MLTSP Documentation

Release 0.2

The MLTSP Team

March 01, 2016

Contents

CHAPTER 1

Installation

1.1 Installation (library)

The latest version of `mltsp` can be installed via `pip`:

```
pip install mltsp
```

The MLTSP library has the following dependencies:

- numpy
- scipy
- pandas
- scikit-learn
- cython
- dask
- xarray

The easiest way to install the necessary dependencies is using `conda`:

```
conda install numpy scipy pandas scikit-learn cython dask xarray
```

The `mltsp` library is compatible with both Python 2 and 3.

1.2 Installation (web app)

- Install the library dependencies above:

```
conda install numpy scipy pandas scikit-learn cython dask xarray
```

- Install RabbitMQ
  - Ensure server is running with `rabbitmq-server -detached`
- Install RethinkDB
- Install Docker (optional)
  - Only required to support the execution of user-specified (custom) feature extraction functions
  - Pull down the required images: `tools/docker_pull.sh`

  - Alternatively (but this takes much longer), build the images on your own machine: `tools/build_docker_images.sh`
- Install the `mltsp` package (from source)
  - Clone the (git repo)[https://github.com/mltsp/mltsp]: `git clone https://github.com/mltsp/mltsp.git`
  - Within the source directory, install via `pip install -e .`
- Setup sample data and configuration files: `mltsp --install`
  - Optional: locate `~/.config/mltsp/mltsp.yaml` and customize authentication tokens

1.3 Starting the web app

- Create the MLTSP database: `mltsp --db-init`
- Launch the web platform: `cd web_client && make`
- User authentication is required by default; disable by modifying 'disable_auth' in the configuration file.
- Navigate to `http://localhost:5000`.
- The port can be configured in `web_client/nginx.conf`.

1.4 Testing

1.4.1 Back-end

- `pip install nose nose-exclude mock`
- `make test_backend`

1.4.2 Front-end

- Install PhantomJS
- `make test_frontend`

CHAPTER 2

API Reference

2.1 mltsp

Machine Learning Time-Series Platform (MLTSP)

See http://mltsp.io for more information.

mltsp.install()	Install MLTSP config file in ~/.config/mltsp/mltsp.yaml.

2.1.1 install

mltsp.**install**()
Install MLTSP config file in ~/.config/mltsp/mltsp.yaml.

2.2 Module: datasets

mltsp.datasets.fetch_andrzejak([data_dir])	Download (if not already downloaded) and load an example EEG da
mltsp.datasets.fetch_asas_training([data_dir])	Download (if not already downloaded) and load example light curve

2.2.1 fetch_andrzejak

mltsp.datasets.**fetch_andrzejak**(*data_dir=None*)
Download (if not already downloaded) and load an example EEG dataset.

Parameters **data_dir** : str, optional

Path where downloaded data should be stored. Defaults to a subdirectory *datasets/andrzejak* within *dsutil.DATA_PATH*.

Returns dict

Dictionary with attributes:

- times: list of (4096,) arrays of time values
- measurements: list of (4096,) arrays of measurement values
- classes: array of class labels for each time series

- archive: path to data archive
- header: path to header file

References

Andrzejak, Ralph G., et al. "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state." *Phys. Rev. E* 64.6 (2001): 061907.

2.2.2 fetch_asas_training

`mltsp.datasets.`**`fetch_asas_training`**(*data_dir=None*)

Download (if not already downloaded) and load example light curve data.

Parameters **data_dir: str, optional**

Path where downloaded data should be stored. Defaults to a subdirectory *datasets/asas_training* within *dsutil.DATA_PATH*.

Returns dict

Dictionary attributes:

- times: list of arrays of time values
- measurements: list of arrays of measurement values
- errors: list of arrays of error values
- classes: Series of classes for each time series indexed by file
- metadata: DataFrame of metafeature values indexed by file
- archive: path to data archive
- header: path to header file

References

Andrzejak, Ralph G., et al. "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state." Physical Review E 64.6 (2001): 061907.

2.3 Module: `featurize`

mltsp.featurize.celery_available()	Test Celery task; this is much faster than running *celery status*.
mltsp.featurize.featurize_data_files(ts_paths)	Generate features for labeled time series data.
mltsp.featurize.featurize_time_series(times, ...)	Versatile feature generation function for one or more time serie
mltsp.featurize.load_and_store_feature_data(...)	Read features from CSV file and save as an xarray.Dataset.
mltsp.featurize.TimeSeries([t, m, e, ...])	Class representing a single time series of measurements and me

2.3.1 celery_available

mltsp.featurize.**celery_available**()
Test Celery task; this is much faster than running *celery status*.

2.3.2 featurize_data_files

mltsp.featurize.**featurize_data_files**(*ts_paths*, *features_to_use=[]*, *output_path=None*, *first_N=None*, *custom_script_path=None*, *use_docker=True*)
Generate features for labeled time series data.

Each file should consist of one comma-separated line of per data point, where each line contains either pairs (time, value) or triples (time, value, error).

If *output_path* is provided, features are saved as an xarray.Dataset in netCDF format.

Parameters **ts_paths** : str

Path to netCDF files containing serialized TimeSeries objects.

features_to_use : list of str, optional

List of feature names to be generated. Defaults to an empty list, which will result in only meta_features features being stored.

output_path : str, optional

Path to which the output xarray.Dataset of feature data will be saved (if applicable).

first_N : int, optional

Integer indicating the maximum number of time series to featurize. Can be used to reduce the number of files for testing purposes. If *first_N* is None then all time series will be featurized.

custom_script_path : str, optional

Path to Python script containing function definitions for the generation of any custom features. Defaults to None.

use_docker : bool, optional

Bool specifying whether to generate custom features inside a Docker container. Defaults to True.

Returns xarray.Dataset

Featureset with *data_vars* containing feature values, and *coords* containing filenames and targets (if applicable).

2.3.3 featurize_time_series

mltsp.featurize.**featurize_time_series**(*times*, *values*, *errors=None*, *features_to_use=[]*, *targets=None*, *meta_features={}*, *labels=None*, *custom_script_path=None*, *custom_functions=None*, *use_docker=True*, *use_celery=False*)
Versatile feature generation function for one or more time series.

For a single time series, inputs may have the form:

- times: (n,) array or (p, n) array (for p channels of measurement)

- values: (n,) array or (p, n) array (for p channels of measurement)
- errors: (n,) array or (p, n) array (for p channels of measurement)

For multiple time series, inputs may have the form:

- times: list of (n,) arrays, list of (p, n) arrays (for p channels of measurement), or list of lists of (n,) arrays (for multichannel data with different time values per channel)
- values: list of (n,) arrays, list of (p, n) arrays (for p channels of measurement), or list of lists of (n,) arrays (for multichannel data with different time values per channel)
- errors: list of (n,) arrays, list of (p, n) arrays (for p channels of measurement), or list of lists of (n,) arrays (for multichannel data with different time values per channel)

In the case of multichannel measurements, each channel will be featurized separately, and the data variables of the output *xarray.Dataset* will be indexed by a *channel* coordinate.

Parameters **times** : array, list of array, or list of lists of array

Array containing time values for a single time series, or a list of arrays each containing time values for a single time series, or a list of lists of arrays for multichannel data with different time values per channel

values : array or list of array

Array containing measurement values for a single time series, or a list of arrays each containing (possibly multivariate) measurement values for a single time series, or a list of lists of arrays for multichannel data with different time values per channel

errors : array or list/tuple of array, optional

Array containing measurement error values for a single time series, or a list of arrays each containing (possibly multivariate) measurement values for a single time series, or a list of lists of arrays for multichannel data with different time values per channel

features_to_use : list of str, optional

List of feature names to be generated. Defaults to an empty list, which will result in only meta_features features being stored.

targets : str/float or array-like, optional

Target or sequence of targets, one per time series (if applicable); will be stored in the *target* coordinate of the resulting *xarray.Dataset*.

meta_features : dict/Pandas.Series or list of dicts/Pandas.DataFrame

dict/Series (for a single time series) or DataFrame (for multiple time series) of metafeature information; features are added to the output featureset, and their values are consumable by custom feature scripts.

labels : str or list of str, optional

Label or list of labels for each time series, if applicable; will be stored in the *name* coordinate of the resulting *xarray.Dataset*.

custom_script_path : str, optional

Path to Python script containing function definitions for the generation of any custom features. Defaults to None.

custom_functions : dict, optional

Dictionary of custom feature functions to be evaluated for the given time series, or a dictionary representing a dask graph of function evaluations. Dictionaries of functions should have keys *feature_name* and values functions that take arguments (t, m, e); in the case of a dask graph, these arrays should be referenced as 't', 'm', 'e', respectively, and any values with keys present in *features_to_use* will be computed.

use_docker : bool, optional

Bool specifying whether to generate custom features inside a Docker container. Defaults to True.

use_celery : bool, optional

Boolean to control whether to distribute tasks to Celery workers (if Celery is available). Defaults to True.

Returns xarray.Dataset

Featureset with *data_vars* containing feature values and *coords* containing labels (*name*) and targets (*target*), if applicable.

2.3.4 load_and_store_feature_data

`mltsp.featurize.`**`load_and_store_feature_data`**(*features_path*, *output_path*, *first_N=None*)

Read features from CSV file and save as an xarray.Dataset.

2.3.5 `TimeSeries`

class `mltsp.featurize.`**`TimeSeries`**(*t=None*, *m=None*, *e=None*, *target=None*, *meta_features={}*, *name=None*, *path=None*, *channel_names=None*)

Bases: `object`

Class representing a single time series of measurements and metadata.

A *TimeSeries* object encapsulates a single set of time-domain measurements, along with any metadata describing the observation. Typically the observations will consist of times, measurements, and (optionally) measurement errors. The measurements can be scalar- or vector-valued (i.e., "multichannel"); for multichannel measurements, the times and errors can also be vector-valued, or they can be shared across all channels of measurement.

Attributes

time	((n,) or (p, n) array or list of (n,) arrays) Array(s) of times corresponding to measurement values. If *measurement* is multidimensional, this can be one-dimensional (same times for each channel) or multidimensional (different times for each channel).
measurement	((n,) or (p, n) array or list of (n,) arrays) Array(s) of measurement values; can be multidimensional for multichannel data. In the case of multichannel data with different numbers of measurements for each channel, *measurement* will be a list of arrays instead of a single two-dimensional array.
error	((n,) or (p, n) array or list of (n,) arrays) Array(s) of measurement errors for each value. If *measurement* is multidimensional, this can be one-dimensional (same times for each channel) or multidimensional (different times for each channel).
target	(str, float, or None) Class label or target value for the given time series (if applicable).
meta_features	(dict) Dictionary of feature names/values specified independently of the featurization process in *featurize*.
name	(str or None) Identifying name/label for the given time series (if applicable). Typically the name of the raw data file from which the time series was created.
path	(str or None) Path to the file where the time series is stored on disk (if applicable).
channel_names	(list of str) List of names of channels of measurement; by default these are simply *channel_{i}*, but can be arbitrary depending on the nature of the different measurement channels.

Methods

`channels`()	Iterates over measurement channels (whether one or multiple).
`to_netcdf`([path])	Store TimeSeries object as a single netCDF.

__init__(*t=None*, *m=None*, *e=None*, *target=None*, *meta_features={}*, *name=None*, *path=None*, *channel_names=None*)
Create a *TimeSeries* object from measurement values/metadata.

See *TimeSeries* documentation for parameter values.

channels()
Iterates over measurement channels (whether one or multiple).

to_netcdf(*path=None*)
Store TimeSeries object as a single netCDF.

Each channel of measurements is stored in a separate HDF5 group; metadata describing the whole time series is stored in the group corresponding to the first channel.

If *path* is omitted then the *path* attribute from the TimeSeries object is used.

2.4 Module: obs_feature_tools

`mltsp.obs_feature_tools.cad_prob`(cads, time)	Given the observed distribution of time lags *cads*, comp
`mltsp.obs_feature_tools.delta_t_hist`(t[, nbins])	Build histogram of all possible delta_t's without storing
`mltsp.obs_feature_tools.double_to_single_step`(cads)	Ratios t[i+2] - t[i] / (t[i+2] - t[i+1]).
`mltsp.obs_feature_tools.find_sorted_peaks`(x)	Find peaks, i.e.
`mltsp.obs_feature_tools.generate_obs_features`(t, m, e)	Generate features dict from given time-series data.

Table 2.5 – continued from previous pa

mltsp.obs_feature_tools.normalize_hist(hist, ...)	Normalize histogram such that integral from t_min to t_
mltsp.obs_feature_tools.peak_bin(peaks, i)	Return the (bin) index of the ith largest peak.
mltsp.obs_feature_tools.peak_ratio(peaks, i, j)	Compute the ratio of the values of the ith and jth largest

2.4.1 cad_prob

mltsp.obs_feature_tools.**cad_prob**(*cads*, *time*)

Given the observed distribution of time lags *cads*, compute the probability that the next observation occurs within *time* minutes of an arbitrary epoch.

2.4.2 delta_t_hist

mltsp.obs_feature_tools.**delta_t_hist**(*t*, *nbins=50*)

Build histogram of all possible delta_t's without storing every value

2.4.3 double_to_single_step

mltsp.obs_feature_tools.**double_to_single_step**(*cads*)

Ratios t[i+2] - t[i] / (t[i+2] - t[i+1]).

2.4.4 find_sorted_peaks

mltsp.obs_feature_tools.**find_sorted_peaks**(*x*)

Find peaks, i.e. local maxima, of an array. Interior points are peaks if they are greater than both their neighbors, and edge points are peaks if they are greater than their only neighbor. In the case of ties, we (arbitrarily) choose the first index in the sequence of equal values as the peak.

Returns a list of tuples (i, x[i]) of peak indices i and values x[i], sorted in decreasing order by peak value.

2.4.5 generate_obs_features

mltsp.obs_feature_tools.**generate_obs_features**(*t, m, e, features_to_compute=['n_epochs', 'avg_err', 'med_err', 'std_err', 'total_time', 'avgt', 'cads_std', 'avg_mag', 'cads_avg', 'cads_med', 'cad_probs_1', 'cad_probs_10', 'cad_probs_20', 'cad_probs_30', 'cad_probs_40', 'cad_probs_50', 'cad_probs_100', 'cad_probs_500', 'cad_probs_1000', 'cad_probs_5000', 'cad_probs_10000', 'cad_probs_50000', 'cad_probs_100000', 'cad_probs_500000', 'cad_probs_1000000', 'cad_probs_5000000', 'cad_probs_10000000', 'med_double_to_single_step', 'avg_double_to_single_step', 'std_double_to_single_step', 'all_times_hist_peak_val', 'all_times_hist_peak_bin', 'all_times_nhist_numpeaks', 'all_times_nhist_peak_val', 'all_times_nhist_peak_1_to_2', 'all_times_nhist_peak_1_to_3', 'all_times_nhist_peak_2_to_3', 'all_times_nhist_peak_1_to_4', 'all_times_nhist_peak_2_to_4', 'all_times_nhist_peak_3_to_4', 'all_times_nhist_peak1_bin', 'all_times_nhist_peak2_bin', 'all_times_nhist_peak3_bin', 'all_times_nhist_peak4_bin']*)

Generate features dict from given time-series data.

Parameters **t** : array_like

Array containing time values.

m : array_like

Array containing data values.

e : array_like

Array containing measurement error values.

features_to_compute : list

Optional list containing names of desired features.

Returns dict

Dictionary containing generated time series features.

2.4.6 normalize_hist

mltsp.obs_feature_tools.**normalize_hist**(*hist*, *total_time*)
Normalize histogram such that integral from t_min to t_max equals 1. cf. np.histogram(..., density=True).

2.4.7 peak_bin

mltsp.obs_feature_tools.**peak_bin**(*peaks*, *i*)
Return the (bin) index of the ith largest peak.

2.4.8 peak_ratio

mltsp.obs_feature_tools.**peak_ratio**(*peaks*, *i*, *j*)
Compute the ratio of the values of the ith and jth largest peaks.

2.5 Module: science_feature_tools

mltsp.science_feature_tools.generate_science_features(t, m, e)	Generate science features for provided time

2.5.1 generate_science_features

mltsp.science_feature_tools.**generate_science_features**(*t, m, e, features_to_compute=['amplitude', 'percent_beyond_1_std', 'flux_percentile_ratio_mid20', 'flux_percentile_ratio_mid35', 'flux_percentile_ratio_mid50', 'flux_percentile_ratio_mid65', 'flux_percentile_ratio_mid80', 'fold2P_slope_10percentile', 'fold2P_slope_90percentile', 'freq1_amplitude1', 'freq1_amplitude2', 'freq1_amplitude3', 'freq1_amplitude4', 'freq1_freq', 'freq1_rel_phase2', 'freq1_rel_phase3', 'freq1_rel_phase4', 'freq1_lambda', 'freq2_amplitude1', 'freq2_amplitude2', 'freq2_amplitude3', 'freq2_amplitude4', 'freq2_freq', 'freq2_rel_phase2', 'freq2_rel_phase3', 'freq2_rel_phase4', 'freq3_amplitude1', 'freq3_amplitude2', 'freq3_amplitude3', 'freq3_amplitude4', 'freq3_freq', 'freq3_rel_phase2', 'freq3_rel_phase3', 'freq3_rel_phase4', 'freq_amplitude_ratio_21', 'freq_amplitude_ratio_31', 'freq_frequency_ratio_21', 'freq_frequency_ratio_31', 'freq_model_max_delta_mags', 'freq_model_min_delta_mags', 'freq_model_phi1_phi2', 'freq_n_alias', 'freq1_signif', 'freq_signif_ratio_21', 'freq_signif_ratio_31', 'freq_varrat', 'freq_y_offset', 'linear_trend', 'maximum', 'max_slope', 'median', 'median_absolute_deviation', 'percent_close_to_median', 'medperc90_2p_p', 'minimum', 'p2p_scatter_2praw', 'p2p_scatter_over_mad', 'p2p_scatter_pfold_over_mad', 'p2p_ssqr_diff_over_var', 'percent_amplitude', 'percent_difference_flux_percentile', 'period_fast',*

Parameters **t** : array_like

Array containing time values.

m : array_like

Array containing data values.

e : array_like

Array containing measurement error values.

features_to_compute : list

Optional list containing names of desired features.

Returns dict

Dictionary containing newly-generated features. Keys are feature names, values are feature values (floats).

2.6 Module: science_features

mltsp.science_features.amplitude(x)	Half the difference between the maximum
mltsp.science_features.flux_percentile_ratio(x, ...)	A ratio of ((50+x) flux percentile - (50-x) f
mltsp.science_features.get_fold2P_slope_percentile(...)	Get alphath percentile of slopes of period-f
mltsp.science_features.get_lomb_amplitude(...)	Get the amplitude of the jth harmonic of th
mltsp.science_features.get_lomb_amplitude_ratio(...)	Get the ratio of the amplitudes of the first h
mltsp.science_features.get_lomb_frequency(...)	Get the ith frequency from a fitted Lomb-S
mltsp.science_features.get_lomb_frequency_ratio(...)	Get the ratio of the ith and first frequencies
mltsp.science_features.get_lomb_lambda(...)	Get the regularization parameter of a fitted
mltsp.science_features.get_lomb_rel_phase(...)	Get the relative phase of the jth harmonic c
mltsp.science_features.get_lomb_signif(...)	Get the significance (in sigmas) of the first
mltsp.science_features.get_lomb_signif_ratio(...)	Get the ratio of the significances (in sigmas
mltsp.science_features.get_lomb_trend(lomb_model)	Get the linear trend of a fitted Lomb-Scarg
mltsp.science_features.get_lomb_varrat(...)	Get the fraction of the variance explained b
mltsp.science_features.get_lomb_y_offset(...)	Get the y-intercept of a fitted Lomb-Scargl
mltsp.science_features.get_max_delta_mags(model)	Largest value minus second largest value o
mltsp.science_features.get_medperc90_2p_p(model)	Get ratio of 90th percentiles of residuals fo
mltsp.science_features.get_min_delta_mags(model)	Second smallest value minus smallest valu
mltsp.science_features.get_model_phi1_phi2(model)	Ratio of distances between the second min
mltsp.science_features.get_p2p_scatter_2praw(model)	Get ratio of variability (sum of squared dif
mltsp.science_features.get_p2p_scatter_over_mad(model)	Get ratio of variability of folded and unfol
mltsp.science_features.get_p2p_scatter_pfold_over_mad(model)	Get ratio of median of period-folded data c
mltsp.science_features.get_p2p_ssqr_diff_over_var(model)	Get sum of squared differences of consecu
mltsp.science_features.get_qso_log_chi2_qsonu(...)	
mltsp.science_features.get_qso_log_chi2nuNULL_chi2nu(...)	
mltsp.science_features.lomb_scargle_fast_period(t, m, e)	Fits a simple sinuosidal model
mltsp.science_features.lomb_scargle_model(...)	Simultaneous fit of a sum of sinusoids by v
mltsp.science_features.max_slope(t, x)	Compute the largest rate of change in the c
mltsp.science_features.maximum(x)	Maximum observed value.
mltsp.science_features.median(x)	Median of observed values.
mltsp.science_features.median_absolute_deviation(x)	Median absolute deviation (from the media
mltsp.science_features.minimum(x)	Minimum observed value.

Table 2.7 – continued from p

mltsp.science_features.num_alias(lomb_model)	Here we check for "1-day" aliases in ASA
mltsp.science_features.p2p_model(x, y, frequency)	Compute features that compare the residua
mltsp.science_features.percent_amplitude(x)	Returns the largest distance from the medi
mltsp.science_features.percent_beyond_1_std(x, e)	Percentage of values more than 1 std.
mltsp.science_features.percent_close_to_median(x)	Percentage of values within window_frac*
mltsp.science_features.percent_difference_flux_percentile(x)	Difference between the 95th and 5th perce
mltsp.science_features.period_folding(x, y, ...)	This section is used to calculate Dubath (1
mltsp.science_features.periodic_model(lomb_model)	Compute features related to the extreme p
mltsp.science_features.qso_fit(time, data, error)	Best-fit qso model determined for Sesar St
mltsp.science_features.scatter_res_raw(t, m, ...)	From arXiv 1101_2406v1 Dubath 201101
mltsp.science_features.skew(x)	Skewness of a dataset.
mltsp.science_features.std(x)	Standard deviation of observed values.
mltsp.science_features.stetson_j(x[, y, dx, dy])	Robust covariance statistic between pairs o
mltsp.science_features.stetson_k(x[, dx])	A robust kurtosis statistic.
mltsp.science_features.weighted_average(x, e)	Arithmetic mean of observed values, weigl

2.6.1 amplitude

mltsp.science_features.**amplitude**(*x*)

Half the difference between the maximum and minimum magnitude.

2.6.2 flux_percentile_ratio

mltsp.science_features.**flux_percentile_ratio**(*x*, *percentile_range*, *base=10.0*, *exponent=-0.4*)

A ratio of ((50+x) flux percentile - (50-x) flux percentile) / (95 flux percentile - 5 flux percentile), where x = percentile_range/2.

Assumes data is log-scaled; by default we assume inputs are scaled as x=10^(-0.4*y), corresponding to units of magnitudes. Computations are performed on the corresponding linear-scale values.

2.6.3 get_fold2P_slope_percentile

mltsp.science_features.**get_fold2P_slope_percentile**(*model*, *alpha*)

Get alphath percentile of slopes of period-folded model.

2.6.4 get_lomb_amplitude

mltsp.science_features.**get_lomb_amplitude**(*lomb_model*, *i*, *j*)

Get the amplitude of the jth harmonic of the ith frequency from a fitted Lomb-Scargle model.

2.6.5 get_lomb_amplitude_ratio

mltsp.science_features.**get_lomb_amplitude_ratio**(*lomb_model*, *i*)

Get the ratio of the amplitudes of the first harmonic for the ith and first frequencies from a fitted Lomb-Scargle model.

2.6.6 get_lomb_frequency

mltsp.science_features.**get_lomb_frequency**(*lomb_model*, *i*)
Get the ith frequency from a fitted Lomb-Scargle model.

2.6.7 get_lomb_frequency_ratio

mltsp.science_features.**get_lomb_frequency_ratio**(*lomb_model*, *i*)
Get the ratio of the ith and first frequencies from a fitted Lomb-Scargle model.

2.6.8 get_lomb_lambda

mltsp.science_features.**get_lomb_lambda**(*lomb_model*)
Get the regularization parameter of a fitted Lomb-Scargle model.

2.6.9 get_lomb_rel_phase

mltsp.science_features.**get_lomb_rel_phase**(*lomb_model*, *i*, *j*)
Get the relative phase of the jth harmonic of the ith frequency from a fitted Lomb-Scargle model.

2.6.10 get_lomb_signif

mltsp.science_features.**get_lomb_signif**(*lomb_model*)
Get the significance (in sigmas) of the first frequency from a fitted Lomb-Scargle model.

2.6.11 get_lomb_signif_ratio

mltsp.science_features.**get_lomb_signif_ratio**(*lomb_model*, *i*)
Get the ratio of the significances (in sigmas) of the ith and first frequencies from a fitted Lomb-Scargle model.

2.6.12 get_lomb_trend

mltsp.science_features.**get_lomb_trend**(*lomb_model*)
Get the linear trend of a fitted Lomb-Scargle model.

2.6.13 get_lomb_varrat

mltsp.science_features.**get_lomb_varrat**(*lomb_model*)
Get the fraction of the variance explained by the first frequency of a fitted Lomb-Scargle model.

2.6.14 get_lomb_y_offset

mltsp.science_features.**get_lomb_y_offset**(*lomb_model*)
Get the y-intercept of a fitted Lomb-Scargle model.

2.6.15 get_max_delta_mags

mltsp.science_features.**get_max_delta_mags**(*model*)
Largest value minus second largest value of fitted Lomb Scargle model.

2.6.16 get_medperc90_2p_p

mltsp.science_features.**get_medperc90_2p_p**(*model*)
Get ratio of 90th percentiles of residuals for data folded by twice the estimated period and the estimated period, respectively.

2.6.17 get_min_delta_mags

mltsp.science_features.**get_min_delta_mags**(*model*)
Second smallest value minus smallest value of fitted Lomb Scargle model.

2.6.18 get_model_phi1_phi2

mltsp.science_features.**get_model_phi1_phi2**(*model*)
Ratio of distances between the second minimum and first maximum, and the second minimum and second maximum, of the fitted Lomb-Scargle model.

2.6.19 get_p2p_scatter_2praw

mltsp.science_features.**get_p2p_scatter_2praw**(*model*)
Get ratio of variability (sum of squared differences of consecutive values) of folded and unfolded models.

2.6.20 get_p2p_scatter_over_mad

mltsp.science_features.**get_p2p_scatter_over_mad**(*model*)
Get ratio of variability of folded and unfolded models.

2.6.21 get_p2p_scatter_pfold_over_mad

mltsp.science_features.**get_p2p_scatter_pfold_over_mad**(*model*)
Get ratio of median of period-folded data over median absolute deviation of observed values.

2.6.22 get_p2p_ssqr_diff_over_var

mltsp.science_features.**get_p2p_ssqr_diff_over_var**(*model*)
Get sum of squared differences of consecutive values as a fraction of the variance of the data.

2.6.23 get_qso_log_chi2_qsonu

mltsp.science_features.**get_qso_log_chi2_qsonu**(*qso_model*)

2.6.24 get_qso_log_chi2nuNULL_chi2nu

mltsp.science_features.**get_qso_log_chi2nuNULL_chi2nu**(*qso_model*)

2.6.25 lomb_scargle_fast_period

mltsp.science_features.**lomb_scargle_fast_period**(*t*, *m*, *e*)

Fits a simple sinuosidal model

y(t) = A sin(2*pi*w*t + phi) + c

and returns the estimated period 1/w. Much faster than fitting the full multi-frequency model used by *science_features.lomb_scargle*.

2.6.26 lomb_scargle_model

mltsp.science_features.**lomb_scargle_model**(*time*, *signal*, *error*, *sys_err=0.05*, *nharm=8*, *nfreq=3*, *tone_control=5.0*)

Simultaneous fit of a sum of sinusoids by weighted least squares: y(t) = Sum_k Ck*t^k + Sum_i Sum_j A_ij sin(2*pi*j*fi*(t-t0)+phi_j), i=[1,nfreq], j=[1,nharm]

Parameters **time** : array_like

Array containing time values.

signal : array_like

Array containing data values.

error : array_like

Array containing measurement error values.

nharm : int

Number of harmonics to fit for each frequency.

nfreq : int

Number of frequencies to fit.

Returns dict

Dictionary containing fitted parameter values. Parameters specific to a specific fitted frequency are stored in a list of dicts at model_dict['freq_fits'], each of which contains the output of fit_lomb_scargle(...)

2.6.27 max_slope

mltsp.science_features.**max_slope**(*t*, *x*)

Compute the largest rate of change in the observed data.

2.6.28 maximum

mltsp.science_features.**maximum**(*x*)

Maximum observed value.

2.6.29 median

mltsp.science_features.**median**(*x*)
Median of observed values.

2.6.30 median_absolute_deviation

mltsp.science_features.**median_absolute_deviation**(*x*)
Median absolute deviation (from the median) of the observed values.

2.6.31 minimum

mltsp.science_features.**minimum**(*x*)
Minimum observed value.

2.6.32 num_alias

mltsp.science_features.**num_alias**(*lomb_model*)
Here we check for "1-day" aliases in ASAS / Deboss sources.

2.6.33 p2p_model

mltsp.science_features.**p2p_model**(*x*, *y*, *frequency*)
Compute features that compare the residuals of data folded by estimated period from Lomb-Scargle model with residuals folded by twice the estimated period.

2.6.34 percent_amplitude

mltsp.science_features.**percent_amplitude**(*x*, *base=10.0*, *exponent=-0.4*)
Returns the largest distance from the median value, measured as a percentage of the median.

Assumes data is log-scaled; by default we assume inputs are scaled as x=10^(-0.4*y), corresponding to units of magnitudes. Computations are performed on the corresponding linear-scale values.

2.6.35 percent_beyond_1_std

mltsp.science_features.**percent_beyond_1_std**(*x*, *e*)
Percentage of values more than 1 std. dev. from the weighted average.

2.6.36 percent_close_to_median

mltsp.science_features.**percent_close_to_median**(*x*, *window_frac=0.1*)
Percentage of values within window_frac*(max(x)-min(x)) of median.

2.6.37 percent_difference_flux_percentile

mltsp.science_features.**percent_difference_flux_percentile**(*x*, *base=10.0*, *exponent=-0.4*)

Difference between the 95th and 5th percentiles of the data, expressed as a percentage of the median value. See Eyer (2005) arXiv:astro-ph/0511458v1, Evans & Belokurov (2005) (there the 98th and 2nd percentiles are used).

Assumes data is log-scaled; by default we assume inputs are scaled as x=10^(-0.4*y), corresponding to units of magnitudes. Computations are performed on the corresponding linear-scale values.

2.6.38 period_folding

mltsp.science_features.**period_folding**(*x*, *y*, *dy*, *lomb_model*, *sys_err=0.05*)

This section is used to calculate Dubath (10. Percentile90:2P/P), which requires regenerating a model using 2P where P is the original found period

NOTE: this essentially runs everything a second time, so makes feature generation take roughly twice as long.

2.6.39 periodic_model

mltsp.science_features.**periodic_model**(*lomb_model*)

Compute features related to the extreme points of the fitted Lomb Scargle model.

2.6.40 qso_fit

mltsp.science_features.**qso_fit**(*time*, *data*, *error*, *filter='g'*, *mag0=19.0*, *sys_err=0.0*, *return_model=False*)

Best-fit qso model determined for Sesar Strip82, ugriz-bands (default r). See additional notes for underlying code qso_engine.

Input: time - measurement times [days] data - measured magnitudes in single filter (also specified) error - uncertainty in measured magnitudes

Output: chi^2/nu - classical variability measure chi^2_qso/nu - fit statistic chi^2_qso/nu_NULL - expected fit statistic for non-qso variable

signif_qso - significance chi^2/nu<chi^2/nu_NULL (rule out false alarm) signif_not_qso - significance chi^2/nu>1 (rule out qso) signif_vary - significance that source is variable at all class - source type (ambiguous, not_qso, qso)

model - time series prediction for each datum given all others (iff return_model==True) dmodel - model uncertainty, including uncertainty in data

Note on use (i.e., how class is defined):

0.signif_vary < 3: ambiguous, else

1.signif_qso > 3: qso, else

2.signif_not_qso > 3: not_qso

2.6.41 scatter_res_raw

mltsp.science_features.**scatter_res_raw**(*t*, *m*, *e*, *lomb_model*)
From arXiv 1101_2406v1 Dubath 20110112 paper.

Scatter: res/raw Median absolute deviation (MAD) of the residuals (obtained by subtracting model values from the raw light curve) divided by the MAD of the raw light-curve values around the median.

2.6.42 skew

mltsp.science_features.**skew**(*x*)
Skewness of a dataset. Approximately 0 for Gaussian data.

2.6.43 std

mltsp.science_features.**std**(*x*)
Standard deviation of observed values.

2.6.44 stetson_j

mltsp.science_features.**stetson_j**(*x*, *y=[]*, *dx=0.1*, *dy=0.1*)
Robust covariance statistic between pairs of observations x,y whose uncertainties are dx,dy. If y is not given, calculates a robust variance for x.

2.6.45 stetson_k

mltsp.science_features.**stetson_k**(*x*, *dx=0.1*)
A robust kurtosis statistic.

2.6.46 weighted_average

mltsp.science_features.**weighted_average**(*x*, *e*)
Arithmetic mean of observed values, weighted by measurement errors.

2.7 Module: build_model

mltsp.build_model.build_model_from_featureset(...)	Build model from (non-rectangular) xarray.Dataset of f
mltsp.build_model.create_and_pickle_model(...)	Build a *scikit-learn* model for the given featureset and s
mltsp.build_model.fit_model_optimize_hyperparams(...)	Optimize estimator hyperparameters.
mltsp.build_model.rectangularize_featureset(...)	Convert xarray.Dataset into (2d) Pandas.DataFrame for

2.7.1 build_model_from_featureset

mltsp.build_model.**build_model_from_featureset**(*featureset*, *model=None*, *model_type=None*, *model_options={}*, *params_to_optimize=None*)
Build model from (non-rectangular) xarray.Dataset of features.

2.7.2 create_and_pickle_model

mltsp.build_model.**create_and_pickle_model**(*featureset*, *model_type*, *output_path*, *model_options={}*, *params_to_optimize=None*)

Build a *scikit-learn* model for the given featureset and store in serialized joblib format at the specified path.

Parameters **featureset** : xarray.Dataset

Features for training model.

model_type : str

Abbreviation of the type of model to be created.

output_path : str

Path where output model is saved as in joblib pickle format.

model_options : dict, optional

Dictionary specifying *scikit-learn* model parameters to be used.

params_to_optimize : dict or list of dict, optional

Dictionary with parameter names as keys and lists of values to try as values, or a list of such dictionaries. Defaults to None.

Returns scikit-learn model

2.7.3 fit_model_optimize_hyperparams

mltsp.build_model.**fit_model_optimize_hyperparams**(*data*, *targets*, *model*, *params_to_optimize*)

Optimize estimator hyperparameters.

Perform hyperparamter optimization using *sklearn.grid_search.GridSearchCV*.

Parameters **data** : Pandas.DataFrame

Features for training model.

targets : Pandas.Series

Targets corresponding to feature vectors in *data*.

model : sklearn estimator object

The model/estimator whose hyperparameters are to be optimized.

params_to_optimize : dict or list of dict

Dictionary with parameter names as keys and lists of values to try as values, or a list of such dictionaries.

Returns *sklearn.grid_search.GridSearchCV* estimator object

2.7.4 rectangularize_featureset

mltsp.build_model.**rectangularize_featureset**(*featureset*)

Convert xarray.Dataset into (2d) Pandas.DataFrame for use with sklearn.

2.8 Module: predict

mltsp.predict.model_predictions(featureset, ...)	Construct a DataFrame of model predictions for given featureset.
mltsp.predict.predict_data_files(ts_paths, ...)	Generate features from new TS data and perform model prediction.

2.8.1 model_predictions

mltsp.predict.**model_predictions**(*featureset*, *model*, *return_probs=True*)

Construct a DataFrame of model predictions for given featureset.

Parameters **featureset** : xarray.Dataset

Dataset containing feature values for which predictions are desired

model : scikit-learn model

Fitted scikit-learn model to be used to generate predictions

return_probs : bool, optional

Parameter to control the type of prediction made in the classification setting (the parameter has no effect for regression models). If True, probabilities for each class are returned where possible; if False, only the top predicted label for each time series is returned.

Returns pandas.DataFrame

DataFrame of model predictions, indexed by *featureset.name*. Each row contains either a single class/target prediction or (for probabilistic predictions) a list of class probabilities.

2.8.2 predict_data_files

mltsp.predict.**predict_data_files**(*ts_paths*, *features_to_use*, *model*, *custom_features_script=None*, *use_docker=True*)

Generate features from new TS data and perform model prediction.

Generates features for new time series file, loads saved estimator model, calculates target predictions with extracted features, and returns a dictionary containing a list of target prediction probabilities, a string containing HTML markup for a table containing a list of the results, the time-series data itself used to generate features, and a dictionary of the features extracted. The respective dict keys of the above-mentioned values are: "pred_results", "results_str", "ts_data", "features_dict".

Parameters **ts_paths** : str

Path to netCDF files containing seriealized TimeSeries objects to be used in prediction.

features_to_use : list of str

List of features to extract for new time series data

model : scikit-learn model

Model to use for making predictions on new input time series

custom_features_script : str, optional

Path to custom features script to be used in feature generation. Defaults to None.

use_docker : bool, optional

Bool specifying whether to generate custom features inside a Docker container. Defaults to True.

Returns dict

Returns dictionary whose keys are the file names of the individual time-series data files used in prediction and whose corresponding values are dictionaries with the following key-value pairs:

- "results_str": String containing table listing results in markup.
- "ts_data": The original time-series data provided.
- "features_dict": A dictionary containing the generated features.
- "pred_results": A list of lists, each containing one of the most-probable targets and its probability.

2.9 Module: `util`

mltsp.util.check_model_param_types(...[, ...])	Cast model parameter strings to expected types.
mltsp.util.docker_images_available()	Return boolean indicating whether Docker images are present.
mltsp.util.extract_time_series(data_path[, ...])	Extract zip- or tarfile of time series file and return file paths.
mltsp.util.get_docker_client([version])	Connect to Docker if available and return a client.
mltsp.util.is_running_in_docker()	Return bool indicating whether running in a Docker container.
mltsp.util.make_list(x)	
mltsp.util.remove_files(paths)	Remove specified files from disk.
mltsp.util.robust_literal_eval(val)	Call *ast.literal_eval* without raising *ValueError*.
mltsp.util.shorten_fname(file_path)	Extract the name of a file (omitting directory names and extensions).
mltsp.util.warn_defaultdict	A recursive *collections.defaultdict*, but with printed warnings when an

2.9.1 check_model_param_types

mltsp.util.**check_model_param_types**(*model_type*, *model_params*, *all_as_lists=False*)

Cast model parameter strings to expected types.

Modifies *model_params* dict in place.

Parameters **model_type** : str

Name of model.

model_params : dict

Dictionary containing model parameters to be checked against expected types.

all_as_lists : bool, optional

Boolean indicating whether *model_params* values are wrapped in lists, as in the case of parameter grids for optimization.

Raises **ValueError**

Raises ValueError if parameter(s) are not of expected type.

2.9.2 docker_images_available

mltsp.util.**docker_images_available**()
Return boolean indicating whether Docker images are present.

2.9.3 extract_time_series

mltsp.util.**extract_time_series**(*data_path*, *cleanup_archive=True*, *cleanup_files=False*, *extract_dir=None*)
Extract zip- or tarfile of time series file and return file paths.

If the given file is not a tar- or zipfile then it is treated as a single time series filepath.

Parameters **data_path** : str

Path to data archive or single data file.

cleanup_archive : bool, optional

Boolean specifying whether to delete the original archive (if applicable). Defaults to True.

cleanup_files : bool, optional

Boolean specifying whether to delete the extracted files when exiting the given context. Defaults to False.

extract_dir : str, optional

Directory into which files are to be extracted (if applicable). If None, a temporary directory is created.

Yields list of str

List of full paths to time series files.

2.9.4 get_docker_client

mltsp.util.**get_docker_client**(*version='1.14'*)
Connect to Docker if available and return a client.

Parameters **version** : str, optional

Protocol version.

Returns docker.Client

Docker client.

Raises **RuntimeError**

If Docker cannot be contacted or contains no images.

2.9.5 is_running_in_docker

mltsp.util.**is_running_in_docker**()
Return bool indicating whether running in a Docker container.

2.9.6 make_list

mltsp.util.**make_list**(*x*)

2.9.7 remove_files

mltsp.util.**remove_files**(*paths*)
Remove specified files from disk.

2.9.8 robust_literal_eval

mltsp.util.**robust_literal_eval**(*val*)
Call *ast.literal_eval* without raising *ValueError*.

Parameters **val** : str

String literal to be evaluated.

Returns Output of *ast.literal_eval(val)', or 'val* if *ValueError* was raised.

2.9.9 shorten_fname

mltsp.util.**shorten_fname**(*file_path*)
Extract the name of a file (omitting directory names and extensions).

2.9.10 warn_defaultdict

class mltsp.util.**warn_defaultdict**
Bases: dict

A recursive *collections.defaultdict*, but with printed warnings when an item is not found.

```
>>> d = warn_defaultdict({1: 2})
>>> d[2][3][4]
[config] WARNING: non-existent key "2" requested
[config] WARNING: non-existent key "3" requested
[config] WARNING: non-existent key "4" requested
```

```
>>> d = warn_defaultdict({'sub': {'a': 'b'}})
>>> print(d['sub']['foo'])
[config] WARNING: non-existent key "foo" requested
{}
```

Methods

clear(() -> None. Remove all items from D.)	
copy(() -> a shallow copy of D)	
fromkeys	Returns a new dict with keys from iterable and values equal to value.
get((k[,d]) -> D[k] if k in D, ...)	
items(...)	

Continued on next page

Table 2.11 – continued from previous page

keys(...)	
pop((k[,d]) -> v, ...)	If key is not found, d is returned if given, otherwise KeyError is raised
popitem(() -> (k, v), ...)	2-tuple; but raise KeyError if D is empty.
setdefault((k[,d]) -> D.get(k,d), ...)	
update(other)	
values(...)	

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

update(*other*)

m

Symbols

A

B

C

D

E

F

G

I

L

M

N

P

Q

R

S

T

U

W