MIML Library

Damian Martinez

Jun 03, 2024

CONTENTS

CHAPTER

ONE

CLASSIFIER

1.1 mi

1.1.1 APRClassifier

miml.classifier.mi.apr_classifier.APRClassifier	Classifier for All-Positive Bags using Axis-Aligned Positive Region.

miml.classifier.mi.apr_classifier.APRClassifier

class miml.classifier.mi.apr_classifier.**APRClassifier**

Bases: object

Classifier for All-Positive Bags using Axis-Aligned Positive Region.

This classifier assigns a positive label to bags that contain instances within a predefined axis-parallel rectangle (APR) defined by the minimum and maximum feature values of positive instances in the training set.

Attributes

apr

[list[np.ndarray of shape (n_labels)]] List containing the minimum and maximum feature values defining the APR.

References

Dietterich, Thomas G., Richard H. Lathrop, and Tomás Lozano-Pérez. "Solving the multiple instance problem with axis-parallel rectangles." Artificial intelligence 89.1 (1997): 31-71.

fit(*x_train: ndarray*, *y_train: ndarray*) → None

Fit the classifier to the training data.

Parameters

x_train
[ndarray of shape (n_bags, n_instances, n_features)] Features values of bags in the training set.

y_train
[ndarray (n_bags, n_instances, n_labels)] Labels of bags in the training set.

predict(*bag: ndarray*) → int

Predict the label of the bag

Parameters

bag: np.ndarray of shape(n_instances, n_features)
Features values of a bag

Returns

label: int
Predicted label of the bag

predict_proba(*x_test: ndarray*) → ndarray

Predict probabilities of given data of having a positive label

Parameters

x_test
[np.ndarray of shape (n_bags, n_instances, n_features)] Data to predict probabilities

Returns

results: np.ndarray of shape (n_instances)
Predicted probabilities for given data

1.1.2 MIWrapperClassifier

miml.classifier.mi.mi_wrapper_classifier.MIWrapperClassifier	MIWrapper Classifier.

miml.classifier.mi.mi_wrapper_classifier.MIWrapperClassifier

class miml.classifier.mi.mi_wrapper_classifier.**MIWrapperClassifier**(*base_classifier=DecisionTreeClassifier()*)

Bases: object

MIWrapper Classifier.

A simple Wrapper method for applying standard propositional learners to multi-instance data.

Attributes

base_classifier
Classifier to be used

References

E. T. Frank, X. Xu (2003). Applying propositional learning algorithms to multi-instance data. Department of Computer Science, University of Waikato, Hamilton, NZ.

fit(*x_train: ndarray*, *y_train: ndarray*, *weight: int = 2*)

Fit the classifier to the training data.

Parameters

x_train
[ndarray of shape (n_bags, n_instances, n_features)] Features values of bags in the training set.

y_train
[ndarray (n_bags, n_instances, n_labels)] Labels of bags in the training set.

weight
[int, default = 2]

The type of weight setting for each single-instance:
0: weight = 1.0 1: weight = 1.0/Total number of single-instance in the corresponding bag 2: weight = Total number of single-instance / (Total number of bags * Total number of single-instance in the corresponding bag).

predict(*bag: ndarray*) → int

Predict the label of the bag

Parameters

bag: np.ndarray of shape(n_instances, n_features)
Features values of a bag

Returns

label: int
Predicted label of the bag

predict_proba(*x_test: ndarray*) → ndarray
Predict probabilities of given data of having a positive label

Parameters

x_test
[np.ndarray of shape (n_bags, n_instances, n_features)] Data to predict probabilities

Returns

results: np.ndarray of shape (n_instances, n_labels)
Predicted probabilities for given data

1.2 MIMLClassifier

miml.classifier.miml_classifier.MIMLClassifier	Class to represent a MIMLClassifier

1.2.1 miml.classifier.miml_classifier.MIMLClassifier

class miml.classifier.miml_classifier.**MIMLClassifier**
Bases: ABC

Class to represent a MIMLClassifier

abstract evaluate(*dataset_test:* MIMLDataset) → ndarray
Evaluate the model on a test dataset

Parameters

dataset_test
[MIMLDataset] Test dataset to evaluate the model on.

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of dataset_test

fit(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

abstract fit_internal(*dataset_train:* MIMLDataset) → None
Internal method to train the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

abstract predict(*x: ndarray*) → ndarray

Predict labels of given data

x
[ndarray of shape (n, n_labels)] Data to predict their labels

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of data

abstract predict_bag(*bag:* Bag) → ndarray
Predict labels of a given bag

Parameters

bag
[Bag] Bag to predict their labels

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of the bag

abstract predict_proba(*dataset_test:* MIMLDataset) → ndarray
Predict probabilities of given dataset of having a positive label

Parameters

dataset_test
[MIMLDataset] Dataset to predict probabilities

Returns

results: np.ndarray of shape (n_instances, n_features)
Predicted probabilities for given dataset

1.3 mimlTOml

1.3.1 MIMLtoMLClassifier

miml.classifier.mimlTOml.
miml_to_ml_classifier.MIMLtoMLClassifier

miml.classifier.mimlTOml.miml_to_ml_classifier.MIMLtoMLClassifier

class miml.classifier.mimlTOml.miml_to_ml_classifier.**MIMLtoMLClassifier**(*ml_classifier*, *transformation:* MIMLtoMLTransformation)

Bases: *MIMLClassifier*

evaluate(*dataset_test:* MIMLDataset) → ndarray
Evaluate the model on a test dataset

Parameters

dataset_test
[MIMLDataset] Test dataset to evaluate the model on

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of dataset_test

fit(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

`fit_internal`(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

`predict`(*x: ndarray*) → ndarray
Predict labels of given data

Parameters

x
[ndarray of shape (n_instances, n_labels)] Data to predict their labels

Returns

results
[ndarray of shape (n_instances, n_labels)] Predicted labels of data

`predict_bag`(*bag:* Bag) → ndarray
Predict labels of a given bag

Parameters

bag
[Bag] Bag to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels of data

`predict_proba`(*dataset_test:* MIMLDataset) → ndarray
Predict probabilities of given dataset of having a positive label

Parameters

dataset_test
[MIMLDataset] Dataset to predict probabilities

Returns

results: np.ndarray of shape (n_instances, n_features)
Predicted probabilities for given dataset

1.4 mimlTOmi

1.4.1 MIMLtoMIClassifier

miml.classifier.mimlTOmi.miml_to_mi_classifier.MIMLtoMIClassifier	Class to represent a multi-instance classifier

miml.classifier.mimlTOmi.miml_to_mi_classifier.MIMLtoMIClassifier

class miml.classifier.mimlTOmi.miml_to_mi_classifier.**MIMLtoMIClassifier**(*mi_classifier*)

Bases: *MIMLClassifier*

Class to represent a multi-instance classifier

evaluate(*dataset_test:* MIMLDataset) → ndarray
Evaluate the model on a test dataset

Parameters

dataset_test
[MIMLDataset] Test dataset to evaluate the model on

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of dataset_test

fit(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

abstract fit_internal(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train: MIMLDataset
Dataset to train the classifier

abstract predict(*x: ndarray*) → ndarray
Predict labels of given data

Parameters

x
[ndarray of shape (n_instances, n_labels)] Data to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels

predict_bag(*bag:* Bag) → ndarray
Predict labels of a given bag

Parameters

bag
[Bag] Bag to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels of the bag

abstract predict_proba(*dataset_test:* MIMLDataset) → ndarray
Predict probabilities of given dataset of having a positive label

Parameters

dataset_test
[MIMLDataset] Dataset to predict probabilities

Returns

results: np.ndarray of shape (n_instances, n_features)
Predicted probabilities for given dataset

1.4.2 MIMLtoMIBRClassifier

miml.classifier.mimlTOmi.miml_to_mi_br_classifier.MIMLtoMIBRClassifier	Class to represent a multi-instance classifier using a binary relevance transformation

miml.classifier.mimlTOmi.miml_to_mi_br_classifier.MIMLtoMIBRClassifier

class miml.classifier.mimlTOmi.miml_to_mi_br_classifier.**MIMLtoMIBRClassifier**(*mi_classifier*)

Bases: *MIMLtoMIClassifier*

Class to represent a multi-instance classifier using a binary relevance transformation

evaluate(*dataset_test:* MIMLDataset) → ndarray

Evaluate the model on a test dataset

Parameters

dataset_test
[MIMLDataset] Test dataset to evaluate the model on

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of dataset_test

fit(*dataset_train:* MIMLDataset) → None

Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

`fit_internal`(*dataset_train:* MIMLDataset) → None
Training the classifier

Parameters

dataset_train: MIMLDataset
Dataset to train the classifier

`predict`(*x: ndarray*) → ndarray
Predict labels of given data

Parameters

x
[ndarray of shape (n_instances, n_labels)] Data to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels

`predict_bag`(*bag:* Bag) → ndarray
Predict labels of a given bag

Parameters

bag
[Bag] Bag to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels of the bag

`predict_proba`(*dataset_test:* MIMLDataset)

Predict probabilities of given dataset of having a positive label

Parameters

dataset_test
[MIMLDataset] Dataset to predict probabilities

results: np.ndarray of shape (n_instances, n_labels)
Predicted probabilities for given dataset

1.4.3 MIMLtoMILPClassifier

miml.classifier.mimlTOmi.miml_to_mi_lp_classifier.MIMLtoMILPClassifier	Class to represent a multi-instance classifier using a label powerset transformation

miml.classifier.mimlTOmi.miml_to_mi_lp_classifier.MIMLtoMILPClassifier

class miml.classifier.mimlTOmi.miml_to_mi_lp_classifier.**MIMLtoMILPClassifier**(*mi_classifier*)

Bases: *MIMLtoMIClassifier*

Class to represent a multi-instance classifier using a label powerset transformation

evaluate(*dataset_test:* MIMLDataset) → ndarray

Evaluate the model on a test dataset

Parameters

dataset_test
[MIMLDataset] Test dataset to evaluate the model on

Returns

results
[ndarray of shape (n_bags, n_labels)] Predicted labels of dataset_test

fit(*dataset_train:* MIMLDataset) → None

Training the classifier

Parameters

dataset_train
[MIMLDataset] Dataset to train the classifier

fit_internal(*dataset_train:* MIMLDataset) → None

Training the classifier

Parameters

dataset_train: MIMLDataset
Dataset to train the classifier

predict(*x: ndarray*) → ndarray
Predict labels of given data

Parameters

x
[ndarray of shape (n_instances, n_features)] Data to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels

predict_bag(*bag:* Bag) → ndarray
Predict labels of a given bag

Parameters

bag
[Bag] Bag to predict their labels

Returns

results
[ndarray of shape (n_labels)] Predicted labels of the bag

predict_proba(*dataset_test:* MIMLDataset)
Predict probabilities of given dataset of having a positive label

Parameters

dataset_test
[MIMLDataset] Dataset to predict probabilities

Returns

results: np.ndarray of shape (n_instances, n_features)
Predicted probabilities for given dataset

CHAPTER

TWO

DATA

2.1 Instance

miml.data.instance.Instance	Class to manage MIML Instance data representation

2.1.1 miml.data.instance.Instance

class miml.data.instance.**Instance**(*values: list | None = None*, *bag=None*)

Bases: object

Class to manage MIML Instance data representation

add_attribute(*value=0*, *position=None*) → None

Add an attribute to the instance

Parameters

value
[float, default=0] Value for the attribute

position: int, default=None
Position for the attribute

delete_attribute(*position*) → None

Delete an attribute of the instance

Parameters

position: int
Position of the attribute

get_attribute(*attribute*) → float

Get value of an attribute of the instance

Parameters

attribute
[int/str] Index/Name of the attribute

Returns

value
[float] Value of the attribute

`get_attributes()` → ndarray

Get data attributes of the instance

Returns

attributes data: ndarray of shape (n_attributes)
Values of the attributes of the instance

`get_attributes_name()` → list[str]

Get attributes name of the instance

Returns

attributes
[list[str]] Attributes name of the instance

`get_features()` → ndarray

Get features values of the instance

Returns

features data: ndarray of shape (n_features)
Values of the features of the instance

`get_features_name()` → list[str]

Get features name of the instance

Returns

features
[list[str]] features name of the instance

`get_labels()` → ndarray

Get labels values of the instance

Returns

labels data
[ndarray of shape (n_labels)] Values of the labels of the instance

`get_labels_name()` → list[str]

Get labels name of the instance

Returns

labels
[list[str]] Labels name of the instance

`get_number_attributes()` → int

Get numbers of attributes of the instance

Returns

numbers of attributes: int
Numbers of attributes of the instance

`get_number_features()` → int

Get numbers of features of the instance

Returns

numbers of features: int
Numbers of features of the instance

`get_number_labels()` → int

Get numbers of labels of the instance

Returns

numbers of labels
[int] Numbers of labels of the instance

`set_attribute`(*attribute*, *value: float*) → None

Update value of an attribute of the instance

Parameters

attribute
[int/str] Index/Name of the attribute

value
[float] New value for the attribute

`set_bag`(*bag*) → None
Set the bag of the instance

Parameters

bag
[Bag] Bag of the instance

`show_instance`() → None
Show instance info in table format

2.2 Bag

miml.data.bag.Bag	Class to manage MIML Bag data representation

2.2.1 miml.data.bag.Bag

class miml.data.bag.**Bag**(*key: str*)
Bases: `object`

Class to manage MIML Bag data representation

`add_attribute`(*position: int*, *values=None*) → None
Add attribute to the bag

Parameters

position
[int] Index for the new attribute

values: array-like of shape (n_attributes)
Values for the new attribute. If not provided, new values would be zero

`add_instance`(*instance:* Instance) → None
Add instance to the bag

Parameters

instance
[Instance] Instance to be added

`delete_attribute`(*position: int*) → None
Delete attribute of the bag

Parameters

position
[int] Position of the attribute in the bag

`delete_instance`(*index: int*) → None
Delete an instance of the bag

Parameters

index
[int] Index of the instance to be removed

`get_attribute`(*instance: int*, *attribute*) → float
Get value of an attribute of the bag

Parameters

instance
[int] Index of the instance in the bag

attribute
[int/str] Index/Name of the attribute

Returns

value
[float] Value of the attribute

`get_attributes`() → ndarray
Get attributes values of the bag

Returns

attributes data: ndarray of shape(n_instances, n_attributes)
Values of the attributes of the bag

`get_attributes_name`() → list[str]
Get attributes name of the bag

Returns

attributes
[list[str]] Attributes name of the bag

`get_features()` → ndarray
Get features values of the bag

Returns

features data: ndarray of shape (n_instances, n_features)
Values of the features of the bag

`get_features_name()` → list[str]
Get features name of the bag

Returns

features
[list[str]] Features name of the bag

`get_instance(index: int)` → *Instance*
Get an Instance of the Bag

Parameters

index
[int] Index of the instance in the bag

Returns

instance
[Instance] Instance of Instance class

`get_labels()` → ndarray
Get labels values of the bag

Returns

labels data
[ndarray of shape (n_instances, n_labels)] Values of the labels of the bag

`get_labels_name()` → list[str]
Get labels name of the bag

Returns

labels
[list[str]] Labels name of the bag

`get_number_attributes()` → int
Get numbers of attributes of the bag

Returns

numbers of attributes: int
Numbers of attributes of the bag

`get_number_features()` → int
Get numbers of features of the bag

Returns

numbers of features: int
Numbers of features of the bag

`get_number_instances()` → int
Get numbers of instances of a bag

Returns

numbers of instances: int
Numbers of instances of a bag

`get_number_labels()` → int
Get numbers of labels of the bag

Returns

numbers of labels: int
Numbers of labels of the bag

`set_attribute`(*instance: int*, *attribute*, *value: float*) → None
Update value from attributes

Parameters

instance
[int] Index of instance to be updated

attribute: int/str
Attribute name/index_instance of the bag to be updated

value: float
New value for the update

`set_dataset`(*dataset*) → None
Set dataset which contains the bag

Parameters

dataset
[MIMLDataset] Dataset for the bag

`show_bag`(*start: int = 0*, *end: int | None = None*, *attributes: list[str] | None = None*, *mode='table'*) → None
Show bag info in table format

Parameters

start
[int] Index of instance to start showing

end
[int] Index of instance to end showing

mode
[str] Mode to show the bag. Modes available are "table" and "csv" (csv format)

attributes
[list[str]] List of attributes to display. If empty, all attributes will be displayed.

2.3 MIMLDataset

miml.data.miml_dataset.MIMLDataset	Class to manage MIML data obtained from datasets

2.3.1 miml.data.miml_dataset.MIMLDataset

class miml.data.miml_dataset.**MIMLDataset**
Bases: `object`

Class to manage MIML data obtained from datasets

`add_attribute`(*name: str*, *position: int | None = None*, *values: ndarray | None = None*, *feature: bool = True*) → None
Add attribute to the dataset

Parameters

name
[str] Name of the new attribute

position
[int, default = None] Index for the new attribute

values: ndarray of shape(n_instances)
Values for the new attribute

feature
[bool] Boolean value to determine if the attribute added is a feature or a label

`add_bag`(*bag:* Bag) → None
Add a bag to the dataset

Parameters

bag
[Bag] Instance of Bag class to be added

`add_instance`(*bag*, *instance:* Instance) → None
Add an Instance to a Bag of the dataset

Parameters

bag
[int/str] Index or key of the bag where the instance will be added

instance
[Instance] Instance of Instance class to be added

`cardinality`()
Computes the Cardinality as the average number of labels per pattern.

Returns

cardinality
[float] Average number of labels per pattern

`delete_attribute`(*position: int*) → None
Delete attribute of the dataset

Parameters

position
[int] Index of the attribute to be deleted

`delete_bag`(*key_bag: str*) → None
Delete a bag of the dataset

Parameters

key_bag
[str] Key of the bag to be deleted

`delete_instance`(*bag*, *index_instance: int*) → None
Delete an instance of a bag of the dataset

Parameters

bag
[int/str] Index or key of the bag which contains the instance to be deleted

index_instance
[int] Index of the instance to be deleted

`density`()
Computes the density as the cardinality / numLabels.

Returns

density
[float] Cardinality divided by number of labels

`describe`()
Print statistics about the dataset

`distinct`()
Computes the numbers of labels combinations used in the dataset respect all the possible ones

Returns

distinct
[float] Numbers of labels combinations used in the dataset divided by all possible combinations

`get_attribute`(*bag*, *instance*, *attribute*) → float
Get value of an attribute of the bag

Parameters

bag
[str] Key of the bag which contains the attribute

instance
[int] Index of the instance in the bag

attribute
[int/str] Index/Name of the attribute

Returns

value
[float] Value of the attribute

`get_attributes()` → ndarray

Get attributes values of the dataset

Returns

attributes data: ndarray of shape (n_instances, n_attributes)
Values of the attributes of the dataset

`get_attributes_name()` → list[str]

Get attributes name

Returns

attributes
[list[str]] Attributes name of the dataset

`get_bag(bag)` → *Bag*

Get data of a bag of the dataset

Parameters

bag: int/str
Index or key of the bag to be obtained

Returns

bag: Bag
Instance of Bag class

`get_features()` → ndarray

Get features values of the dataset

Returns

features: ndarray of shape (n_instances, n_features)
Values of the features of the dataset

`get_features_by_bag()` → ndarray

Get features values of the dataset by bag

Returns

features: ndarray of shape (n_bags, n_instances, n_features)
Values of the features of the dataset

`get_features_name()` → list[str]

Get function for dataset features name

Returns

attributes
[list[str]] Attributes name of the dataset

`get_instance`(*bag*, *index_instance*) → *Instance*

Get an Instance of the dataset

Parameters

bag
[int/str] Index/Key of the bag

index_instance
[int] Index of the instance in the bag

Returns

instance
[Instance] Instance of Instance class

`get_labels()`

Get labels values of the dataset

Returns

labels: ndarray of shape (n_instances, n_labels)
Values of the labels of the dataset

`get_labels_by_bag()`

Get labels values of the dataset

Returns

labels
[ndarray of shape (n_bags, n_labels)] Values of the labels of the dataset

`get_labels_name()` → list[str]

Get function for dataset labels name

Returns

labels
[list[str]] Labels name of the dataset

`get_name()` → str

Get function for dataset name

Returns

name
[str] Name of the dataset

`get_number_attributes()` → int

Get numbers of attributes of the bag

Returns

numbers of attributes: int
Numbers of attributes of the bag

`get_number_bags()` → int

Get numbers of bags of the dataset

Returns

numbers of bags: int
Numbers of bags of the dataset

`get_number_features()` → int

Get numbers of attributes of the dataset

Returns

numbers of attributes: int
Numbers of attributes of the dataset

`get_number_instances()` → int

Get numbers of instances of the dataset

Returns

numbers of instances: int
Numbers of instances of the dataset

`get_number_labels()` → int
Get numbers of labels of the dataset

Returns

numbers of labels: int
Numbers of labels of the dataset

`get_statistics()`
Calculate statistics of the dataset

Returns

n_instances
[int] Numbers of instances of the dataset

min_instances
[int] Number of instances in the bag with minimum number of instances

max_instances
[int] Number of instances in the bag with maximum number of instances

distribution
[dict] Distribution of number of instances in bags

`save_arff`(*path*) → None
Save MIMLDataset as arff file

Parameters

path
[str] Path to store the arff file

`save_csv`(*path*)
Save MIMLDataset as csv file

Parameters

path
[str] Path to store the csv file

`set_attribute`(*bag*, *index_instance: int*, *attribute*, *value: float*) → None
Update value from attributes

Parameters

bag
[int/str] Index or key of the bag of the dataset

index_instance
[int] Index of the instance

attribute: int/str
Attribute of the dataset

value: float
New value for the update

`set_features_name`(*features: list[str]*) → None
Set function for dataset features name

Parameters

features
[list[str]] List of the features name of the dataset

`set_labels_name`(*labels: list[str]*) → None
Set function for dataset labels name

Parameters

labels: list[str]
List of the labels name of the dataset

`set_name`(*name*) → None
Set function for dataset name

Parameters

name
[str] Name of the dataset

`show_dataset`(*start: int = 0*, *end: int | None = None*, *attributes=None*, *mode: str = 'table'*, *info=False*) → None
Function to show information about the dataset

Parameters

start
[int] Index of bag to start showing

end
[int] Index of bag to end showing

attributes: List of string
Attributes to show

mode
[str] Mode to show the dataset. Modes available are “table” and “csv” (csv format)

info: Boolean
Show more info

`split_dataset`(*train_percentage: float = 0.8*, *seed=0*)

Split dataset in two parts, one for training and the other for test

Parameters

train_percentage
[float] Percentage of bags in train dataset

seed: int
Seed to generate random numbers

Returns

dataset_train
[MIMLDataset] Train dataset

dataset_test
[MIMLDataset] Test dataset

`split_dataset_cv`(*folds: int = 4*, *seed=0*)

CrossValidation K-Fold split of dataset

Parameters

folds
[int] Number of datasets

seed: int
Seed to generate random numbers

Returns

dataset_train
[list[MIMLDataset]] Datasets

2.4 Dataset utils

```
miml.data.dataset_utils
```

CHAPTER

THREE

REPORT

3.1 Report

miml.report.report.Report	Class to generate a report

3.1.1 miml.report.report.Report

class miml.report.report.**Report**(*y_pred: ndarray*, *label_probs: ndarray*, *dataset_test:* MIMLDataset, *metrics: list[str] | None = None*, *header: bool = True*, *per_label: bool = False*)

Bases: `object`

Class to generate a report

calculate_metrics(*beta: int = 0.5*)

Calculate metrics of the predicted data

Parameters

beta
[int, default = 0.5] Beta value for the fbeta_score function

to_csv(*path: str | None = None*, *metrics: list[str] | None = None*)

Print/save data as csv format

Parameters

path
[str, default=None] Path to csv where the data would be stored

metrics
[list[str], default=None] List of metrics to show. If empty, show all metrics.

to_string(*metrics: list[str] | None = None*)

Print data as string format

Parameters

metrics
[list[str], default=None] List of metrics to show. If empty, show all metrics.

CHAPTER

FOUR

TRANSFORMATION

4.1 mimlTOmi

4.1.1 BinaryRelevanceTransformation

miml.transformation.mimlTOmi.binary_relevance_transformation.BinaryRelevanceTransformation	Class that performs a binary relevance transformation to convert a MIMLDataset class to numpy ndarray.

miml.transformation.mimlTOmi.binary_relevance_transformation.BinaryRelevanceTransformation

class miml.transformation.mimlTOmi.binary_relevance_transformation.**BinaryRelevanceTransformation**

Bases: object

Class that performs a binary relevance transformation to convert a MIMLDataset class to numpy ndarray.

transform_bag(*bag:* Bag) → list

Transform miml bag to multi instance bags

Parameters

bag :
Bag to be transformed to multi-instance bag

Returns

bags
[list[Bag]] List of n_labels transformed bags

transform_dataset(*dataset:* MIMLDataset) → list

Transform the dataset to multi-instance datasets dividing the original dataset into n datasets with a single label, where n is the number of labels.

Returns

datasets: list
Multi instance datasets

4.1.2 LabelPowersetTransformation

miml.transformation.mimlTOmi.label_powerset_transformation.LabelPowersetTransformation	Class that performs a label powerset transformation.

miml.transformation.mimlTOmi.label_powerset_transformation.LabelPowersetTransformation

class miml.transformation.mimlTOmi.label_powerset_transformation.**LabelPowersetTransformation**

Bases: object

Class that performs a label powerset transformation.

lp_to_ml_label(*label: int*) → ndarray
Transform lp label to multilabel

Parameters

label
Lp label to be transformed

Returns

labels
[np.ndarray] Multilabel labels

ml_to_lp_label(*labels: ndarray*) → float
Transform multilabel to lp label

Parameters

labels
[np.ndarray] Multilabel labels to be transformed

Returns

label
Lp label to be transformed

transform_bag(*bag:* Bag) → *Bag*

Transform miml bag to multi instance bags

Parameters

bag :
Bag to be transformed to multi-instance bag

Returns

transformed_bag
[Bag] Transformed bag

transform_dataset(*dataset:* MIMLDataset) → *MIMLDataset*

Transform the dataset to multi-instance dataset converting the labels to one label using binary to decimal codification

Returns

datasets: MIMLDataset
Multi instance dataset

4.2 mimlTOml

4.2.1 ArithmeticTransformation

miml.transformation.mimlTOml.arithmetic.ArithmeticTransformation	Class that performs an arithmetic transformation to convert a MIMLDataset class to numpy ndarray.

miml.transformation.mimlTOml.arithmetic.ArithmeticTransformation

class miml.transformation.mimlTOml.arithmetic.**ArithmeticTransformation**

Bases: *MIMLtoMLTransformation*

Class that performs an arithmetic transformation to convert a MIMLDataset class to numpy ndarray.

transform_bag(*bag:* Bag)

Transform a bag to a multilabel instance

Parameters

bag
[Bag] Bag to transform

Returns

transformed_bag
[Bag] Transformed bag

`transform_dataset`(*dataset:* MIMLDataset) → *MIMLDataset*

Transform the dataset to multilabel dataset converting each bag into a single instance being the value of each attribute the mean value of the instances in the bag.

Parameters

dataset
[MIMLDataset] Dataset to transform

Returns

transformed_dataset
[MIMLDataset] Transformed dataset

4.2.2 GeometricTransformation

miml.transformation.mimlTOml.geometric.GeometricTransformation	Class that performs a geometric transformation to convert a MIMLDataset class to numpy ndarray.

miml.transformation.mimlTOml.geometric.GeometricTransformation

class miml.transformation.mimlTOml.geometric.**GeometricTransformation**

Bases: *MIMLtoMLTransformation*

Class that performs a geometric transformation to convert a MIMLDataset class to numpy ndarray.

`transform_bag`(*bag:* Bag) → *Bag*

Transform a bag to a multilabel instance

Parameters

bag
[Bag] Bag to be transformed to multilabel instance

Returns

features
[ndarray of shape (n_features)] Numpy array with feature values

labels
[ndarray of shape (n_labels)] Numpy array with label values

transform_dataset(*dataset:* MIMLDataset) → *MIMLDataset*

Transform the dataset to multilabel dataset converting each bag into a single instance being the value of each attribute the geometric center of the instances in the bag.

Parameters

dataset
[MIMLDataset] Dataset to transform

Returns

transformed_dataset
[MIMLDataset] Transformed dataset

4.2.3 MinMaxTransformation

miml.transformation.mimlTOml.minmax.MinMaxTransformation	Class that performs a minmax transformation to convert a MIMLDataset class to numpy ndarray.

miml.transformation.mimlTOml.minmax.MinMaxTransformation

class miml.transformation.mimlTOml.minmax.**MinMaxTransformation**

Bases: *MIMLtoMLTransformation*

Class that performs a minmax transformation to convert a MIMLDataset class to numpy ndarray.

transform_bag(*bag:* Bag)

Transform a bag to a multilabel instance

Parameters

bag
[Bag] Bag to be transformed to multilabel instance

Returns

transformed_bag
[Bag] Transformed bag

`transform_dataset`(*dataset:* MIMLDataset)

Transform the dataset to multilabel dataset converting each bag into a single instance with the min and max value of each attribute as two new attributes.

Parameters

dataset
[MIMLDataset] Dataset to transform

Returns

transformed_dataset
[MIMLDataset] Transformed dataset

4.2.4 MIMLtoMLTransformation

miml.transformation.mimlTOml.miml_to_ml_transformation.MIMLtoMLTransformation	Abstract class to represent a MIMLtoML Transformation

miml.transformation.mimlTOml.miml_to_ml_transformation.MIMLtoMLTransformation

class miml.transformation.mimlTOml.miml_to_ml_transformation.**MIMLtoMLTransformation**

Bases: ABC

Abstract class to represent a MIMLtoML Transformation

abstract transform_bag(*bag:* Bag) → *Bag*

abstract transform_dataset(*dataset:* MIMLDataset) → *MIMLDataset*

INDEX

G

I

L

M

P

R

S

T