PICA Documentation

Release 0.1.0

Lukas Lüftinger

Oct 28, 2019

CONTENTS:

CHAPTER

ONE

PICA2

Microbial Phenotype Prediction, re-implemented with Python 3.7 and scikit-learn

- Supported platforms: Linux, MacOS, Windows
- Free software: MIT license

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install PICA2, run this command in your terminal:

```
$ pip install pica
```

This is the preferred method to install PICA2, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for PICA2 can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/univieCUBE/PICA2
```

Or download the tarball:

```
$ curl  -OL https://github.com/univieCUBE/PICA2/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use PICA2 in a project:

```
import pica
```

CHAPTER

FOUR

PICA

4.1 pica package

4.1.1 Subpackages

pica.io package

Submodules

pica.io.io module

pica.io.io.**collate_training_data**(*genotype_records: List[pica.structure.records.GenotypeRecord], phenotype_records: List[pica.structure.records.PhenotypeRecord], group_records: List[pica.structure.records.GroupRecord], universal_genotype: bool = False, verb: bool = False*) → List[pica.structure.records.TrainingRecord]

Returns a list of TrainingRecord from two lists of GenotypeRecord and PhenotypeRecord. To be used for training and CV of TrexClassifier. Checks if 1:1 mapping of phenotypes and genotypes exists, and if all PhenotypeRecords pertain to same trait.

Parameters

- **genotype_records** – List[GenotypeRecord]
- **phenotype_records** – List[PhenotypeRecord]
- **group_records** – List[GroupRecord] optional, if leave one group out is the split strategy
- **universal_genotype** – Whether to use an universal genotype file.
- **verb** – toggle verbosity.

Returns List[TrainingRecord]

pica.io.io.**load_genotype_file**(*input_file: str*) → List[pica.structure.records.GenotypeRecord]

Loads a genotype .tsv file and returns a list of GenotypeRecord for each entry.

Parameters **input_file** – The path to the input genotype file.

Returns List[GenotypeRecord] of records in the genotype file

pica.io.io.**load_groups_file**(*input_file: str, selected_rank: str = None*) → List[pica.structure.records.GroupRecord]

Loads a .tsv file which contains group or taxid for each sample in the other training files. Group-Ids may be ncbi-taxon-ids or arbitrary group names. Taxon-Ids are only used if a standard rank is selected, otherwise user-specified group-ids are assumed. Automatically classifies the [TODO missing text?]

Parameters

- **input_file** – path to the file that is processed
- **selected_rank** – the standard rank that is selected (optional) if not set, the input file is assumed to contain groups, i.e., each unique entry of the ID will be a new group

Returns a list of GroupRecords

pica.io.io.**load_phenotype_file**(*input_file: str*, *sign_mapping: Dict[str, int] = None*) → List[pica.structure.records.PhenotypeRecord]

Loads a phenotype .tsv file and returns a list of PhenotypeRecord for each entry.

Parameters

- **input_file** – The path to the input phenotype file.
- **sign_mapping** – an optional Dict to change mappings of trait sign. Default: {"YES": 1, "NO": 0}

Returns List[PhenotypeRecord] of records in the phenotype file

pica.io.io.**load_training_files**(*genotype_file: str*, *phenotype_file: str*, *groups_file: str = None*, *selected_rank: str = None*, *universal_genotype: bool = False*, *verb=False*) → Tuple[List[pica.structure.records.TrainingRecord], List[pica.structure.records.GenotypeRecord], List[pica.structure.records.PhenotypeRecord], List[pica.structure.records.GroupRecord]]

Convenience function to load phenotype and genotype file together, and return a list of TrainingRecord.

Parameters

- **genotype_file** – The path to the input genotype file.
- **phenotype_file** – The path to the input phenotype file.
- **groups_file** – The path to the input groups file.
- **selected_rank** – The selected standard rank to use for taxonomic grouping
- **universal_genotype** – Whether to use an universal genotype file.
- **verb** – toggle verbosity.

Returns Tuple[List[TrainingRecord], List[GenotypeRecord], List[PhenotypeRecord]]

pica.io.io.**write_cccv_accuracy_file**(*output_file: str*, *cccv_results*)

Function to write the cccv accuracies in the exact format that phendb uses as input.

Parameters

- **output_file** – file
- **cccv_results** –

Returns nothing

pica.io.io.**write_misclassifications_file**(*output_file: str, records: List[pica.structure.records.TrainingRecord], misclassifications, use_groups: bool = False*)

Function to write the misclassifications file.

Parameters

- **output_file** – name of the outputfile

- **records** – List of trainingRecord objects
- **misclassifications** – List of percentages of misclassifications
- **use_groups** – toggles average over groups and groups output

Returns

pica.io.io.**write_weights_file**(*weights_file: str*, *weights: Dict*)

Function to write the weights to specified file in tab-separated fashion with header

Parameters

- **weights_file** – The path to the file to which the output will be written
- **weights** – sorted dictionary storing weights with feature names as indices

Returns nothing

Module contents

pica.ml package

Subpackages

pica.ml.classifiers package

Submodules

pica.ml.classifiers.svm module

pica.ml.classifiers.xgbm module

Module contents

Submodules

pica.ml.cccv module

class pica.ml.cccv.**CompleContaCV**(*pipeline: sklearn.pipeline.Pipeline*, *scoring_function: Callable = <function balanced_accuracy_score>*, *cv: int = 5*, *comple_steps: int = 20*, *conta_steps: int = 20*, *n_jobs: int = -1*, *n_replicates: int = 10*, *random_state: numpy.random.mtrand.RandomState = None*, *verb: bool = False*, *reduce_features: bool = False*, *n_features: int = 10000*)

Bases: object

A class containing all custom completeness/contamination cross-validation functionality.

Parameters

- **pipeline** – target pipeline which describes the vectorization and estimator/classifier used
- **scoring_function** – Sklearn-like scoring function of crossvalidation. Default: Balanced Accuracy.

- **cv** – Number of folds in crossvalidation. Default: 5
- **comple_steps** – number of steps between 0 and 1 (relative completeness) to be simulated
- **conta_steps** – number of steps between 0 and 1 (relative contamination level) to be simulated
- **n_jobs** – Number of parallel jobs. Default: -1 (All processors used)
- **n_replicates** – Number of times the crossvalidation is repeated
- **reduce_features** – toggles feature reduction using recursive feature elimination
- **n_features** – minimal number of features to retain (if feature reduction is used)
- **random_state** – An integer random seed or instance of np.random.RandomState

run(*records: List[pica.structure.records.TrainingRecord]*)
Perform completeness/contamination cross-validation.

Parameters **records** – List[TrainingRecords] to perform compleconta-crossvalidation on.

Returns A dictionary with mean balanced accuracies for each combination: dict[comple][conta]=mba

pica.ml.feature_select module

pica.ml.feature_select.**compress_vocabulary**(*records: List[pica.structure.records.TrainingRecord], pipeline: sklearn.pipeline.Pipeline*)
Method to group features, that store redundant information, to avoid overfitting and speed up process (in some cases). Might be replaced or complemented by a feature selection method in future versions.

Compressing vocabulary is optional, for the test dataset it took 30 seconds, while the time saved later on is not significant.

Parameters

- **records** – a list of TrainingRecord objects.
- **pipeline** – the targeted pipeline where the vocabulary should be modified

Returns nothing, sets the vocabulary for CountVectorizer step

pica.ml.feature_select.**multiple_step_rfecv**(*records: List[pica.structure.records.TrainingRecord], pipeline: sklearn.pipeline.Pipeline, n_features: int, step=(0.01, 0.01, 0.01), random_state: numpy.random.mtrand.RandomState = None*)
Function to apply multiple steps-sizes of RFECV in series, currently not used. Strategy might be problematic, no clear benefit. #TODO rethink or remove

Parameters

- **records** – Data used
- **pipeline** – The base estimator used
- **n_features** – Goal number of features
- **step** – List of steps that should be applied
- **random_state** – random state for deterministic results

Returns

pica.ml.feature_select.**recursive_feature_elimination**(*records: List[pica.structure.records.TrainingRecord], pipeline: sklearn.pipeline.Pipeline, step: float = 0.0025, n_features: int = None, random_state: numpy.random.mtrand.RandomState = None*)

Function to apply RFE to limit the vocabulary used by the CustomVectorizer, optional step.

Parameters

- **records** – list of TrainingRecords, entire training set.
- **pipeline** – the pipeline which vocabulary should be modified
- **step** – rate of features to eliminate at each step. the lower the number, the more steps
- **n_features** – number of features to select (if None: half of the provided features)
- **random_state** – random state for deterministic results

Returns number of features used

pica.ml.trex_classifier module

class pica.ml.trex_classifier.**TrexClassifier**(*random_state: int = None, verb: bool = False*)

Bases: abc.ABC

Abstract base class of Trex classifier.

crossvalidate(*records: List[pica.structure.records.TrainingRecord], cv: int = 5, scoring: Union[str, Callable] = 'balanced_accuracy', n_jobs=-1, n_replicates: int = 10, groups: bool = False, reduce_features: bool = False, n_features: int = 10000, demote=False, **kwargs*) → Tuple[float, float, numpy.ndarray]

Perform cv-fold crossvalidation or leave-one(-group)-out validation if groups == True

Parameters

- **records** – List[TrainingRecords] to perform crossvalidation on.
- **scoring** – String identifying scoring function of crossvalidation, or Callable. If a callable is passed, it must take two parameters *y_true* and *y_pred* (iterables of true and predicted class labels, respectively) and return a (numeric) score.
- **cv** – Number of folds in crossvalidation. Default: 5
- **n_jobs** – Number of parallel jobs. Default: -1 (All processors used)
- **n_replicates** – Number of replicates of the crossvalidation
- **groups** – If True, use group information stored in records for splitting. Otherwise, stratify split according to labels in records. This also resets n_replicates to 1.
- **reduce_features** – toggles feature reduction using recursive feature elimination
- **n_features** – minimum number of features to retain when reducing features
- **demote** – toggles logger that is used. if true, msg is written to debug else info
- **kwargs** – Unused

Returns A list of mean score, score SD, and the percentage of misclassifications per sample

crossvalidate_cc(*records: List[pica.structure.records.TrainingRecord], cv: int = 5, comple_steps: int = 20, conta_steps: int = 20, n_jobs: int = -1, n_replicates: int = 10, reduce_features: bool = False, n_features: int = 10000*)

Instantiates a CompleContaCV object, and calls its run_cccv method with records. Returns its result.

Parameters

- **records** – List[TrainingRecord] on which completeness_contamination_CV is to be performed
- **cv** – number of folds in StratifiedKFold split
- **comple_steps** – number of equidistant completeness levels
- **conta_steps** – number of equidistant contamination levels
- **n_jobs** – number of parallel jobs (-1 for n_cpus)
- **n_replicates** – Number of times the crossvalidation is repeated
- **reduce_features** – toggles feature reduction using recursive feature elimination
- **n_features** – selects the minimum number of features to retain (if feature reduction is used)

Returns A dictionary with mean balanced accuracies for each combination: dict[comple][conta]=mba

abstract get_feature_weights() → Dict

Extract the weights for features from pipeline.

Returns sorted Dict of feature name: weight

classmethod get_instance(**args*, ***kwargs*)

parameter_search(*records: List[pica.structure.records.TrainingRecord], search_params: Dict[str, List] = None, cv: int = 5, scoring: str = 'balanced_accuracy', n_jobs: int = -1, n_iter: int = 10, return_optimized: bool = False*)

Perform stratified, randomized parameter search. If desired, return a new class instance with optimized training parameters.

Parameters

- **records** – List[TrainingRecords] to perform crossvalidation on.
- **search_params** – A dictionary of iterables of possible model training parameters. If None, use default search parameters for the given classifier.
- **scoring** – Scoring function of crossvalidation. Default: Balanced Accuracy.
- **cv** – Number of folds in crossvalidation. Default: 5
- **n_jobs** – Number of parallel jobs. Default: -1 (All processors used)
- **n_iter** – Number of grid points to evaluate. Default: 10
- **return_optimized** – Whether to return a ready-made classifier with the optimized params instead of a dictionary of params.

Returns A dictionary containing best found parameters or an optimized class instance.

predict(*X: List[pica.structure.records.GenotypeRecord]*) → Tuple[List[str], numpy.ndarray]

Predict trait sign and probability of each class for each supplied GenotypeRecord.

Parameters **X** – A List of GenotypeRecord for each of which to predict the trait sign

Returns a Tuple of predictions and probabilities of each class for each GenotypeRecord in X.

scoring_function_mapping = {'accuracy': <function accuracy_score>, 'balanced_accuracy

train(*records: List[pica.structure.records.TrainingRecord], reduce_features: bool = False, n_features: int = 10000, **kwargs*)

Fit CountVectorizer and train LinearSVC on a list of TrainingRecord.

Parameters

- **records** – a List[TrainingRecord] for fitting of CountVectorizer and training of LinearSVC.
- **reduce_features** – toggles feature reduction using recursive feature elimination
- **n_features** – minimum number of features to retain when reducing features
- **kwargs** – additional named arguments are passed to the fit() method of Pipeline.

Returns Whether the Pipeline has been fitted on the records.

pica.ml.vectorizer module

class pica.ml.vectorizer.**CustomVectorizer**(*input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>*)

Bases: sklearn.feature_extraction.text.CountVectorizer

modified from CountVectorizer to override the _validate_vocabulary function, which invoked an error because multiple indices of the dictionary contained the same feature index. However, this is we intend with the compress_vocabulary function. Other functions had to be adopted: get_feature_names (allow decompression), _count_vocab (reduce the matrix size)

get_feature_names()

Array mapping from feature integer indices to feature name

Module contents

pica.structure package

Submodules

pica.structure.records module

class pica.structure.records.**GenotypeRecord**(*identifier: str, features: List[str]*)

Bases: object

TODO add docstring

class pica.structure.records.**GroupRecord**(*identifier: str, group_name: str, group_id: int*)

Bases: object

TODO add docstring

class pica.structure.records.**PhenotypeRecord**(*identifier: str, trait_name: str, trait_sign: int*)

Bases: object

TODO add docstring

class pica.structure.records.**TrainingRecord**(*identifier: str, group_name: str, group_id: int, trait_name: str, trait_sign: int, features: List[str]*)

Bases: *pica.structure.records.GenotypeRecord*, *pica.structure.records.PhenotypeRecord*, *pica.structure.records.GroupRecord*

TODO add docstring

Module contents

pica.transforms package

Submodules

pica.transforms.resampling module

class pica.transforms.resampling.**TrainingRecordResampler**(*random_state: float = None, verb: bool = False*)

Bases: object

Instantiates an object which can generate versions of a TrainingRecord resampled to defined completeness and contamination levels. Requires prior fitting with full List[TrainingRecord] to get sources of contamination for both classes.

Parameters

- **random_state** – Randomness seed to use while resampling
- **verb** – Toggle verbosity

fit(*records: List[pica.structure.records.TrainingRecord]*)

Fit TrainingRecordResampler on full TrainingRecord list to determine set of positive and negative features for contamination resampling.

Parameters **records** – the full List[TrainingRecord] on which ml training will commence.

Returns True if fitting was performed, else False.

get_resampled(*record: pica.structure.records.TrainingRecord, comple: float = 1, conta: float = 0*) → pica.structure.records.TrainingRecord

Resample a TrainingRecord to defined completeness and contamination levels. Comple=1, Conta=1 will double set size.

Parameters

- **comple** – completeness of returned TrainingRecord features. Range: 0 - 1
- **conta** – contamination of returned TrainingRecord features. Range: 0 - 1
- **record** – the input TrainingRecord

Returns a resampled TrainingRecord.

Module contents

pica.util package

Submodules

pica.util.helpers module

pica.util.helpers.**get_groups**(*records: List[pica.structure.records.TrainingRecord]*) → numpy.ndarray

Get groups from list of TrainingRecords

Parameters **records** –

Returns list for groups

pica.util.helpers.**get_x_y_tn**(*records: List[pica.structure.records.TrainingRecord]*) → Tuple[numpy.ndarray, numpy.ndarray, str]

Get separate X and y from list of TrainingRecord. Also infer trait name from first TrainingRecord.

Parameters **records** – a List[TrainingRecord]

Returns separate lists of features and targets, and the trait name

pica.util.logging module

pica.util.logging.**get_logger**(*initname*, *verb=False*)

This function provides a logger to all scripts used in this project.

Parameters

- **initname** – The name of the logger to show up in log.
- **verb** – Toggle verbosity

Returns the finished Logger object.

pica.util.plotting module

pica.util.plotting.**compleconta_plot**(*cccv_results: List[Dict[float, Dict[float, Dict[str, float]]]], conditions: List[str] = (), each_n: List[int] = None, title: str = '', fontsize: int = 16, figsize=(10, 7), plot_comple: bool = True, plot_conta: bool = True, colors: List = None, save_path: Union[str, pathlib.Path] = None, **kwargs*)

Plots Compleconta CV result for one or multiple models. For perfect completeness and variable contamination as well as perfect contamination and variable completeness, the resulting mean balanced accuracy over folds is plotted.

Parameters

- **cccv_results** – a ComplecontaCV result, or list thereof
- **conditions** – A list of condition names associated cccv_results
- **each_n** – A list of sample counts in datasets associated with cccv_results
- **title** – The plot title

- **fontsize** – The fontsize of the plot
- **figsize** – The figure size (tuple of width, height)
- **plot_comple** – Whether to plot completeness
- **plot_conta** – Whether to plot contamination
- **colors** –
- **save_path** – The save path of the plot; if None, display it with plt.show()
- **kwargs** – any further keyword arguments passed to plt.plot()

Returns None

pica.util.serialization module

pica.util.serialization.**load_classifier**(*filename: str*, *verb=False*)
Load a pickled TrexClassifier to a usable object.

Parameters

- **filename** – Input filename
- **verb** – Toggle verbosity

Returns a unpickled PICA ml classifier

pica.util.serialization.**save_classifier**(*obj*, *filename: str*, *overwrite=False*, *verb=False*)
Save a TrexClassifier as a pickled object.

Parameters

- **obj** – the Python3 object to be saved.
- **filename** – Output filename
- **overwrite** – Overwrite existing files with same name
- **verb** – Toggle verbosity

pica.util.taxonomy module

Module contents

4.1.2 Submodules

4.1.3 pica.run_pica module

4.1.4 Module contents

Top-level package for PICA.

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Lukas Lüftinger <lukas.lueftinger@outlook.com>

5.2 Contributors

- Patrick Hyden <hydenp89@univie.ac.at>
- Roman Feldbauer <roman.feldbauer@univie.ac.at>

CHAPTER

SIX

HISTORY

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX

T

W