pyFUME

Release 0.1.13

C.E.M. Fuchs

Mar 21, 2022

MODEL CREATION:

CHAPTER

ONE

A PYTHON PACKAGE FOR FUZZY MODEL ESTIMATION

pyFUME is a Python package for automatic Fuzzy Models Estimation from data [1]. pyFUME contains functions to estimate the antecedent sets and the consequent parameters of a Takagi-Sugeno fuzzy model directly from data. This information is then used to create an executable fuzzy model using the Simpful library. pyFUME also provides facilities for the evaluation of the performance of the developed model.

1.1 Installation

You can install pyFUME with the command

```
pip install pyfume
```

1.2 Additional information

If you want to check out some example code or need more information on the usage of pyFUME, please visit our GitHub repository, or check our published article [1].

1.3 References

[1] Fuchs, C., Spolaor, S., Nobile, M. S., & Kaymak, U. (2020) "pyFUME: a Python package for fuzzy model estimation". In 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-8). IEEE.

1.3.1 pyFUME model builder

class pyfume.pyfume.**pyFUME**(*datapath=None*, *dataframe=None*, *nr_clus=2*, *process_categorical=False*, *method='Takagi-Sugeno'*, *variable_names=None*, *merge_threshold=1.0*, ***kwargs*)

Bases: object

Creates a new fuzzy model.

Parameters

- **datapath** – The path to the csv file containing the input data (argument 'datapath' or 'dataframe' should be specified by the user).
- **dataframe** – Pandas dataframe containing the input data (argument 'datapath' or 'dataframe' should be specified by the user).

- **nr_clus** – Number of clusters that should be identified in the data (default = 2).
- **process_categorical** – Boolean to indicate whether categorical variables should be processed (default = False).
- **method** – At this moment, only Takagi Sugeno models are supported (default = 'Takagi-Sugeno')
- **variable_names** – Names of the variables, if not specified the names will be read from the first row of the csv file (default = None).
- **merge_threshold** – Threshold for GRABS to drop fuzzy sets from the model. If the jaccard similarity between two sets is higher than this threshold, the fuzzy set will be dropped from the model.
- ****kwargs** – Additional arguments to change settings of the fuzzy model.

Returns An object containing the fuzzy model, information about its setting (such as its antecedent and consequent parameters) and the different splits of the data.

calculate_error(*method='MAE'*)

Calculates the performance of the model given the test data.

Args: method: The performance metric to be used to evaluate the model (default = 'MAE'). Choose from: Mean Absolute Error ('MAE'), Mean Squared Error ('MSE'), Root Mean Squared Error ('RMSE'), Mean Absolute Percentage Error ('MAPE').

Returns The performance as expressed by the chosen performance metric.

denormalize_values(*data*)

Takes normalized data points, and returns the denormalized (raw) values of that data point. This method only works when during modeling the data was normalized using the min-max method.

Parameters **xdata** – The input data (as numpy array with each row a different data instance and variables in the same order as in the original training data set) for which the normalized values should be calculated.

Returns Normalized values.

get_cluster_centers()

Returns the cluster centers as identified by pyFUME.

Returns cluster centers.

get_data(*data_set='test'*)

Returns the test or training data set.

Parameters **data_set** – Used to specify whether the function should return the training (data_set = "train"), test set (data_set = "test") or both training and test data (data_set = "all"). By default, the function returns the test set.

Returns Tuple (x_data, y_data) containing the test or training data set.

get_firing_strengths(*data*, *normalize=True*)

Calculates the (normalized) firing strength/ activition level of each rule for each data instance of the given data.

Parameters

- **xdata** – The input data (as numpy array with each row a different data instance and variables in the same order as in the original training data set) for which the labels should be calculated.

- **normalize** – Boolean that indicates whether the retuned fiing strengths should be normalized (normalize = True) or not (normalize = False), When the firing strenghts are nomalized the summed fiing strengths for each data instance equals one.

Returns Firing strength/activition level of each rule (columns) for each data instance (rows).

get_fold_indices()
Returns a list with the fold indices of each model that is created if crossvalidation is used when training.

Returns Perfomance of each cross validation model.

get_model()
Returns the fuzzy model created by pyFUME.

Returns The fuzzy model (as an executable object).

get_performance_per_fold()
Returns a list with the performances of each model that is created if crossvalidation is used when training..

Returns Perfomance of each cross validation model.

normalize_values(*data*)
Calculates the normalized values of a data point, using the same scaling that was used to training data of the model. This method only works when the data was normalized using the min-max method.

Parameters **xdata** – The input data (as numpy array with each row a different data instance and variables in the same order as in the original training data set) for which the normalized values should be calculated.

Returns Normalized values.

predict_label(*xdata*)
Calculates the predictions labels of a data set using the fuzzy model.

Parameters **xdata** – The input data (as numpy array with each row a different data instance and variables in the same order as in the original training data set) for which the labels should be calculated.

Returns Prediction labels.

predict_test_data()
Calculates the predictions labels of the test data using the fuzzy model.

Returns Prediction labels.

test_model(*xdata*, *ydata*, *error_metric='MAE'*)
Calculates the performance of the model using the given data.

Parameters

- **xdata** – The input data (as numpy array with each row a different data instance and variables in the same order as in the original training data set) for which the labels should be calculated.
- **ydata** – The target data (as single-column numpy array).
- **error_metric** – The error metric in which the performance should be expressed (default = 'MAE'). Choose from: Mean Absolute Error ('MAE'), Mean Squared Error ('MSE'), Root Mean Squared Error ('RMSE'), Mean Absolute Percentage Error ('MAPE').

Returns The performance as expressed in the chosen metric.

1.3.2 Takagi-Sugeno Model Builder

class pyfume.BuildTakagiSugeno.**BuildTSFIS**(*datapath=None*, *dataframe=None*, *nr_clus=None*, *variable_names=None*, *process_categorical=False*, *merge_threshold=1.0*, ***kwargs*)

Bases: object

Learns a a new Takagi-Sugeno fuzzy model.

Parameters

- **datapath** – The path to the csv file containing the input data (argument 'datapath' or 'dataframe' should be specified by the user).
- **dataframe** – Pandas dataframe containing the input data (argument 'datapath' or 'dataframe' should be specified by the user).
- **nr_clus** – Number of clusters that should be identified in the data (default = 2).
- **process_categorical** – Boolean to indicate whether categorical variables should be processed (default = False).
- **method** – At this moment, only Takagi Sugeno models are supported (default = 'Takagi-Sugeno')
- **variable_names** – Names of the variables, if not specified the names will be read from the first row of the csv file (default = None).
- **merge_threshold** – Threshold for GRABS to drop fuzzy sets from the model. If the jaccard similarity between two sets is higher than this threshold, the fuzzy set will be dropped from the model.
- ****kwargs** – Additional arguments to change settings of the fuzzy model.

Returns An object containing the fuzzy model, information about its setting (such as its antecedent and consequent parameters) and the different splits of the data.

1.3.3 Data Loader

class pyfume.LoadData.**DataLoader**(*datapath=None*, *dataframe=None*, *process_categorical=False*, *variable_names=None*, *normalize=False*, *delimiter=','*, *verbose=True*)

Bases: object

Creates an object that loads data from csv files and normalizes this data if requested.

Parameters

- **datapath** – The path to where the CSV file can be found. The data should be delimited using commas.
- **dataframe** – Data can be loaded by the user and specified as a dataframe. If a dataframe is specified, the datapath is automatically ignored.
- **variable_names** – If the CSV file does not contain the variable names, the user can specify them here. If this argument is not specified, the variable names are read from the first line of the CSV file (default = None).
- **normalize** – If switch on, the data will be normalized. The user can choose between 'minmax' or 'zscore' normalization (default = False).

- **delimiter** – Specify the symbol used to separate data in the dataset (default = ‘,’).
- **verbose** – Enable verbose mode (default = True).

1.3.4 Data Sampler

class pyfume.Sampler.**Sampler**(*train_x*, *train_y*, *number_of_bins=5*, *histogram=False*)
Bases: object

Creates a new Sampler object that makes it possible to oversample unbalanced data sets to make them more balanced.

Parameters

- **train_x** – The input data.
- **train_y** – The output data (true label/golden standard) on basis which will be sampled.
- **number_of_bins** – Number of clusters that should be identified in the data.
- **histogram** – True/False flag that determines whether a histogram of the frequencies of the output data will be plotted of both the old and new (= sampled) situation (default = False). The package ‘matplotlib.pyplot’ is required for this functionality.

oversample()
Created a more balanced data set by oversampling underrepresented data instances (based on values of the output variable) in the data set.

Returns

Tuple containing (new_train_x, new_train_y)

- new_train_x: The oversampled input data metrix.
- new_train_y: The oversampled output data matrix.

1.3.5 Data Splitter

class pyfume.Splitter.**DataSplitter**
Bases: object

Creates an object that can (provide the indices to) split the data in a training and test for model validation.

holdout(*dataX*, *dataY*, *percentage_training=0.75*)
Splits the data in a training and test set using the hold-out method.

Args: dataX: The input data. dataY: The output data (true label/golden standard). percentage_training: Number between 0 and 1 that indicates the

percentage of data that should be in the training data set (default = 0.75).

Returns: Tuple containing (x_train, y_train, x_test, y_test)

- x_train: Input variables of the training data.
- y_train: Output variables (true label/golden standard) of the training data.
- x_test: Input variables of the test data.
- y_test: Output variables (true label/golden standard) of the test data.

kfold(*data_length*, *number_of_folds=10*)

Provides the user with indices for ‘k’ number of folds for the training and testing of the model.

Parameters

- **data_length** – The total number of instances in the data sets (number of rows).
- **number_of_folds** – The number of folds the data should be split in (default = 10)

Returns A list with k (non-overlapping) sublists each containing the indices for one fold.

1.3.6 Clustering

class pyfume.Clustering.**Clusterer**(*nr_clus*, *x_train=None*, *y_train=None*, *data=None*, *relational_data=None*, *verbose=False*)

Bases: object

Creates a new clusterer object that can cluster the (training) data in the input-output feature space. The user should specify the 'data'argument OR the 'x_train' and 'y_train' argument.

Parameters

- **nr_clus** – Number of clusters that should be identified in the data.
- **x_train** – The input data (default = None).
- **y_train** – The output data (true label/golden standard) (default = None).
- **data** – The data to be clustered (default = None).

cluster(*method='fcm'*, ***kwargs*)

Clusters the data using the clustering method as specified by the user.

Parameters

- **method** – The method used for the clustering. The user can choose 'fcm' (fuzzy c-means), 'fst-pso' (fst-pso based clustering) and 'gk' (Gustafson-Kessel).
- ****kwargs** – Additional arguments to change settings of the clustering method.

Returns

Tuple containing (centers, partion_matrix, jm)

- centers: The location of the identified cluster centers.
- partition_matrix: A matrix containing the cluster memberships of each data point to each of the clusters.
- jm: Fitness function of the best solution.

1.3.7 Antecedent Set Estimator

class pyfume.EstimateAntecendentSet.**AntecedentEstimator**(*x_train*, *partition_matrix*)

Bases: object

Creates a new antecedent estimator object.

Parameters

- **x_train** – The input data.
- **partition_matrix** – The partition matrix of the input data (generated by a clustering the data).

determineMF(*mf_shape='gauss'*, *merge_threshold=1.0*)

Estimates the parameters of the membership functions that are used as antecedents of the fuzzy rules.

Parameters

- **mf_shape** – The desired shape of the fuzzy sets. The user can choose from 'gauss' (gaussian), 'gauss2' (double gaussion) or 'sigmf' (sigmoidal) (default = gauss).
- **merge_threshold** – Threshold for the merging of fuzzy sets for the GRABS approach. By default no merging takes place (default = 1.0).

Returns mu, sigma; if gauss2: mu1, sigma1, mu2, sigma2)

Return type A list with the estimated parameters of the membership functions (format if gauss

1.3.8 Consequent Parameter Estimator

class pyfume.EstimateConsequentParameters.**ConsequentEstimator**(*x_train*, *y_train*, *firing_strengths*)

Bases: object

Creates a new consequent estimator object.

Parameters

- **x_train** – The input data.
- **y_train** – The output data (true label/golden standard).
- **firing_strengths** – Matrix containing the degree to which each rule fires for each data instance.

suglms(*global_fit=False*, *df=0*)

Estimates the consequent parameters in the first-order Sugeno-Takagi model using least squares.

Parameters

- **global_fit** – Use the local (global_fit=False) or global (global_fit=True) least mean squares estimates. Global estimates functionality is still in beta mode, so use with caution.
- **df** – default value returned when the sum of grades equals to one (default = 0).

Returns The parameters for the consequent function.

zero_order()

Estimates the consequent parameters of the zero-order Sugeno-Takagi model using normalized means.

Parameters **df** – default value returned when the sum of grades equals to one (default = 0).

Returns The parameters for the consequent function.

1.3.9 Fire Strength Calculator

class pyfume.FireStrengthCalculator.**FireStrengthCalculator**(*antecedent_parameters*, *nr_clus*, *variable_names*, *fuzzy_sets_to_drop=None*, ***kwargs*)

Bases: object

Creates a new fire strength calculator object.

Parameters

- **antecedent_parameters** – The parameters of the antecedent sets of the fuzzy model as given as output

- **pyFUME's AntecedentEstimator clas (format** (*by*) – shape of the mf, parameters).
- **nr_clus** – Number of clusters in the data.
- **variable_names** – Names of the variables.
- **= Fuzzy sets identified by GRsABS to be dropped from the model** (*fuzzy_sets_to_drop*) –
- ****kwargs** – Additional arguments to change settings of the fuzzy model.

calculate_fire_strength(*data*)

Calculates the firing strength per rule of the fuzzy model given a data set.

Parameters **data** – The data of which the firing strengths per rule should be calculated.

Returns data point index, column: rule/cluster number).

Return type The firing strengths per rule per data point (rows

1.3.10 Feature Selector

class pyfume.FeatureSelection.**FeatureSelector**(*dataX*, *dataY*, *nr_clus*, *variable_names*, *model_order='first'*, *performance_metric='MAE'*, *verbose=True*, ***kwargs*)

Bases: object

Creates a new feature selection object.

Parameters

- **dataX** – The input data.
- **dataY** – The output data (true label/golden standard).
- **nr_clus** – Number of clusters that should be identified in the data.
- **variable_names** – Names of the variables
- ****kwargs** – Additional arguments to change settings of the fuzzy model.

fst_pso_feature_selection(*max_iter=100*, *min_clusters=2*, *max_clusters=10*, *performance_metric='MAE'*, ***kwargs*)

Perform feature selection using the FST-PSO [1] variant of the Integer and Categorical PSO (ICPSO) proposed by Strasser and colleagues [2]. ICPSO hybridizes PSO and Estimation of Distribution Algorithm (EDA), which makes it possible to convert a discrete problem to the (real-valued) problem of estimating the distribution vector of a probabilistic model. Each fitness evaluation a random solution is generated according to the probability distribution encoded by the particle. Because the implementation is a variant on FST-PSO, the optimal settings for the PSO are set automatically.

If the number of clusters is set to None, this method simultaneously choses the optimal number of clusters.

[1] Nobile, M. S., Cazzaniga, P., Besozzi, D., Colombo, R., Mauri, G., & Pasi, G. (2018). Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. Swarm and evolutionary computation, 39, 70-85.

[2] Strasser, S., Goodman, R., Sheppard, J., & Butcher, S. (2016). A new discrete particle swarm optimization algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference 2016 (pp. 53-60).

Parameters

- **max_iter** – The maximum number of iterations used in the PSO (default = 10).
- **min_clusters** – The minimum number of clusters to be identified in the data set (only
- **nr_clusters = None)** (*when*) –
- **max_clusters** – The maximum number of clusters to be identified in the data set (only
- **nr_clusters = None)** –
- **performance_metric** – The performance metric on which each solution is evaluated (default
- **Absolute Error** (*Mean*) –
- ****kwargs** – Additional arguments to change settings of the fuzzy model.

Returns

Tuple containing (selected_features, selected_feature_names, optimal_number_clusters)

- selected_features: The indices of the selected features.
- selected_feature_names: The names of the selected features.
- optimal_number_clusters: If initially nr_clusters = None, this argument encodes the optimal number of clusters in the data set. If nr_clusters is not None, the optimal_number_clusters is set to nr_clusters.

log_wrapper(***kwargs*)

Performs feature selection using the wrapper method while also checking whether .

Parameters ****kwargs** – Additional arguments to change settings of the fuzzy model.

Returns

Tuple containing (selected_features, selected_feature_names)

- selected_features: The indices of the selected features.
- selected_feature_names: The names of the selected features.

wrapper(***kwargs*)

Performs feature selection using the wrapper method.

Parameters ****kwargs** – Additional arguments to change settings of the fuzzy model.

Returns

Tuple containing (selected_features, selected_feature_names)

- selected_features: The indices of the selected features.
- selected_feature_names: The names of the selected features.

1.3.11 Model tester

class pyfume.Tester.**SugenoFISTester**(*model*, *test_data*, *variable_names*, *golden_standard=None*, *list_of_outputs=['OUTPUT']*)

Bases: object

Creates a new Tester object to be able to calculate performance metrics of the fuzzy model.

Parameters

- **model** – The model for which the performance metrics should be calculated
- **test_data** – The data to be used to compute the performance metrics
- **variable_names** – A list of the variables names of the test data (which should correspond with the variable names used in the model).
- **golden_standard** – The 'True' labels of the test data. If not provided, the only predictions labels can be generated, but the error will not be calculated (default = None).
- **list_of_outputs** – List of the output names (which should correspond with the output names used in the model) (default: OUTPUT).

calculate_MAE()

Calculates the Mean Absolute Error of the model given the test data.

Returns The Mean Absolute Error of the fuzzy model.

calculate_MAPE()

Calculates the Mean Absolute Percentage Error of the model given the test data.

Returns The Mean Absolute Percentage Error of the fuzzy model.

calculate_MSE()

Calculates the Mean Squared Error of the model given the test data.

Returns The Mean Squared Error of the fuzzy model.

calculate_RMSE()

Calculates the Root Mean Squared Error of the model given the test data.

Returns The Root Mean Squared Error of the fuzzy model.

calculate_performance(*metric='MAE'*)

Calculates the performance of the model given the test data.

Args: metric: The performance metric to be used to evaluate the model. Choose from: Mean Absolute Error ('MAE'), Mean Squared Error ('MSE'), Root Mean Squared Error ('RMSE'), Mean Absolute Percentage Error ('MAPE').

Returns The performance as expressed by the chosen performance metric.

predict()

Calculates the predictions labels of the test data using the fuzzy model.

Returns

Tuple containing (result, error)

- result: Prediction labels.
- error: The difference between the prediction label and the 'true' label.

PYTHON MODULE INDEX

p

INDEX

N

O

P

S

T

W

Z