# Namespace Smart-house-REST-API.AutoML

## Sub-modules

* Smart-house-REST-API.AutoML.AutoML Module Smart-house-REST-API.AutoML.AutoML =========================================

## Classes

AutoTuningHyperparameters(model, data, params, cost, time\_series\_size=3, forecast=3, standarization\_rule='sensor', y\_rule='device\_blinds')

Initialization of AutoTuningHyperparameters object.

Args: model (object): Model that will be optimized. EX. xgboost classifier. data (object): Whole dataset in the form of pandas dataframe. It will be splitted into train and test dataset. params (dict): Search space for the ml algorithm. cost (function): Cost function for the model, higher value means metter model (maximization). time\_series\_size (int, optional): Size of the time series. Defaults to 3. forecast (int, optional): How many records into the future should model look. Defaults to 3. standarization\_rule (str, optional): Column name (partial) in the dataset that will be standatized. Defaults to ‘sensor’. y\_rule (str, optional): Column name that wil be predicted. Defaults to ‘device\_blinds’.

### Static methods

generate\_all\_possible\_params(params)

Static method generationg all possible combinations of parameters.

Args: params (dict): dictionary representing each parameter in the form: {‘parameter name’: [min value, max value, step]}.

Returns: dict: all possible params.

### Methods

auto\_tune\_pipeline(self, pipeline=['random', 'grid'], narrow\_to=0.2, params=None)

Pipleline of algorithms given by the ‘pipeline’ parameter. After each search, search space is narrowed by the parameter ‘narrow\_to’ around current best model.

Args: pipeline (list, optional): List of searches that will be performed. Defaults to [‘random’, ‘grid’]. narrow\_to (float, optional): How much the search space should be narrowed by after each iteration. Defaults to 0.2. params ([type], optional): Search space. Defaults to None.

Returns: object: best model found after all searches

fit\_model(self, params)

Methods that fits provided ml model on given hyperparameters from the search space, and returns it withm its score.

Args: params (dicr): hyperparameters of the model.

Returns: tuple: object and its score on the test dataset.

grid\_search(self, params=None, verbose=1)

Grid search algorithm on the provided search space.

Args: params (dict, optional): Search space for the algorithm. Defaults to None. verbose (int, optional): Parameter defining if anything should be printed to the console. verbose>=1 -> yes, verobose < 1 - no. Defaults to 1.

Returns: tuple: best found parameters and score of the model trained on them

random\_search(self, verbose=1, group\_size=5, iterations=20, params=None)

Random search algorithm on the provided search space.

Args: verbose (int, optional): Parameter defining if anything should be printed to the console. verbose>=1 -> yes, verobose < 1 - no. Defaults to 1. group\_size (int, optional): Number of random samples taken by the random searcj to compare. Defaults to 5. iterations (int, optional): Number of algorithm iterations. Defaults to 20. params ([type], optional): Search space for the algorithm. Defaults to None.

Returns: tuple: best found parameters and score of the model trained on them

DataPreprocessor(data, standarization\_rule='sensor', tolerance=Timedelta('0 days 00:00:05'), nan\_value=-1, numerical\_date=True)

init function of class DataProcessor

Args: data (dict): raw data taken from http request standarization\_rule (str, optional): columns which will be standarized. Defaults to ‘sensor’. tolerance (pd.Timedelta, optional): optional parameter for merging data from different devices, based on simmilarity in date of sending. Defaults to pd.Timedelta(‘5s’). nan\_value (int, optional): value which will replace NaN falues. Defaults to -1. numerical\_date (bool, optional): if True, numerical represetation of date will be added to each row. Defaults to True.

### Static methods

standarize(X, standarization\_rule='sensor')

static method used for data standarization

Args: X (pandas.DataFare): whole data in the form of DataFrame. standarization\_rule (str, optional): Columns which will be standarized. Defaults to ‘sensor’.

Returns: pandas.DataFrame: standarized DataFrame.

### Methods

time\_series(self, time\_series\_size=3, forecast=1, y\_rule='device')

Method of class DataProcessor that returns padded time series

Args: time\_series\_size (int, optional): parameter used for determining how many rows represent one data point. Defaults to 3. forecast (int, optional): value representing how many rows in the future should Y be. Defaults to 3. y\_rule (str, optional): name of columns which will be our target. Defaults to ‘device’.

Returns: tuple: tuple of two dataframes, first one representing X (model input), second one Y (target).