generic classifier

Technical Document

*Meghnad*

Generic Classifier Document Version

|  |  |  |
| --- | --- | --- |
| ***Author*** | ***Version*** | ***Date*** |
| *Kirankumar A M* | *1* | *22.06.2022* |

**Contents**

[1. Introduction 4](#_Toc107399165)

[2. Pre-requisites 4](#_Toc107399166)

[3. Config 5](#_Toc107399167)

[4. Data connectors 5](#_Toc107399168)

[5. Basic Structure of Files 6](#_Toc107399169)

[6. Code Module 7](#_Toc107399170)

[*GenericClfConfig* 7](#_Toc107399171)

[ *get\_model\_by\_modes(self, mode: str)* 7](#_Toc107399172)

[ *get\_other\_gridsearchcv\_params(self)* 7](#_Toc107399173)

[ *get\_other\_general\_params(self)* 7](#_Toc107399174)

[*GenericClfTrnPrep* 7](#_Toc107399175)

[ *prep(self)* 7](#_Toc107399176)

[ *read\_data(self)* 7](#_Toc107399177)

[ *label\_encode\_target\_variable(self)* 7](#_Toc107399178)

[ *drop\_rows\_and\_columns(self)* 7](#_Toc107399179)

[ *numerical\_categorical(self)* 7](#_Toc107399180)

[ *cleaning\_target\_variable(self)* 8](#_Toc107399181)

[ *label\_dummy\_encoding(self)* 8](#_Toc107399182)

[ *imbalance\_correction(self)* 8](#_Toc107399183)

[ *standardizing(self)* 8](#_Toc107399184)

[*GenericClfTrn* 8](#_Toc107399185)

[ *config\_connectors(self, data\_path: str, data\_type: str, target\_cols: [str], feature\_cols:[str], data\_org: str, multi\_targets: bool, multi\_labels: bool, seperator: str, multi\_labels\_seperator: str = None, dir\_to\_save\_model: str = None, model\_mode: str = ‘slow’)* 8](#_Toc107399186)

[ *trn(self, perf\_metric: str, alg : dict = None)* 8](#_Toc107399187)

[*GenericClfPredPrep* 8](#_Toc107399188)

[ *pred\_prep(self)* 8](#_Toc107399189)

[ *get\_data(self)* 9](#_Toc107399190)

[ *one\_hot\_encoding(self)* 9](#_Toc107399191)

[ *standardize(self)* 9](#_Toc107399192)

[*GenericClfPred* 9](#_Toc107399193)

[ *best\_model\_prediction(self)* 9](#_Toc107399194)

[ *other\_model\_prediction(self)* 9](#_Toc107399195)

[7. Conclusion 9](#_Toc107399196)

# Introduction

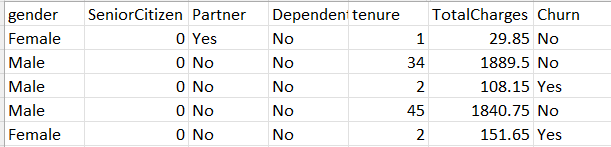
Generic classifier, a sub-module of Meghnad, is a user-friendly classification algorithm developed to learn from your present data and then predict the outcomes of your future data. It automatically pre-processes the available data to make it system ready which can be further used to train and test various AI/ML models. This module supports data in different formats and the user can input the models on which the data needs to be trained or the generic classifier module itself has a built-in model configuration which automatically trains the data for different classification models on pre-defined hyperparameters. The performance of various models is assessed using the user specified metric and all the trained models are saved in the system. The best performing model on the train data and other trained models can be further used to predict the target in test data.

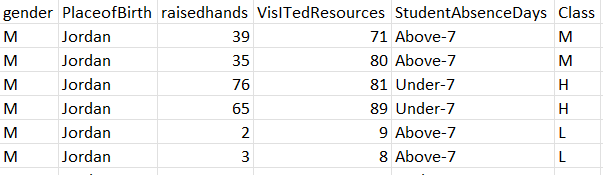
# Pre-requisites

The data which is to be trained and predicted in this module should follow some necessary formats. This module at present can only read csv, txt and excel files. The features should either be a qualitative (categorical) or a quantitative (numerical). The target variable can be of the below types:

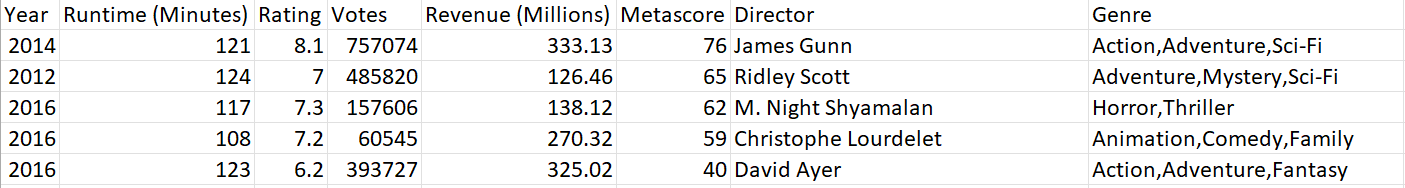
* Binary class (Yes/No, True/False, 0/1, etc,)
* Multi-class (H/M/L, 0/1/2, etc,)
* Multi-label ([Horror, Comedy], [North, North-west], etc,)

Sample datasets are as shown below:



**Binary class dataset (Target – Churn)**

**Multi-class dataset (Target – Class)**

****

**Multi-label dataset (Target – Genre)**

# Config

The default models that are configured in this module are:

* Logistic Regression
* KNeighbors Classifier
* Random Forest
* Adaboost Classifier
* Support Vector Machines (SVM)
* Gradient Boosting Classifier
* XGBoost Classifier
* LGBM Classifier
* CatBoost Classifier
* Multi-Layer Perceptron classifier

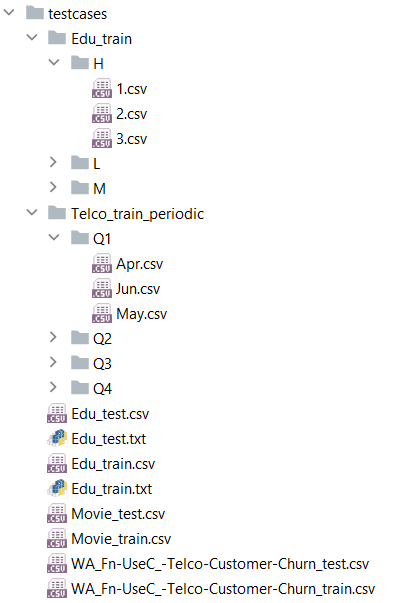
# Data connectors

* **Data path:** Path of train dataset.
* **Data type:** Type of the data file. Currently, only csv, txt and excel is supported.
* **Target Cols:** Name of the target column.
* **Feature columns:** List of columns which are to be used as features for training model.
* **Data Org:** Organization of the data.
* **Multi Targets:** If there are multiple target variables, then True else False. Currently not supported.
* **Multi Labels:** If the target variables have multiple labels, then True else False *.*
* **Seperator**: Seperator of columns in data.
* **Multi Labels Seperator:** Seperator of labels in target*.*
* **Directory to save model**: Directory where the trained model needs to be saved
* **Model\_mode:** Training mode ('slow', 'medium', 'fast'), Decides the number of default models to be selected while training.

# Basic Structure of Files

The structural format of the files and folders present in the generic classifier module are as below:

Currently training data can be accessed from a single directory, multi-directory with target variable as folder name and multi-directory with periodic data. Sample data organisation is shown below:



Multi-directory with target variable as folder name

Multi-directory with periodic data

Single directory data

# Code Module

## *GenericClfConfig*

This class belongs to ‘\gen\_clf\cfg\config.py’. The details of the functions under this class are as follows,

### ***get\_model\_by\_modes(self, mode: str)***

Returns a dictionary of different classification models and their hyper parameters based on the mode (slow / medium / fast) given by the user.

### ***get\_other\_gridsearchcv\_params(self)***

Returns a dictionary of default parameters of grid search cv.

### ***get\_other\_general\_params(self)***

Returns a dictionary of other general parameters required for training.

## *GenericClfTrnPrep*

This class belongs to ‘\gen\_clf\src\trn\_prep.py’. The details of the functions under this class are as follows,

### ***prep(self)***

This function returns the processed training data along with the features, target, categorical variables, final features, and the encoders of target variable.

### ***read\_data(self)***

This function reads the train data from a single directory or multi-directory and gets the features and target separately.

### ***label\_encode\_target\_variable(self)***

This function encodes the target variable to a numeric format

### ***drop\_rows\_and\_columns(self)***

This function replaces the empty strings(‘’) and blank spaces(‘ ‘) in data to null values and converts those empty string replaced columns from object data type to float data type if possible. It first drops columns with more than half of the rows having missing values and then drops rows with missing values. Also, nominal variables like ID, S.no. , name are dropped.

### ***numerical\_categorical(self)***

This function separates the numerical and categorical variables based on data type first. Then if a numerical variable has less than 5 unique values then the same is also appended to the categorical variable list. All these categorical variables are checked for dependency with target variable using the chi-square test and the independent variables are dropped from the train data. The dependency of numerical variable on target is checked using the Kruskal-Wallis-H-Test and the independent numerical variables are dropped from the train data.

### ***cleaning\_target\_variable(self)***

This function transforms the multi-label target variable to binary matrix using the MultiLabelBinarizer function from sklearn.preprocessing.

### ***label\_dummy\_encoding(self)***

This function encodes the categorical variables by creating dummy variables. Finally, the processed features are got in a list.

### ***imbalance\_correction(self)***

In this function, if the target variable is not multi-label and the target variable has an un-equal distribution of classes, then the train data is treated for imbalance using SMOTE resampling technique

### ***standardizing(self)***

This function standardizes the features of training data so that all the columns are in the same scale with mean 0 and variance 1.

## *GenericClfTrn*

This class belongs to ‘\gen\_clf\src\trn.py’. The details of the functions under this class are as follows,

### ***config\_connectors(self, data\_path: str, data\_type: str, target\_cols: [str], feature\_cols:[str], data\_org: str, multi\_targets: bool, multi\_labels: bool, seperator: str, multi\_labels\_seperator: str = None, dir\_to\_save\_model: str = None, model\_mode: str = ‘slow’)***

This function sets all the data connector parameters passed from user.

### ***trn(self, perf\_metric: str, alg : dict = None)***

This function tunes the hyper-parameters depending upon the size of data and starts training of the models. If the user has not specified any models (alg = None), then data will be trained on the default models and if the user had specified a set of models (alg = {}), then data will be trained on those models. Models are assessed on the metric which the user has specified and returns the directory where the model is saved along with the feature importance of the trained model.

## *GenericClfPredPrep*

This class belongs to ‘\gen\_clf\src\pred\_prep.py’. The details of the functions under this class are as follows,

### ***pred\_prep(self)***

This function returns the processed prediction data along with the actual value of target variable and the index of the rows that have been deleted.

### ***get\_data(self)***

This function drops the columns in prediction data that had been dropped in train data. It also does the initial processing by cleaning the cells from empty spaces and removes rows that has missing values.

### ***one\_hot\_encoding(self)***

This function transforms the target variable using the transformation that had been done on train data and retains the transformed actual value. It also creates dummy variables for categorical variables in the prediction data and finally checks if the columns in the processed prediction data is same as the columns that were present in the processed training data just before training.

### ***standardize(self)***

This function standardizes the features of prediction data so that all the columns are in the same scale with mean 0 and variance 1.

## *GenericClfPred*

This class belongs to ‘\gen\_clf\src\pred.py’. The details of the functions under this class are as follows,

### ***best\_model\_prediction(self)***

This function predicts the value of prediction data using the best model obtained from training and stores the results in a dictionary.

### ***other\_model\_prediction(self)***

This function predicts the value of prediction data using the models apart from the best model obtained from training and stores the results in a dictionary.

# Conclusion

Thus, generic classifier module can be used to predict the value of a categorical target variable present in many types. This classifier pipeline enables user to spend the least effort and get the most effective output.