – kaishi image dataset

kaishi.image.transforms

Pipeline components that transform data points based on user-defined criteria, specifically for image datasets.

Submodules

kaishi.image.transforms.fix_rotation

Class definition for fixing image rotation.

Module Contents

class kaishi.image.transforms.fix_rotation.**TransformFixRotation**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Fix rotations of each image in a dataset given pre-determined labels (uses the default convnet for labels).

__call__(*self*, *dataset*)
Perform the transformation operation on an image dataset.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – image dataset to perform operation on

kaishi.image.transforms.limit_dimensions

Class definition for limiting the max dimension of each image in an image dataset.

Module Contents

class kaishi.image.transforms.limit_dimensions.**TransformLimitDimensions**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Transform to limit max dimension of each image in a dataset.

__call__(*self*, *dataset*)
Perform operation on a specified dataset.

Parameters **dataset** (kaishi.image.dataset.ImageDatset) – image dataset to perform operation on

configure(*self*, *max_dimension=None*, *max_width=None*, *max_height=None*)
Configure the component. Any combination of these parameters can be defined or not, where smallest specified max value in each case is the assumed value (e.g. if *max_width* is 300 but *max_dimension* is 200, the maximum width is effectively 200).

Parameters

- **max_dimension** (*int*) – maximum dimension for each image (either width or height)
- **max_width** (*int*) – maximum width for each image
- **max_height** (*int*) – maximum height for each image

kaishi.image.transforms.to_grayscale

Class definition for transforming images to grayscale.

Module Contents

class kaishi.image.transforms.to_grayscale.**TransformToGrayscale**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Transform images in a dataset to grayscale.

__call__(*self*, *dataset*)
Perform operation on a given dataset.

Parameters **dataset** (*kaishi.image.dataset.ImageDataset*) – image dataset with images to convert

Submodules

kaishi.image.dataset

Definition for kaishi image datasets.

Module Contents

class kaishi.image.dataset.**ImageDataset**
Factory for image dataset objects.

kaishi.image.file

Definition for image files.

Module Contents

kaishi.image.file.**THUMBNAIL_SIZE = [64, 64]**

kaishi.image.file.**MAX_DIM_FOR_SMALL = 224**

kaishi.image.file.**PATCH_SIZE = [64, 64]**

kaishi.image.file.**RESAMPLE_METHOD**

class kaishi.image.file.**ImageFile**(*basedir: str*, *relpath: str*, *filename: str*)
Bases: *kaishi.core.file.File*

Image file object that inherits from the core file class and adds image-specific attributes and methods.

verify_loaded(*self*)
Verify image and derivatives are loaded (only performs the load if the image is unloaded).

update_derived_images(*self*)
Update images derived from the base image (i.e. thumbnail, small version, and random patch).

rotate (*self*, *ccw_rotation_degrees: int*)
Rotate all instances of image by 'ccw_rotation_degrees'.

Parameters **ccw_rotation_degrees** (*int*) – degrees to rotate the image by

limit_dimensions (*self*, *max_width: int = None*, *max_height: int = None*, *max_dimension: int = None*)
Limit the max dimension of the image and resize accordingly. Any combination of these arguments can be defined, however, if none are defined, this method does nothing.

Parameters

- **max_width** (*int*) – maximum width of the image
- **max_height** (*int*) – maximum height of the image
- **max_dimension** (*int*) – maximum width or height (applies to both)

convert_to_grayscale (*self*)
Convert image to grayscale.

compute_perceptual_hash (*self*, *hashfunc=imagehash.average_hash*)
Calculate perceptual hash (close in value to similar images.

Parameters **hashfunc** (*function*) – function object to be used to calculate the hash value (defualts to *imagehash.average_hash*)

Returns hash value (as computed by *hashfunc*)

kaishi.image.file_group

Definition for groups of image files.

Module Contents

kaishi.image.file_group.**THUMBNAIL_SIZE = [64, 64]**

kaishi.image.file_group.**MAX_DIM_FOR_SMALL = 224**

kaishi.image.file_group.**PATCH_SIZE = [64, 64]**

kaishi.image.file_group.**RESAMPLE_METHOD**

class kaishi.image.file_group.**ImageFileGroup** (*source: str*, *recursive: bool*)
Bases: *kaishi.core.file_group.FileGroup*

Group of image files that inherits from the core file group class.

load_all (*self*)
Load all files in the directory that this class was initialized with.

build_numpy_batches (*self*, *channels_first: bool = True*, *batch_size: int = None*, *image_type: str = 'small_image'*)
Build a tensor from the entire image corpus (or generate batches if specified).

If a batch size is specified, this acts as a generator of batches and returns a list of file objects to manipulate. Otherwise, a single batch of all images is returned in an array format.

Parameters

- **channels_first** (*bool*) – flag indicating channels first (e.g. PyTorch) vs. channels last (e.g. Keras)

- **batch_size** (*int*) – size of each batch (default is *None*, which will return a single batch)
- **image_type** (*str*) – choice of "small_image", "thumbnail", or "patch", indicating which version of each image to use

Returns batch of images (generator if batch size specified)

Return type *numpy.array*

save(*self*, *out_dir: str*)
Save image data set in the current structure while preserving any changes.

Parameters **out_dir** (*str*) – output directory

run_pipeline(*self*, *verbose: bool = False*)
Run the pipeline as configured.

Parameters **verbose** (*bool*) – flag indicating verbosity

report(*self*)
Run a descriptive report (currently just prints the file report).

kaishi.image.generator

Data generator for image datasets.

Module Contents

kaishi.image.generator.**augment_and_label**(*imobj*)
Augment an image with common issues and return the modified image + label vector.

Labels at output layer (probabilities, no softmax): [DOCUMENT, RECTIFIED, ROTATED_RIGHT, ROTATED_LEFT, UPSIDE_DOWN, STRETCHING]

Parameters **imobj** (*kaishi.image.file.ImageFile*) – image object to randomly augment and label

Returns augmented image and label vector applied

Return type *kaishi.image.file.ImageFile* and *numpy.array*

kaishi.image.generator.**train_generator**(*self*, *batch_size: int = 32*, *string_to_match: str = None*)
Generator for training the data labeler. Operates on a *kaishi.image.dataset.ImageDataset* object.

Parameters

- **self** (kaishi.image.dataset.ImageDatset) – image dataset
- **batch_size** (*int*) – batch size for generated data
- **string_to_match** (*str*) – string to match (ignores files without this string in the relative path)

Returns batch arrays and label vectors

Return type numpy.array and list

kaishi.image.generator.**generate_validation_data**(*self*, *n_examples: int = 400*, *string_to_match: str = None*)
Generate a reproducibly random validation data set.

Parameters

- **n_examples** (*int*) – number of examples in the validation set
- **string_to_match** (*str*) – string to match (ignores files without this string in the relative path)

Returns stacked training examples (first dimension is batch) and stacked labels

Return type *numpy.array* and *numpy.array*

kaishi.image.model

Definition for PyTorch model abstraction.

Module Contents

class kaishi.image.model.**Model**(*n_classes: int = 6*, *model_arch: str = 'resnet18'*)

Abstraction for working with PyTorch models.

vgg16_bn(*self*, *n_classes: int*)

Basic VGG16 model with variable number of output classes.

Parameters **n_classes** (*int*) – number of classes at output layer

Returns PyTorch VGG16 model object with batch normalization

Return type *torchvision.models.vgg16_bn*

resnet18(*self*, *n_classes: int*)

Basic ResNet18 model with specified number of output classes.

Parameters **n_classes** (*int*) – number of classes at the output layer

Returns PyTorch ResNet18 model object

Return type *torchvision.models.resnet18*

resnet50(*self*, *n_classes: int*)

Basic ResNet50 model with specified number of output classes.

Parameters **n_classes** (*int*) – number of classes at the output layer

Returns PyTorch ResNet50 model object

Return type *torchvision.models.resnet50*

predict(*self*, *numpy_array*)

Make predictions from a numpy array, where dimensions are (batch, channel, x, y).

Parameters **numpy_array** (*numpy.array*) – input array to predict

Returns predictions, where the dimensions are (batch, output)

Return type *numpy.array*

kaishi.image.ops

Definitions for common operations on images.

Module Contents

kaishi.image.ops.**extract_patch**(*im*, *patch_size*)
Extract a center cropped patch of size 'patch_size' (2-element tuple).

Parameters

- **im** (*PIL image object*) – input image
- **patch_size** (*tuple, array, or similar*) – size of patch

Returns center-cropped patch

Return type PIL image object

kaishi.image.ops.**make_small**(*im*, *max_dim: int = 224*, *resample_method=Image.NEAREST*)
Make a small version of an image while maintaining aspect ratio.

Parameters

- **im** (*PIL image object*) – input image
- **max_dim** (*int*) – maximum dimension of resized image (x or y)
- **resample_method** (*PIL resample method*) – method for resampling the image

Returns resized image

Return type PIL image object

kaishi.image.ops.**add_jpeg_compression**(*im*, *quality_level: int = 30*)
Apply JPEG compression to an image with a given quality level.

Parameters

- **im** (*PIL image object*) – input image
- **quality_level** (*int*) – JPEG qualit level, where: 0 < value <= 100

Returns compressed image

Return type PIL image object

kaishi.image.ops.**add_rotation**(*im*, *ccw_rotation_degrees: int = 90*)
Rotate an image CCW by *ccw_rotation_degrees* degrees.

Parameters

- **im** (*PIL image object*) – input image
- **ccw_rotation_degrees** (*int*) – number of degrees to rotate counter-clockwise

Returns rotated image

Return type PIL image object

kaishi.image.ops.**add_stretching**(*im*, *w_percent_additional*, *h_percent_additional*)
Stretch an image by the specified percentages.

Parameters

- **im** (*PIL image object*) – input image

- **w_percent_additional** (*int or float greater than 0*) – amount of width stretching to add (0 maintains the same size, 100 doubles the size)
- **h_percent_additional** (*int or float greater than 0*) – amount of height stretching to add (0 maintains the same size, 100 doubles the size)

Returns stretched image

Return type PIL image object

kaishi.image.ops.**add_poisson_noise**(*im*, *param: float = 1.0*, *rescale: bool = True*)

Add Poisson noise to image, where (poisson noise * *param*) is the final noise function.

See http://kmdouglass.github.io/posts/modeling-noise-for-image-simulations for more info. If *rescale* is set to True, the image will be rescaled after noise is added. Otherwise, the noise will saturate.

Parameters

- **im** (*PIL image object*) – input image
- **param** (*float*) – noise parameter
- **rescale** (*bool*) – flag indicating whether or not to rescale the image after adding noise (maintaining original image extrema)

Returns image with Poisson noise added

Return type PIL image object

kaishi.image.util

Image utilities and helper functions.

Module Contents

kaishi.image.util.**swap_channel_dimension**(*tensor*)

Swap between channels_first and channels_last.

If 'tensor' has 4 elements, it's assumed to be the shape vector. Otherwise, it's assumed that it's the actual tensor. Returns the edited shape vector or tensor.

Parameters **tensor** (*numpy.array*) – shape vector or tensor to have channel dimensions swapped

Returns altered input with the channel dimensions swapped

Return type *numpy.array*

kaishi.image.util.**validate_image_header**(*filename: str*)

Validate that an image has a valid header.

Returns True if valid, False if invalid.

Parameters **filename** (*str*) – name of file to analyze

Returns flag indicating whether header is valid (by using *imghdr.what()*)

Return type bool

kaishi.image.util.**get_batch_dimensions**(*self*, *batch_size: int*, *channels_first: bool = True*, *image_type: str = 'small_image'*)

Get dimensions of the batch tensor. Note that the 'batch_size' argument can be the full data set.

Parameters

- **self** (*kaishi.image.dataset.ImageDataset*) – image dataset object
- **batch_size** (*int*) – batch size
- **channels_first** (*bool*) – flag indicating whether channels are first or last dimension in each image
- **image_type** (*str*) – one of "small_image", "thumbnail", or "patch"

Returns batch dimesions (4D tuple)

Return type tuple

kaishi.tabular

Kaishi framework for tabular datasets.

Subpackages

kaishi.tabular.aggregators

Pipeline components that aggregate data based on user-defined criteria, specifically for tabular datasets.

Submodules

kaishi.tabular.aggregators.concatenate_dataframes

Class definition for concatenating tabular data files.

Module Contents

class kaishi.tabular.aggregators.concatenate_dataframes.**AggregatorConcatenateDataframes**

Bases: *kaishi.core.pipeline_component.PipelineComponent*

Concatenate all data frames.

__call__(*self*, *dataset*)

Perform concatenation on a given dataset (all files must have the same schema).

Parameters **dataset** (*kaishi.tabular.dataset.TabularDataset*) – tabular dataset to perform operation on

kaishi.tabular.filters

Pipeline components that filter data points based on user-defined criteria, specifically for tabular datasets.

Submodules

kaishi.tabular.filters.duplicate_rows_after_concatenation

Class definition for filtering duplicate rows after concatenation.

Module Contents

class kaishi.tabular.filters.duplicate_rows_after_concatenation.**FilterDuplicateRowsAfterCo**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter duplicate rows in the concatenated dataframe (dataset will be concatenated if it hasn't been already).

__call__(*self*, *dataset*)
Perform filter on a given tabular dataset.

Parameters **dataset** (*kaishi.tabular.dataset.TabularDataset*) – tabular dataset to perform operation on

kaishi.tabular.filters.duplicate_rows_each_dataframe

Class definition for filtering duplicate rows in each dataframe.

Module Contents

class kaishi.tabular.filters.duplicate_rows_each_dataframe.**FilterDuplicateRowsEachDataframe**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter duplicate rows in each dataframe of a tabular dataset.

__call__(*self*, *dataset*)
Perform the filter operation on a given tabular dataset.

Parameters **dataset** (*kaishi.tabular.dataset.TabularDataset*) – dataset to perform operation on

kaishi.tabular.filters.invalid_file_extensions

Class definition for filtering invalid tabular file extensions.

Module Contents

kaishi.tabular.filters.invalid_file_extensions.**VALID_EXT = ['.json', '.jsonl', '.json.gz',**

class kaishi.tabular.filters.invalid_file_extensions.**FilterInvalidFileExtensions**
Bases: *kaishi.core.pipeline_component.PipelineComponent*

Filter file list if non-tabular extensions exist.

__call__(*self*, *dataset*)
Perform operation on a tabular dataset.

Parameters **dataset** (*kaishi.tabular.dataset.TabularDataset*) – dataset to perform file extension filter on

configure(*self*, *valid_extensions=VALID_EXT*)
Configure the file extension filter (default list defined in *VALID_EXT*).

Parameters **valid_extensions** (*list[str]*) – list of file extensions that are valid (each must begin with ".")

Submodules

kaishi.tabular.dataset

Class definition for data exploration utilities for tabular data (csv files, database tables).

Module Contents

class kaishi.tabular.dataset.**TabularDataset**
Object for analyzing and manipulating tabular datasets.

kaishi.tabular.file

Class definition for tabular data files.

Module Contents

class kaishi.tabular.file.**TabularFile**(*basedir: str*, *relpath: str*, *filename: str*)
Bases: *kaishi.core.file.File*

Class for tabular data file-specific attributes and methods.

_has_csv_file_ext(*self*)
Check if file is variant of .csv

Returns flag indicating if file is valid

Return type bool

_has_json_file_ext(*self*)
Check if file is variant of .json

Returns flag indicating if file is valid

Return type bool

verify_loaded(*self*)
Load the file if supported.

get_summary(*self*)
Create summary for this data frame.

Returns summary dictionary containing common analyses

Return type dict

kaishi.tabular.file_group

Class definition for group of tabular files.

Module Contents

class kaishi.tabular.file_group.**TabularFileGroup**(*source: str*, *recursive: bool*, *use_predefined_pipeline: bool = False*, *out_dir: str = None*)

Bases: *kaishi.core.file_group.FileGroup*

Object containing groups of `kaishi.tabular.file.File` objects with methods to perform common operations on them.

_get_indexes_with_valid_dataframe(*self*)
Get a list of indexes with valid dataframes.

Returns indexes with valid dataframe

Return type list

_get_valid_dataframes(*self*)
Get a list of valid dataframe objects.

Returns valid dataframes

Return type list[`pandas.core.frame.DataFrame`]

save(*self*, *out_dir: str*, *file_format: str = 'csv'*)
Save the processed dataset as individual files or as one file with all the data.

Parameters

- **out_dir** (*str*) – The path of the output directory. If the directory does not exist, it will be created.
- **file_format** (*str*) – The format of output files. Currently only supports "csv".

load_all(*self*)
Load all files from the source directory.

run_pipeline(*self*, *verbose: bool = False*)
Run the pipeline as configured.

Parameters **verbose** (*bool*) – flag indicating verbosity

report(*self*)
Print a report of the dataset in its current state.

Package Contents

class kaishi.tabular.**TabularDataset**
Object for analyzing and manipulating tabular datasets.

CHAPTER

TWO

USER GUIDE

2.1 Installation

Kaishi is hosted on PyPI. To install, simply run:

```
pip install kaishi
```

or, alternatively, if you're a developer/contributor, clone the source from github and then run:

```
pip install -r requirements.txt
pip install .
```

2.2 Pipeline Components

Pipeline components are broken up into several distinct categories, and the classes that define them MUST begin with the correct keyword:

- `Filter*` - removes elements of a dataset
- `Transform*` - changes one or more data objects in some fundamental way
- `Labeler*` - creates labels for data objects without modifying the underlying data
- `Aggregator*` - combines data objects in some way

Look at how other pipeline components are implemented. Feel free to write your own, while following the below rules:

- Inherits from the `PipelineComponent` class
- Has no initialization arguments
- Has a single `__call__` method with a single argument (a dataset object)
- If specific configuration is needed, a method named `self.configure(...)` must be written with named arguments with defaults. `self.configure()` must be called as part of the `__init__(...)` call for configuration to work.
- `self.applies_to_available = True` must be in the `__init__(...)` call if the component takes advantage of the `self.applies_to()` and `self.get_target_indexes()` methods from `kaishi.core.pipeline_component.PipelineComponent` method
- If an artifact (i.e. some result) is created from the operation, as in the case of Aggregators, the artifact should be added to the `dataset.artifacts` dictionary (automatically initialized with any new dataset)

You can then enable usage of the component with your instantiated dataset object, e.g.:

```
from your_definition import YourNewComponent
imd.YourNewComponent = YourNewComponent
imd.configure_pipeline(['YourNewComponent'])
```

CHAPTER

THREE

TUTORIALS

3.1 Image Datasets

Image datasets in this context are directories of files. Kaishi has a lot of built in functionality for interacting with them. While kaishi has many built-in pipeline components that operate on image datasets, a lot of the standard ETL is handled for you in the event you want to add your own custom code (without the ETL hassle).

3.1.1 Initializing datasets

Let's start by downloading a sample dataset to work with. You will need wget installed unless using your own directory of files.

```
import wget
import pickle
from PIL import Image
import tarfile
import os

wget.download("http://www.vision.caltech.edu/Image_Datasets/Caltech101/101_
↪ObjectCategories.tar.gz")
tarfile.open("101_ObjectCategories.tar.gz").extractall()
os.remove("101_ObjectCategories.tar.gz")
```

First, initialize a kaishi image dataset object and print a descriptive report of files.

```
from kaishi.image.dataset import ImageDataset
imd = ImageDataset("101_objectCategories", recursive=True)
imd.file_report()
```

```
Current file list:
+-------+--------------------------+---------------------------------------+--------+
| Index |        File Name         | Children                              | Labels |
+-------+--------------------------+---------------------------------------+--------+
|   0   |  gerenuk/image_0032.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   1   |  gerenuk/image_0026.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   2   |  gerenuk/image_0027.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   3   |  gerenuk/image_0033.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   4   |  gerenuk/image_0019.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   5   |  gerenuk/image_0025.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   6   |  gerenuk/image_0031.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|   7   |  gerenuk/image_0030.jpg  |     {'duplicates': [], 'similar': []} |   []   |
|  ...  |                          |                                       |        |
```

(continues on next page)

(continued from previous page)

```
|  9137 | metronome/image_0015.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9138 | metronome/image_0029.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9139 | metronome/image_0028.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9140 | metronome/image_0014.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9141 | metronome/image_0016.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9142 | metronome/image_0002.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9143 | metronome/image_0003.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  9144 | metronome/image_0017.jpg |     {'duplicates': [], 'similar': []} |    []    |
+-------+--------------------------+---------------------------------------+----------+
Filtered files:
+-----------+---------------+
| File Name | Filter Reason |
+-----------+---------------+
+-----------+---------------+


/Users/mwharton/.miniconda3/envs/kaishi/lib/python3.7/site-packages/kaishi/image/
↪dataset.py:20: UserWarning: No GPU detected, ConvNet prediction tasks will be very␣
↪slow
  warnings.warn("No GPU detected, ConvNet prediction tasks will be very slow")
```

There are almost 10k images in this directory, let's use a subdirectory to keep the dataset small.

```
imd = ImageDataset("101_objectCategories/Faces")
imd.file_report()
```

```
Current file list:
+-------+---------------+---------------------------------------+----------+
| Index |   File Name   | Children                              | Labels   |
+-------+---------------+---------------------------------------+----------+
|   0   | image_0146.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   1   | image_0152.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   2   | image_0185.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   3   | image_0191.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   4   | image_0378.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   5   | image_0344.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   6   | image_0422.jpg |     {'duplicates': [], 'similar': []} |    []    |
|   7   | image_0350.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  ...  |                |                                       |          |
|  427  | image_0349.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  428  | image_0375.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  429  | image_0413.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  430  | image_0407.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  431  | image_0361.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  432  | image_0188.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  433  | image_0177.jpg |     {'duplicates': [], 'similar': []} |    []    |
|  434  | image_0163.jpg |     {'duplicates': [], 'similar': []} |    []    |
+-------+---------------+---------------------------------------+----------+
Filtered files:
+-----------+---------------+
| File Name | Filter Reason |
+-----------+---------------+
+-----------+---------------+
```

3.1.2 Interaction with datasets

Now, let's look at a couple ways to access the images.

Each of these "files" are actually kaishi.image.file.File objects, which have quite a few interesting methods to enable rapid analysis. Each image object is initialized to None by default, but verify_loaded() will load an individual file, whereas load_all() will load all of them. If running a pipeline, load_all() will be called for you.

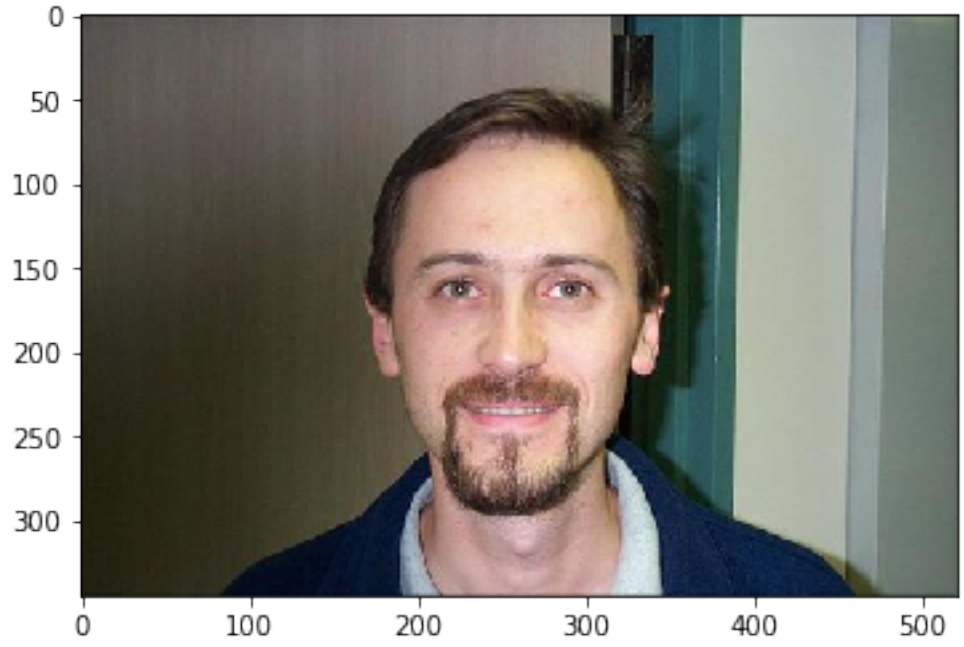
```
imd.files[:10]
```

```
[image_0146.jpg,
 image_0152.jpg,
 image_0185.jpg,
 image_0191.jpg,
 image_0378.jpg,
 image_0344.jpg,
 image_0422.jpg,
 image_0350.jpg,
 image_0387.jpg,
 image_0393.jpg]
```

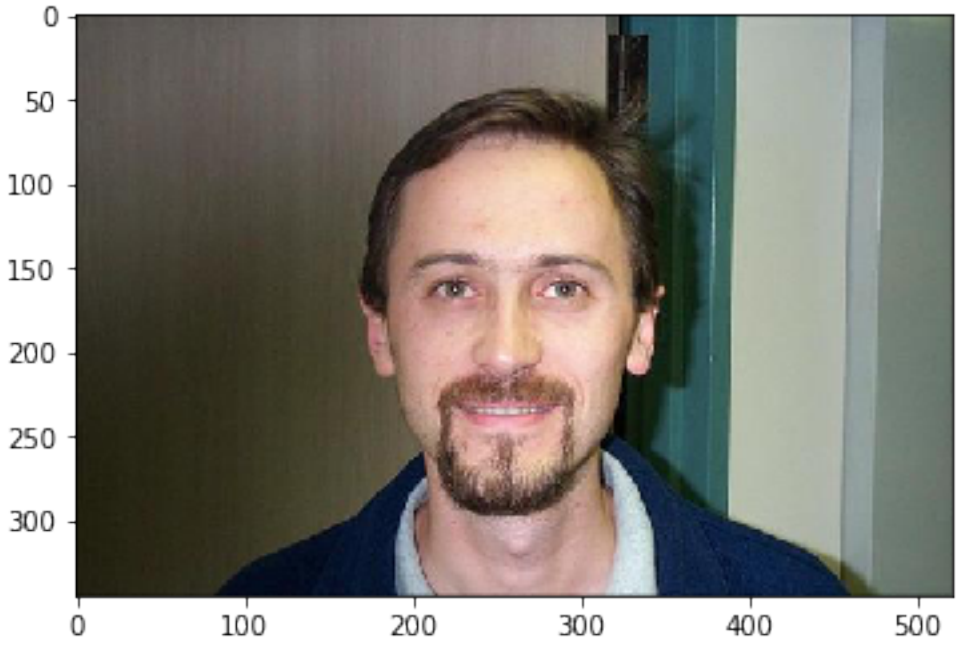
```
print(imd.files[0].image is None)
imd.files[0].verify_loaded()
print(imd.files[0].image is None)
```

```
True
False
```

```
import matplotlib.pyplot as plt
plt.imshow(imd.files[0].image)
plt.show()
plt.imshow(imd["image_0146.jpg"].image)
plt.show()
```



png



png

3.1.3 Image processing pipelines

Let's see the pipeline options.

```
imd.get_pipeline_options()
```

```
['FilterByLabel',
 'FilterByRegex',
 'FilterDuplicateFiles',
 'FilterInvalidFileExtensions',
 'FilterInvalidImageHeaders',
 'FilterSimilar',
 'FilterSubsample',
 'LabelerGenericConvnet',
 'LabelerValidationAndTest',
 'TransformFixRotation',
 'TransformLimitDimensions',
 'TransformToGrayscale']
```

Now we can create, configure, and run a pipeline.

Note: the default regex pattern does not perform any filtering.

```
imd.configure_pipeline(["FilterInvalidImageHeaders", "FilterDuplicateFiles",
↪"FilterByRegex", "TransformLimitDimensions", "TransformToGrayscale"])
print(imd.pipeline)
```

```
Kaishi pipeline:
0: FilterInvalidImageHeaders
1: FilterDuplicateFiles
2: FilterByRegex
     pattern: '/(?=a)b/'
```

(continues on next page)

(continued from previous page)

```
3: TransformLimitDimensions
     max_dimension: None
     max_width: None
     max_height: None
4: TransformToGrayscale
```

```
imd.pipeline.components[2].configure(pattern="image_02.*jpg")
imd.pipeline.components[3].configure(max_dimension=400)
print(imd.pipeline)
```

```
Kaishi pipeline:
0: FilterInvalidImageHeaders
1: FilterDuplicateFiles
2: FilterByRegex
     pattern: 'image_02.*jpg'
3: TransformLimitDimensions
     max_dimension: 400
     max_width: None
     max_height: None
4: TransformToGrayscale
```

```
imd.run_pipeline()
```

Now we can analyze the results.

```
imd.file_report()
```

```
Current file list:
+-------+----------------+---------------------------------------+----------------+
| Index |   File Name    | Children                              |     Labels     |
+-------+----------------+---------------------------------------+----------------+
|   0   | image_0146.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   1   | image_0152.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   2   | image_0185.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   3   | image_0191.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   4   | image_0378.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   5   | image_0344.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   6   | image_0422.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|   7   | image_0350.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  ...  |                |                                       |                |
|  327  | image_0349.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  328  | image_0375.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  329  | image_0413.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  330  | image_0407.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  331  | image_0361.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  332  | image_0188.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  333  | image_0177.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
|  334  | image_0163.jpg |     {'duplicates': [], 'similar': []} | ['GRAYSCALE'] |
+-------+----------------+---------------------------------------+----------------+
Filtered files:
+----------------+---------------+
|   File Name    | Filter Reason |
+----------------+---------------+
| image_0215.jpg |     regex     |
| image_0201.jpg |     regex     |
```

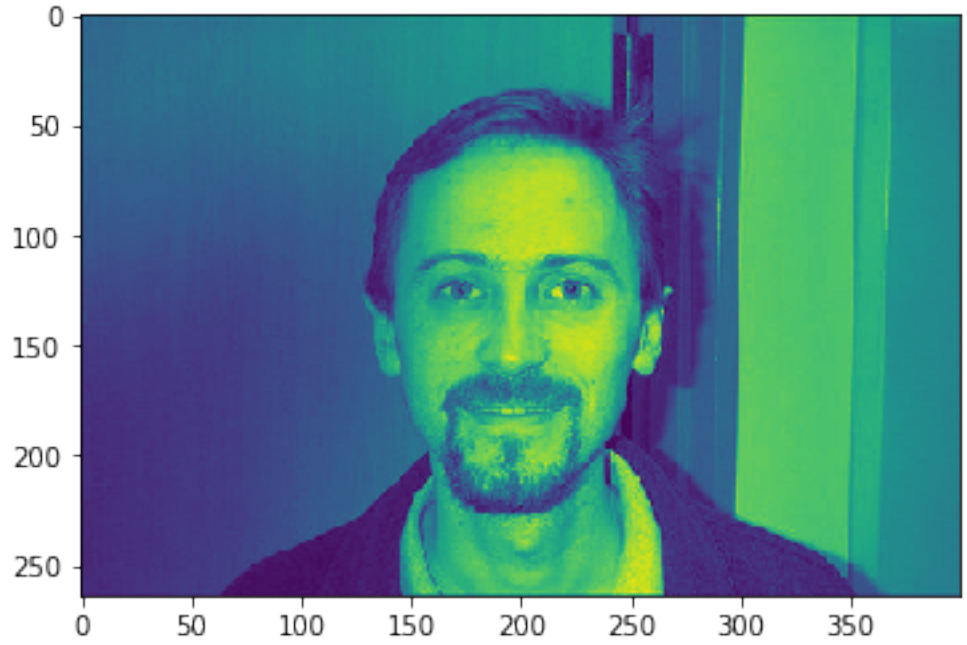
(continues on next page)

(continued from previous page)

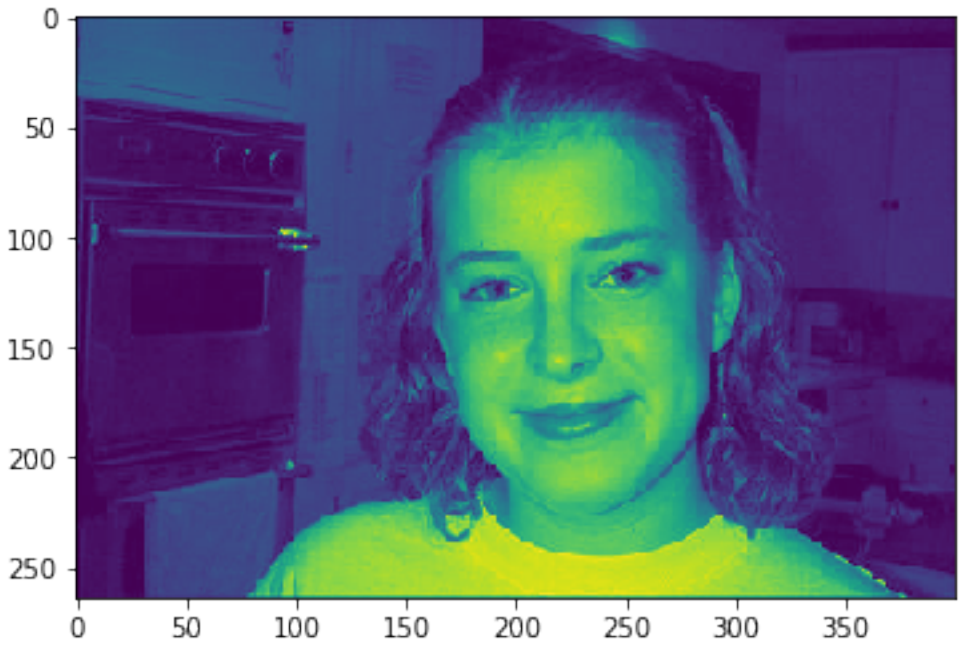
```
| image_0229.jpg |      regex      |
| image_0228.jpg |      regex      |
| image_0200.jpg |      regex      |
|       ...      |                 |
| image_0231.jpg |      regex      |
| image_0225.jpg |      regex      |
| image_0224.jpg |      regex      |
| image_0230.jpg |      regex      |
| image_0218.jpg |      regex      |
+----------------+-----------------+
```

Note that the images have been sized down (max dimension is 400) and are now grayscale (as expected)

```
plt.imshow(imd["image_0146.jpg"].image)
plt.show()
plt.imshow(imd["image_0361.jpg"].image)
plt.show()
```



png



png

3.1.4 Custom pipeline components

What if we wanted to create a custom pipeline component?

Let's create one that quantizes each of the images that passed our previous filter operations.

```
from kaishi.core.pipeline_component import PipelineComponent

# Follow rules specified in the pipeline comonent guide
class TransformByQuantizing(PipelineComponent):
    """Transform that quantizes images."""

    def __init__(self):
        super().__init__()
        self.configure()
        self.applies_to_available = True  # Set this to true if using the "get_target_
↪indexes" method

    def __call__(self, dataset):
        # Trim any files without image extensions
        for i in self.get_target_indexes(dataset):
            dataset.files[i].image = dataset.files[i].image.quantize(colors=self.n_
↪colors).convert("L")
            dataset.files[i].update_derived_images()  # This updates thumbnails/etc.

    def configure(self, n_colors=32):
        self.n_colors = n_colors

imd.TransformByQuantizing = TransformByQuantizing
```

Check to see that it was properly added

```
imd.get_pipeline_options()
```

```
['FilterByLabel',
 'FilterByRegex',
 'FilterDuplicateFiles',
 'FilterInvalidFileExtensions',
 'FilterInvalidImageHeaders',
 'FilterSimilar',
 'FilterSubsample',
 'LabelerGenericConvnet',
 'LabelerValidationAndTest',
 'TransformByQuantizing',
 'TransformFixRotation',
 'TransformLimitDimensions',
 'TransformToGrayscale']
```

```
imd.configure_pipeline(["TransformByQuantizing"])
print(imd.pipeline)
```

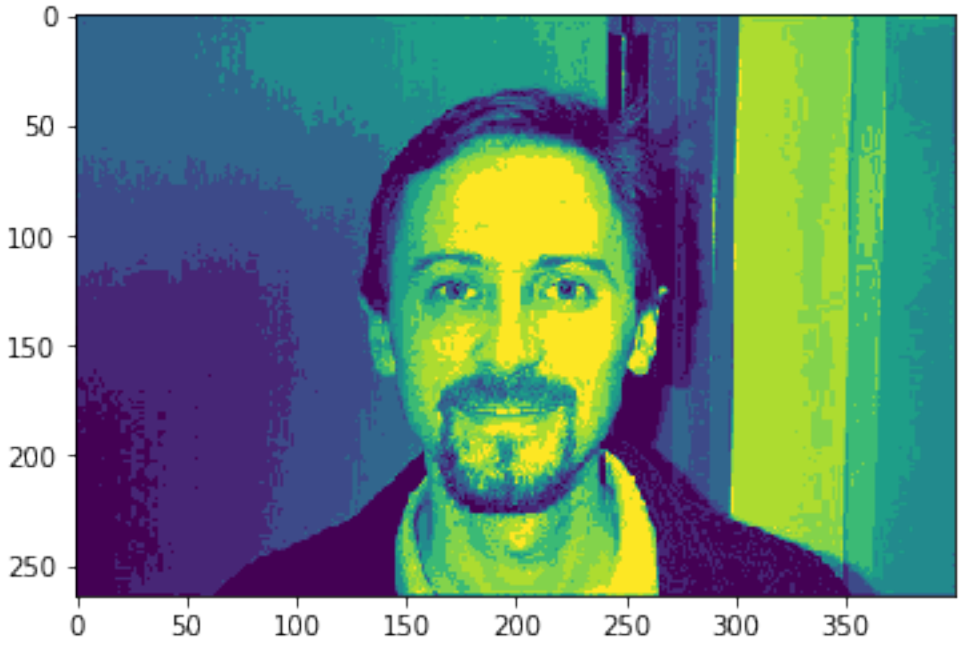
```
Kaishi pipeline:
0: TransformByQuantizing
     n_colors: 32
```

```
imd.pipeline.components[0].configure(n_colors=10)
imd.pipeline.components[0].applies_to("image_01.*jpg")
print(imd.pipeline)
imd.run_pipeline()
```

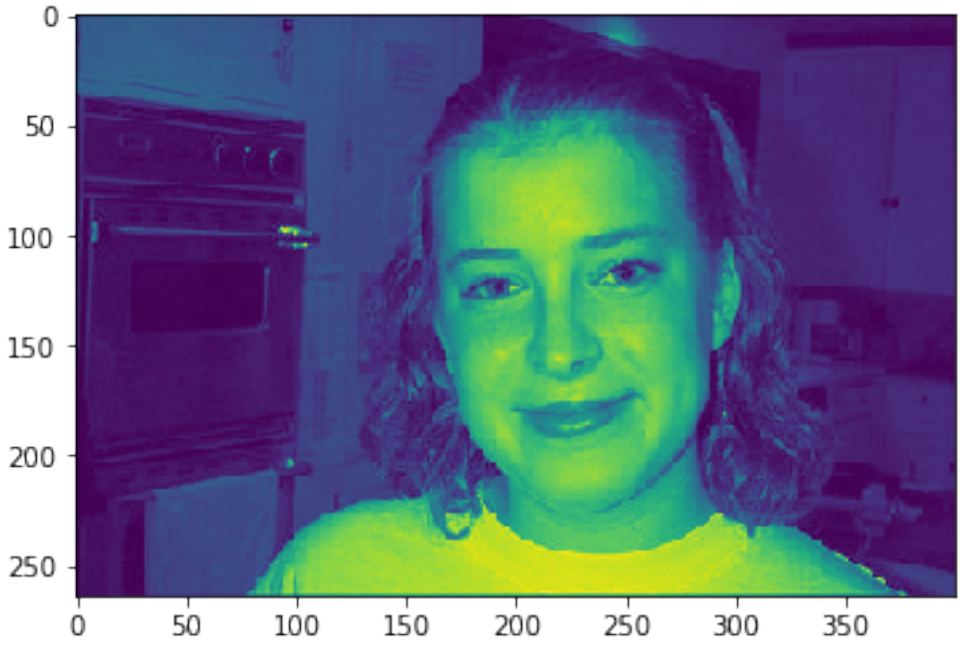
```
Kaishi pipeline:
0: TransformByQuantizing
     n_colors: 10
```

As expected, any image with the pattern image_01… is quantized (most notable in the background) whereas any image not matching this pattern remained intact.

```
plt.imshow(imd["image_0146.jpg"].image)
plt.show()
plt.imshow(imd["image_0361.jpg"].image)
plt.show()
```


0
50
100
150
200
250
0
50
100
150
200
250
300
350

png


0
50
100
150
200
250
0
50
100
150
200
250
300
350

png

3.1.5 Saving

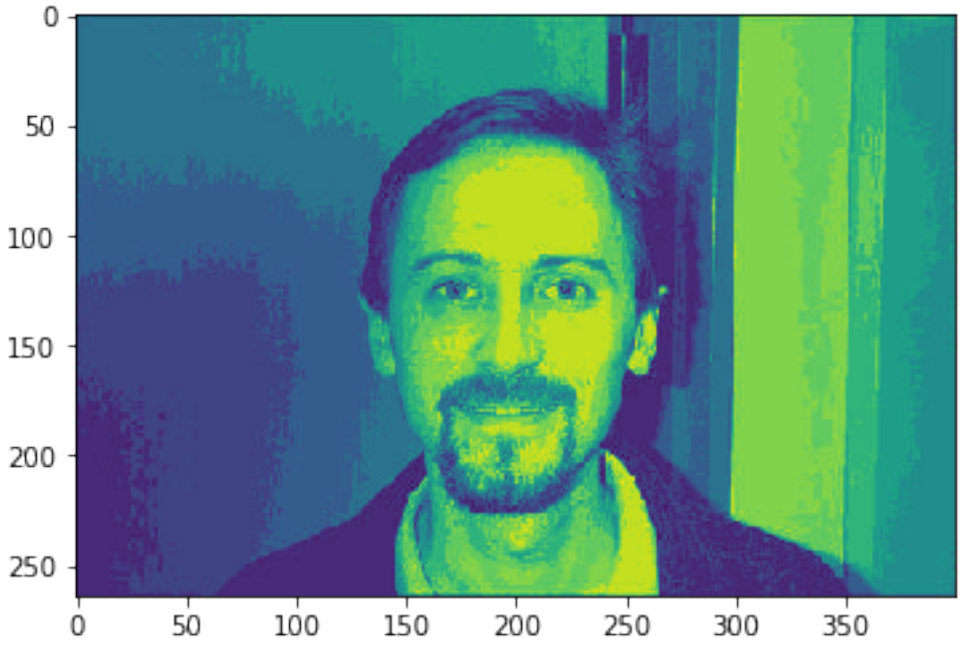
Finally, we can save the edited dataset

```
imd.save("Faces_edited")
```

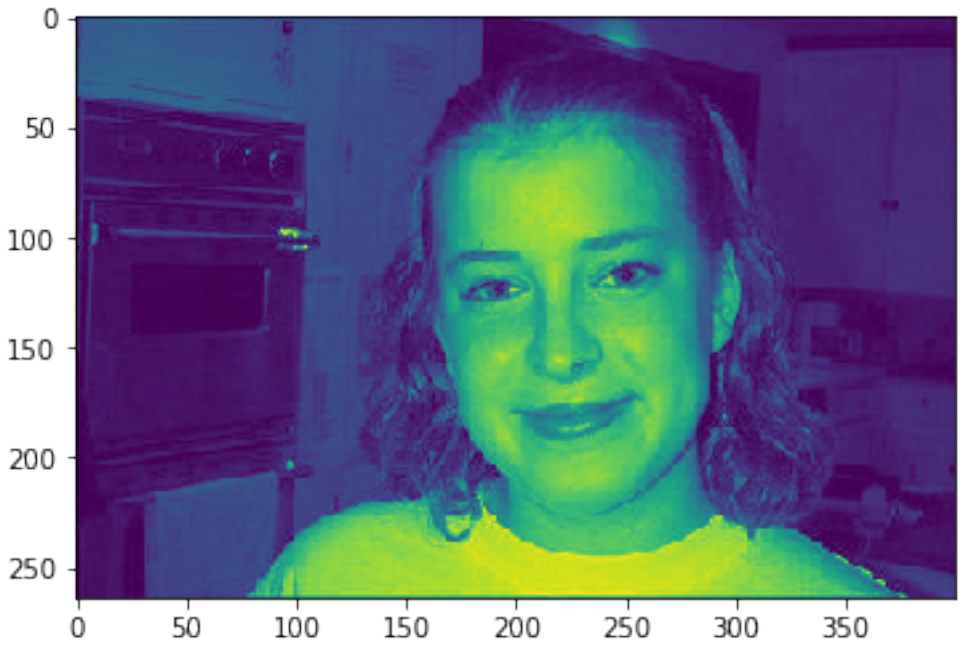
```
imd_edited = ImageDataset("Faces_edited")
imd_edited.load_all()
```

```
/Users/mwharton/.miniconda3/envs/kaishi/lib/python3.7/site-packages/kaishi/image/
↪dataset.py:20: UserWarning: No GPU detected, ConvNet prediction tasks will be very␣
↪slow
  warnings.warn("No GPU detected, ConvNet prediction tasks will be very slow")
```

```
plt.imshow(imd_edited["image_0146.jpg"].image)
plt.show()
plt.imshow(imd_edited["image_0361.jpg"].image)
plt.show()
```



png



png

3.2 Tabular Datasets

Tabular datasets in this context are directories of files (any variant of .csv or .json is accepted).

3.2.1 Initializing datasets

Let's start by creating our own toy dataset (with one duplicate, i.e. files 1 and 2)

```
import pandas as pd
import os
outdir = "toy_csv"
os.mkdir(outdir)

csv1 = """
"Index", "Living Space (sq ft)", "Beds", "Baths"
 1, 2222, 3, 3.5
 2, 1628, 3, 2
 3, 3824, 5, 4
 4, 1137, 3, 2
 5, 3560, 6, 4
 6, 2893, 4, 3
 7, 3631, 4, 3
 8, 2483, 4, 3
 9, 2400, 4, 4
10, 1997, 3, 3
"""
csv2 = """
"Index", "Living Space (sq ft)", "Beds", "Baths"
 1, 2222, 3, 3.5
 2, 1628, 3, 2
```

(continues on next page)

(continued from previous page)

```
 3, 3824, 5, 4
 4, 1137, 3, 2
 5, 3560, 6, 4
 6, 2893, 4, 3
 7, 3631, 4, 3
 8, 2483, 4, 3
 9, 2400, 4, 4
10, 1997, 3, 3
"""
csv3 = """
"Index", "Living Space (sq ft)", "Beds", "Baths"
11, 2222, 3, 3.5
12, 1628, 3, 2
13, 3824, 5, 4
14, 1137, 3, 2
15, 3560, 6, 4
16, 2893, 4, 3
17, 3631, 4, 3
18, 2483, 4, 3
19, 2400, 4, 4
20, 1997, 3, 3
"""
with open(outdir + "/1.csv", "w") as fd:
    fd.write(csv1)
with open(outdir + "/2.csv", "w") as fd:
    fd.write(csv2)
with open(outdir + "/3.csv", "w") as fd:
    fd.write(csv3)
```

```
from kaishi.tabular.dataset import TabularDataset
td = TabularDataset(outdir)
td.file_report()
```

```
Current file list:
+-------+-----------+-------------------------+--------+
| Index | File Name | Children                | Labels |
+-------+-----------+-------------------------+--------+
|   0   |   1.csv   |     {'duplicates': []} |   []   |
|   1   |   3.csv   |     {'duplicates': []} |   []   |
|   2   |   2.csv   |     {'duplicates': []} |   []   |
+-------+-----------+-------------------------+--------+
Filtered files:
+-----------+---------------+
| File Name | Filter Reason |
+-----------+---------------+
+-----------+---------------+
```

3.2.2 Interaction with datasets

There are several ways to interact with tabular datasets. Let's start by looking at a detailed report.

```
td.report()
```

```
Dataframe 0
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/1.csv
====================================
NO DATA OR NOT LOADED (try running 'dataset.load_all()')


Dataframe 1
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/3.csv
====================================
NO DATA OR NOT LOADED (try running 'dataset.load_all()')


Dataframe 2
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/2.csv
====================================
NO DATA OR NOT LOADED (try running 'dataset.load_all()')
```

Our data weren't loaded, let's fix that and try again

```
td.load_all()
td.report()
```

```
Dataframe 0
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/1.csv
====================================
4 columns: ['Index', ' "Living Space (sq ft)"', ' "Beds"', ' "Baths"']

---  Column 'Index'
count    10.00000
mean      5.50000
std       3.02765
min       1.00000
25%       3.25000
50%       5.50000
75%       7.75000
max      10.00000
Name: Index, dtype: float64

---  Column ' "Living Space (sq ft)"'
count      10.00000
mean     2577.50000
std       894.97725
min      1137.00000
25%      2053.25000
50%      2441.50000
75%      3393.25000
max      3824.00000
Name:  "Living Space (sq ft)", dtype: float64

---  Column ' "Beds"'
count    10.000000
```

(continues on next page)

(continued from previous page)

```
mean      3.900000
std       0.994429
min       3.000000
25%       3.000000
50%       4.000000
75%       4.000000
max       6.000000
Name:  "Beds", dtype: float64

---  Column ' "Baths"'
count    10.000000
mean      3.150000
std       0.747217
min       2.000000
25%       3.000000
50%       3.000000
75%       3.875000
max       4.000000
Name:  "Baths", dtype: float64

***** Fraction of missing data in each column *****
Index: 0.0
 "Living Space (sq ft)": 0.0
 "Beds": 0.0
 "Baths": 0.0


Dataframe 1
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/3.csv
====================================
4 columns: ['Index', ' "Living Space (sq ft)"', ' "Beds"', ' "Baths"']

---  Column 'Index'
count    10.00000
mean     15.50000
std       3.02765
min      11.00000
25%      13.25000
50%      15.50000
75%      17.75000
max      20.00000
Name: Index, dtype: float64

---  Column ' "Living Space (sq ft)"'
count      10.00000
mean     2577.50000
std       894.97725
min      1137.00000
25%      2053.25000
50%      2441.50000
75%      3393.25000
max      3824.00000
Name:  "Living Space (sq ft)", dtype: float64

---  Column ' "Beds"'
count    10.000000
mean      3.900000
```

(continues on next page)

(continued from previous page)

```
std       0.994429
min       3.000000
25%       3.000000
50%       4.000000
75%       4.000000
max       6.000000
Name:  "Beds", dtype: float64

---  Column ' "Baths"'
count    10.000000
mean      3.150000
std       0.747217
min       2.000000
25%       3.000000
50%       3.000000
75%       3.875000
max       4.000000
Name:  "Baths", dtype: float64

***** Fraction of missing data in each column *****
Index: 0.0
 "Living Space (sq ft)": 0.0
 "Beds": 0.0
 "Baths": 0.0


Dataframe 2
source: /Users/mwharton/Code/kaishi/notebooks/toy_csv/2.csv
====================================
4 columns: ['Index', ' "Living Space (sq ft)"', ' "Beds"', ' "Baths"']

---  Column 'Index'
count    10.00000
mean      5.50000
std       3.02765
min       1.00000
25%       3.25000
50%       5.50000
75%       7.75000
max      10.00000
Name: Index, dtype: float64

---  Column ' "Living Space (sq ft)"'
count      10.00000
mean     2577.50000
std       894.97725
min      1137.00000
25%      2053.25000
50%      2441.50000
75%      3393.25000
max      3824.00000
Name:  "Living Space (sq ft)", dtype: float64

---  Column ' "Beds"'
count    10.000000
mean      3.900000
std       0.994429
```

(continues on next page)

(continued from previous page)

```
min        3.000000
25%        3.000000
50%        4.000000
75%        4.000000
max        6.000000
Name:  "Beds", dtype: float64

---  Column ' "Baths"'
count    10.000000
mean      3.150000
std       0.747217
min       2.000000
25%       3.000000
50%       3.000000
75%       3.875000
max       4.000000
Name:  "Baths", dtype: float64

***** Fraction of missing data in each column *****
Index: 0.0
 "Living Space (sq ft)": 0.0
 "Beds": 0.0
 "Baths": 0.0
```

To look at a specific file object, you can access via either index or key

```
td.files[0].df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

```
td["1.csv"].df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

3.2.3 Tabular data processing pipelines

Let's see the pipeline options

```
td.get_pipeline_options()
```

```
['FilterByLabel',
 'FilterByRegex',
 'FilterDuplicateFiles',
 'FilterDuplicateRowsAfterConcatenation',
 'FilterDuplicateRowsEachDataframe',
 'FilterInvalidFileExtensions',
 'FilterSubsample',
 'LabelerValidationAndTest',
 'AggregatorConcatenateDataframes']
```

Now let's configure our own pipeline and run it

```
td.configure_pipeline(["FilterDuplicateFiles", "AggregatorConcatenateDataframes"])
print(td.pipeline)
td.run_pipeline()
```

```
Kaishi pipeline:
0: FilterDuplicateFiles
1: AggregatorConcatenateDataframes
```

As expected, the duplicate file was filtered

```
td.file_report()
```

```
Current file list:
+-------+-----------+------------------------------+--------+
| Index | File Name | Children                     | Labels |
+-------+-----------+------------------------------+--------+
|   0   |   1.csv   |     {'duplicates': [2.csv]} |   []   |
|   1   |   3.csv   |     {'duplicates': []}      |   []   |
+-------+-----------+------------------------------+--------+
Filtered files:
+-----------+---------------+
| File Name | Filter Reason |
+-----------+---------------+
|   2.csv   |   duplicates  |
+-----------+---------------+
```

But what about the concatenated dataframe? When Kaishi pipeline components create artifacts, they are added to the artifacts member of a dataset.

```
print(td.artifacts.keys())
```

```
dict_keys(['df_concatenated'])
```

```
td.artifacts["df_concatenated"]
```

```
.dataframe tbody tr th {
    vertical-align: top;
```

(continues on next page)

(continued from previous page)

```
}

.dataframe thead th {
    text-align: right;
}
```

This ultimately sets the framework for being able to manipulate your own tabular data sets and add custom functionality, without the hassle of dealing with the boring and monotonous ETL steps.

3.3 File datasets

If there's a particular data type you are working with, it's usually better to use the type-specific dataset object. However, there are still a few core operations that can be performed on generic files.

3.3.1 Initializing datasets

Let's start by creating a simple dataset of text files.

```
import os
outdir = "toy_files"
os.mkdir(outdir)

file1 = "file1_contents"
file2 = "file2_contents"
file2_duplicate = "file2_contents"
file3 = "file3_contents"
with open(outdir + "/1.file", "w") as fd:
    fd.write(file1)
with open(outdir + "/2.file", "w") as fd:
    fd.write(file2)
with open(outdir + "/2_dup.file", "w") as fd:
    fd.write(file2_duplicate)
with open(outdir + "/3.file", "w") as fd:
    fd.write(file3)
```

```
from kaishi.core.dataset import FileDataset
fd = FileDataset(outdir)
fd.file_report()
```

```
Current file list:
+-------+------------+-------------------------+--------+
| Index | File Name  | Children                | Labels |
+-------+------------+-------------------------+--------+
|   0   |   3.file   |     {'duplicates': []} |   []   |
|   1   |   2.file   |     {'duplicates': []} |   []   |
|   2   | 2_dup.file |     {'duplicates': []} |   []   |
|   3   |   1.file   |     {'duplicates': []} |   []   |
+-------+------------+-------------------------+--------+
Filtered files:
+-----------+---------------+
| File Name | Filter Reason |
+-----------+---------------+
+-----------+---------------+
```

3.3.2 File procesing pipelines

There are fewer components available for files compared to other types, as the other types inherit from the FileGroup class. However, there are still plenty of options available to perform some common operations.

```
fd.get_pipeline_options()
```

```
['FilterByLabel',
 'FilterByRegex',
 'FilterDuplicateFiles',
 'FilterSubsample',
 'LabelerValidationAndTest']
```

```
fd.configure_pipeline(["FilterDuplicateFiles", "FilterSubsample"])
print(fd.pipeline)
```

```
Kaishi pipeline:
0: FilterDuplicateFiles
1: FilterSubsample
     N: None
     seed: None
```

```
fd.pipeline.components[1].configure(N=2, seed=42)
print(fd.pipeline)
```

```
Kaishi pipeline:
0: FilterDuplicateFiles
1: FilterSubsample
     N: 2
     seed: 42
```

```
fd.run_pipeline()
fd.file_report()
```

```
Current file list:
+-------+-----------+-----------------------------------+--------+
| Index | File Name | Children                          | Labels |
+-------+-----------+-----------------------------------+--------+
|   0   |   3.file  |     {'duplicates': []}            |   []   |
|   1   |   2.file  |     {'duplicates': [2_dup.file]} |   []   |
+-------+-----------+-----------------------------------+--------+
Filtered files:
+------------+---------------+
| File Name  | Filter Reason |
+------------+---------------+
| 2_dup.file |   duplicates  |
|   1.file   |   subsample   |
+------------+---------------+
```

PYTHON MODULE INDEX

k

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

P

R

S

T

U

V