RapidML Documentation

Release 1.0.0

Ritabrata Maiti

Jul 24, 2018

Contents:

Getting started is as easy as `pip install RapidML`

GitHub Repository: https://github.com/ritabratamaiti/RapidML

What is **RapidML**?

Well, RapidML is your Smart Machine Learning assistant that not only automates the creation of machine learning models but also enables you to easily deploy the models to the cloud.

RapidML is perfect for Python programmers at all levels, ranging from beginners who want to get into Data Science and Machine Learning to intermediate and advanced programmers who want to bring Machine Learning to consumer and industry usage applications.

Apart from making predictions in Python, RapidML models can be exported as Web APIs to develop Machine Learning applications in a wide variety of platforms, such as Javascript, Android, iOS. . . . **and almost everything else which can make and receive web requests**!

RapidML

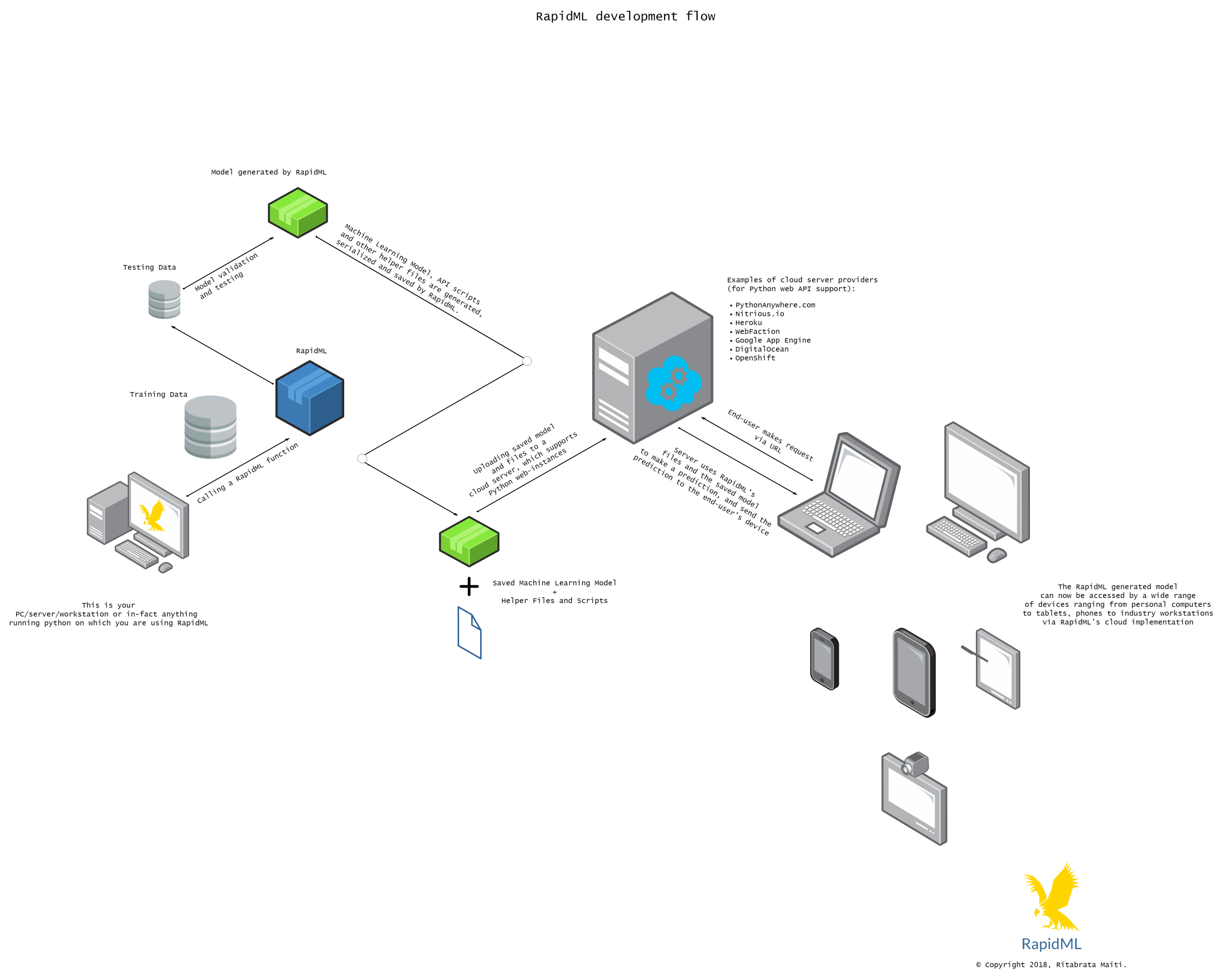

RapidML development flow
Model generated by RapidML
Testing Data
Model validation and testing
Machine Learning Model, API scripts and other helper files are generated, serialized and saved by RapidML.
RapidML
Training Data
Calling a RapidML function
This is your PC/server/workstation or in-fact anything running python on which you are using RapidML
Uploading saved model and files to a cloud server, which supports Python web-instances
Saved Machine Learning Model + Helper Files and Scripts
Examples of cloud server providers (for Python web API support):
• PythonAnywhere.com
• Nitrious.io
• Heroku
• WebFaction
• Google App Engine
• DigitalOcean
• OpenShift
End-user makes request via URL
Server uses RapidML's files and the saved model to make a prediction, and send the prediction to the end-user's device
The RapidML generated model can now be accessed by a wide range of devices ranging from personal computers to tablets, phones to industry workstations via RapidML's cloud implementation
RapidML
© Copyright 2018, Ritabrata Maiti.

Fig. 1: (Click to enlarge)

CHAPTER 1

Installation

Recommended, using pip:

```
pip install RapidML
```

Do note that `RapidML` is built on the following libraries:

- NumPy
- scikit-learn
- pandas
- TPOT
- Flask

Recommended: Most of the necessary Python packages can be installed via the Anaconda Python distribution (https://www.continuum.io/downloads). Python 3 is recommended over Python 2

NumPy, SciPy, scikit-learn, and pandas can be installed in Anaconda via the command:

```
conda install numpy scipy scikit-learn pandas
```

Installing TPOT:

```
pip install tpot
```

Installing Flask:

```
pip install flask
```

CHAPTER 2

Using RapidML

2.1 Why use RapidML?

2.1.1 Extremely easy to use and intuitive API

RapidML can accept DataFrame inputs, which are essentially the easiest ways to programmatically represent .csv (Comma separated values) or .xlsx (Excel) files in Python. Since .csv files and .xlsx are the most commonly used file types in organizing and storing large amounts of data, popular python libraries like Pandas provide methods for converting these files into DataFrames in Python, leading to the popularity of the DataFrame datatype in Data Science and Machine Learning.

Not all data is numeric, and it is common for some categorical data to be in a textual format. However, many machine learning algorithms can only work with numeric data. RapidML easily solves this problem for data scientists by performing automatic label encoding on such data. An example of this could be a gender attribute (field) containing 'female', 'male' and 'other' as possible values, then RapidML can encode these as, say, {'female': 0, 'other': 1, 'male': 2}

2.1.2 Rapid development of Machine Learning Models and Web APIs

RapidML utilizes TPOT (Tree-based Pipeline Optimization Tool) as its backend for automated algorithm selection. TPOT utilizes genetic programming to select the best Scikit-Learn model and performs hyperparameter optimization on the model. RapidML then trains this model and uses it to perform future predictions.

Using the model, RapidML programmatically generates a Web API using the Flask framework, which can be easily self-hosted or uploaded to a remote server in order to make machine learning predictions in the cloud. Requests are quite easy to make, by using URL parameters in the `/query?ip=` part of the request. See making requests for more info.

2.1.3 Machine Learning models can be quickly deployed to production

This is an extension of the previous point and serves to highlight the fact that RapidML models can be used for easy and quick prototyping of production-level applications, on multiple platforms such as through the web, as well as Android and iOS applications, just to name a few.

2.1.4 RapidML is very versatile and it can be extended for use by more experienced Developers and Data Scientists

If you prefer to select machine learning algorithms on your own, then RapidML allows you to train the model and run it in the cloud. For example, if you don't wish to use TPOT's automated machine learning, but would rather implement a neural network (Scikit-learn's implementation), then you can easily do so using RapidML's `udm` (user-defined model) function in order to run the neural network in the cloud. RapidML's versatility makes it a tool that encompasses the differences between programmers, Data Scientists, and Web Developers at all expertise levels and enables everyone to take part in the exciting field of Machine Learning and Artificial Intelligence.

2.1.5 RapidML in Action

RapidML has a wide variety of application, ranging from the field of medicine to scientific predictions in statistics, like a prediction of radiation emissions or meteorological prediction (weather prediction) and even predicting stocks prices. RapidML is already being used in the development of a Machine Learning Web API that is utilized in an android application for ASD detection in adult patients.

Find this project here: https://github.com/ritabratamaiti/Autism-Detection-API. *This project was developed to showcase RapidML's use cases, and shouldn't be use for making diagnosis without clinical trials, and permission of the author.*

2.2 RapidML with code

The following code should give a fair idea on RapidML usage. Note: Visit *Examples* for more code samples and projects.

```
import RapidML
import os
import pandas as pd


# This Autism Screening Adult Data Set is from UCI Machine Learning Repository and is␣
↪available here: https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult

df = pd.read_csv('out.csv')
df = df.drop(columns = ['Unnamed: 0'])
df.head()

ml_model = RapidML.rapid_classifier(df,name='ASDapi')
```

Note: The training data is an Autism Screening Adult DataSet from UCI Machine Learning Repository and is available here: https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult

The code generates the following output. Here `ml_model` is assigned an `rml` object by the `RapidML.rapid_classifier` function. To learn about the `rml` class, as well as RapidML functions, go to *RapidML API*.

```
RapidML, Version: 0.1, Author: Ritabrata Maiti


       .---.        .-----------
      /     \  __  /    ------
     / /     \(  )/    -----
    //////   ' \/ `   ---
   //// / // :    : ---
  // /   /  /`    '--
 //          //..\
        ====UU====UU====
            '//||\\`
              ''``

Warning: xgboost.XGBClassifier is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.

Using the RapidML Classifier; Experimental, For Issues Contact Author:␣
↪ritabratamaiti@hiretrex.com
Label Encoding is being done....

Training....

Generation 1 - Current best internal CV score: 1.0
Generation 2 - Current best internal CV score: 1.0
Generation 3 - Current best internal CV score: 1.0
Generation 4 - Current best internal CV score: 1.0
Generation 5 - Current best internal CV score: 1.0

Best pipeline: DecisionTreeClassifier(input_matrix, criterion=entropy, max_depth=2,␣
↪min_samples_leaf=4, min_samples_split=6)

Sample Output from input dataframe:
1,1,0,1,0,0,1,1,0,1,6,35.0,f,White-European,no,yes,United States,no,Self,NO
```

CHAPTER 3

RapidML API

Getting started with RapidML is easy. All RapidML functions return an object of the `rml` class.

3.1 The `RapidML.rml` Class

3.1.1 `rml` class attributes

3.1.1.1 `model`:

This is the machine learning model generated by RapidML. It has already been trained on the training data and the target that was provided by the user, either as a DataFrame or in the form of X,y arrays wherein X is training data and y is target variables. This attribute is never null.

3.1.1.2 `m_tpot`:

Note: This may be null depending on the type of the functions use. See function usage here. This is a TPOT object which may be a TPOTClassifier or a TPOTRegressor. RapidML uses this object to find the optimal machine learning model for the supplied data.

You can use the various functions and attributes of rml.m_tpot in order to evaluate the trained model. For example: `rml.m_tpot.score(testing_features, testing_classes)` will allow us to evaluate our model on training data by returning an accuracy score. See the TPOT documentation for all the available functions and attributes of `rml.m_tpot`

3.1.1.3 `d`:

Note: This may be null depending on the type of the functions use. See function usage here.

This is a defaultdict containing the labels and their corresponding transformed values, should we choose to labelencode the table. See sklearn.preprocessing.LabelEncoder for more details.

3.1.2 `rml` class functions

3.1.2.1 `put(self, mdl, d=None):`

This is a method used by the RapidML functions for assignment of attributes of rml objects. Here `mdl` can either be the model supplied by the user or supplied by RapidML via TPOT.

If `mdl` is a TPOT object then the `model` attribute is `mdl.fitted_pipeline_` (the best pipeline found with TPOT for the training data) and the `m_tpot` attribute is a TPOT object. However if `mdl` is a fitted (trained) machine learning model then the `model` attribute will be mdl and the `m_tpot` attribute will be null.

If we decide to labelencode the training data, then the `d` attribute will be the *d* supplied as the function argument. Otherwise, the `d` attribute will be null.

3.1.2.2 `le(self, df):`

This function may be called by the user from an `rml` object, in order to perform label encoding on another dataset, using the same encoding table used on a previous similar dataset.

For example, if we wish to perform the same transformation of labels on two DataFrames with same types of columns but different rows, then we first labelencode the first table, and then use this function to labelencode the next table.

3.2 `RapidML.rapid_classifier`

The `rapid_classifier` performs label encoding on the input DataFrame `df` (which are the features), depending on the user's input. It then uses a TPOT backend to perform an intelligent search to find and optimize the best classifier in accordance with the input data. Finally, it populates an `rml` object's attributes and returns this object.

3.2.1 Parameters

3.2.1.1 `df`

Type: `pandas.DataFrame`

This is the input DataFrame provided by the users as the training features as well as the initial columns and the target as the last column on the DataFrame.

3.2.1.2 `le`

Type: `str`

The default value is `'Yes'`. If `le` is `'Yes'`, then RapidML will labelencode the input DataFrame supplied as `df`, and store the `LabelEncoder` in a `defaultdict`. Or, if `le` is `'No'` then LabelEncoding will not be done. For any other value of `le`, a value error will be raised.

3.2.1.3 `model`

Type: `tpot.TPOTClassifier`

The default value is `tpot.TPOTClassifier(generations=5, population_size=50, verbosity=2)`. This is a TPOTClassifier object. You can pass a TPOTClassifier object with different parameter configurations as per your requirement. In general, increasing the `generations` and `population_size` increases the model's accuracy. See TPOTClassifier for more details.

3.2.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `dill` files, as well as the `API.py and ``helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the web API.

3.2.2 Returns

Returns a `rml` object. If `le` is `'Yes'` then `rml.d` is populated, otherwise, it is null. `rml.model` and `rml.m_tpot` are always populated, when using `rapid_classifier`.

3.2.3 Files Created

3.2.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `dill`.

3.2.3.2 `d`

This is the `DefaultDict` (like `dict`) containing the LabelEncoder used to encode the labels in the DataFrame. It has been saved after serialization via `dill`.

3.2.3.3 `df`

This is the skeletal DataFrame, which contains only headers and no data. It has been saved after serialization via `dill`.

3.2.3.4 `dt`

This is a list containing the data types of the columns in the input `DataFrame`. It has been saved after serialization via `dill`.

3.2.3.5 `f`

This is a string containing a dummy input value and can be fed to the API as an URL argument. It is the second row of the input `DataFrame`, converted to a string. It has been saved after serialization via `dill`.

3.2.3.6 `API.py`

This is the actual Flask-API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

3.2.3.7 `helper.py`

This is a helper module used by `API.py` and performs the actual predictions using the RapidML generated model.

3.3 `RapidML.rapid_regressor`

The `rapid_regressor` performs label encoding on the input DataFrame `df` (which are the features), depending on the user's input. It then uses a TPOT backend to perform an intelligent search to find and optimize the best regressor in accordance with the input data. Finally, it populates an `rml` object's attributes and returns this object.

3.3.1 Parameters

3.3.1.1 `df`

Type: `pandas.DataFrame`

This is the input DataFrame provided by the users as the training features as well as the initial columns and the target as the last column on the DataFrame.

3.3.1.2 `le`

Type: `str`

The default value is `'No'`. If `le` is `'Yes'`, then RapidML will labelencode the input DataFrame supplied as `df`, and store the `LabelEncoder` in a `defaultdict`. Or, if `le` is `'No'` then LabelEncoding will not be done. For any other value of `le`, a value error will be raised.

3.3.1.3 `model`

Type: `tpot.TPOTRegressor`

The default value is `tpot.TPOTRegressor(generations=5, population_size=50, verbosity=2)`. This is a TPOTRegressor object. You can pass a TPOTRegressor object with different parameter configurations as per your requirement. In general, increasing the `generations` and `population_size` increases the model's accuracy. See TPOTRegressor for more details.

3.3.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `dill` files, as well as the `API.py and ``helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the web API.

3.3.2 Returns

Returns a `rml` object. If `le` is `'Yes'` then `rml.d` is populated, otherwise, it is null. `rml.model` and `rml.m_tpot` are always populated, when using `rapid_regressor`.

3.3.3 Files Created

3.3.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `Dill`.

3.3.3.2 `d`

This is the `DefaultDict` (like `dict`) containing the LabelEncoder used to encode the labels in the DataFrame. It has been saved after serialization via `Dill`.

3.3.3.3 `df`

This is the skeletal DataFrame, which contains only headers and no data. It has been saved after serialization via `Dill`.

3.3.3.4 `dt`

This is a list containing the data types of the columns in the input `DataFrame`. It has been saved after serialization via `Dill`.

3.3.3.5 `f`

This is a string containing a dummy input value and can be fed to the API as an URL argument. It is the second row of the input `DataFrame`, converted to a string. It has been saved after serialization via `Dill`.

3.3.3.6 `API.py`

This is the actual Flask-API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

3.3.3.7 `helper.py`

This is a helper module used by `API.py` and performs the actual predictions using the RapidML generated model.

3.4 `RapidML.rapid_classifier_arr`

The `rapid_classifier_arr` function is similar to the `rapid_classifier`, except rather receiving the features and targets as a single input DataFrame, the function receives the features as X (type numpy.array), and the target as Y (type `numpy.array`). Another important point of difference is that this function doesn't perform label encoding.

3.4.1 Parameters

3.4.1.1 `x`

Type: `numpy.array` or array-like

This is the input data.

3.4.1.2 `y`

Type: `numpy.array` or array-like

This is the target.

3.4.1.3 `model`

Type: `tpot.TPOTClassifier`

Default value is `TPOTClassifier(generations=5, population_size=50, verbosity=2)`. This is a TPOTClassifier object. You can pass a TPOTClassifier object with different parameter configurations as per your requirement. In general, increasing the `generations` and `population_size` increases the model's accuracy. See the TPOTClassifier for more details.

3.4.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `Dill` files, as well as the `API.py` and `helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the internet.

3.4.2 Returns

Returns a `rml` object. `rml.d` is always null. `rml.model` and `rml.m_tpot` are always populated.

3.4.3 Files Created

3.4.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `Dill`.

3.4.3.2 `API.py`

This is the actual `Flask` API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

3.5 `RapidML.rapid_regressor_arr`

The `rapid_regressor_arr` function is similar to the `rapid_regressor`, except rather receiving the features and targets as a single input DataFrame, the function receives the features as X (type numpy.array), and the target as Y (type `numpy.array`). Another important point of difference is that this function doesn't perform label encoding.

3.5.1 Parameters

3.5.1.1 `X`

Type: `numpy.array` or array-like

This is the input data.

3.5.1.2 `Y`

Type: `numpy.array` or array-like

This is the target.

3.5.1.3 `model`

Type: `tpot.TPOTRegressor`

Default value is `TPOTRegressor(generations=5, population_size=50, verbosity=2)`. This is a TPOTRegressor object. You can pass a TPOTRegressor object with different parameter configurations as per your requirement. In general, increasing the `generations` and `population_size` increases the model's accuracy. See the TPOTRegressor for more details.

3.5.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `Dill` files, as well as the `API.py` and `helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the internet.

3.5.2 Returns

Returns a `rml` object. `rml.d` is always null. `rml.model` and `rml.m_tpot` are always populated.

3.5.3 Files Created

3.5.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `Dill`.

3.5.3.2 `API.py`

This is the actual `Flask` API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

3.6 `RapidML.rapid_udm`

This allows RapidML to be a versatile model in the hands of experienced Data Scientists and developers. It works similarly to the `rapid_regressor` or the `rapid_classifier` wherein a single DataFrame is passed which contains the input data as well as the target (which is the last column).

However, it allows the user to provide a `sklearn` model of their choice. Depending on the user's choice, label encoding is done or ignored. The model that is supplied is then fitted (trained) on the input data and then stored, by populating the `rml.model` attribute.

3.6.1 Parameters

3.6.1.1 `df`

Type: `pandas.DataFrame`

This is the input DataFrame provided by the users as the training features as well as the initial columns and the target as the last column on the DataFrame.

3.6.1.2 `model`

Type: `sklearn` model

This may be any model which supports the syntax `sklearn.model.fit(X,y)` where X is input data and y is target.

3.6.1.3 `le`

Type: `str`

The default value is `'Yes'`. If `le` is `'Yes'`, then RapidML will labelencode the input DataFrame supplied as `df`, and store the `LabelEncoder` in a `defaultdict`. Or, if `le` is `'No'` then LabelEncoding will not be done. For any other value of `le`, a value error will be raised.

3.6.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `dill` files, as well as the `API.py and ``helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the web API.

3.6.2 Returns

Returns a `rml` object. If `le` is `'Yes'` then `rml.d` is populated, otherwise, it is null. `rml.model` is always populated, while `rml.m_tpot` is always empty.

3.6.3 Files Created

3.6.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `dill`.

3.6.3.2 `d`

This is the `DefaultDict` (like `dict`) containing the LabelEncoder used to encode the labels in the DataFrame. It has been saved after serialization via `dill`.

3.6.3.3 `df`

This is the skeletal DataFrame, which contains only headers and no data. It has been saved after serialization via `dill`.

3.6.3.4 `dt`

This is a list containing the data types of the columns in the input `DataFrame`. It has been saved after serialization via `dill`.

3.6.3.5 `f`

This is a string containing a dummy input value and can be fed to the API as an URL argument. It is the second row of the input `DataFrame`, converted to a string. It has been saved after serialization via `dill`.

3.6.3.6 `API.py`

This is the actual Flask-API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

3.6.3.7 `helper.py`

This is a helper module used by `API.py` and performs the actual predictions using the RapidML generated model.

3.7 `RapidML.rapid_udm_arr`

The `rapid_udm _arr` function is similar to the `rapid_udm`, except rather receiving the features and targets as a single input DataFrame, the function receives the features as X (type numpy.array), and the target as Y (type `numpy.array`). Another important point of difference is that this function doesn't perform label encoding.

3.7.1 Parameters

3.7.1.1 `X`

Type: `numpy.array` or array-like

This is the input data.

3.7.1.2 `Y`

Type: `numpy.array` or array-like

This is the target.

3.7.1.3 `model`

Type: `sklearn` model

This may be any model which supports the syntax `sklearn.model.fit(X,y)` where X is input data and y is target.

3.7.1.4 `name`

Type: `str`

Default value is `"RapidML_Files"`. The value of the string is the name of the directory in which RapidML creates for storing the machine learning models, LabelEncoder dictionary, skeletal input DataFrame, a datatype list and a dummy user input as serialized `Dill` files, as well as the `API.py` and `helper.py` scripts. This directory is to be uploaded to a web-server, in order to serve (use) the model generated by RapidML for making predictions via the internet.

3.7.2 Returns

Returns a `rml` object. `rml.model` is always populated. `rml.d` and `rml.m_tpot` are always null.

3.7.3 Files Created

3.7.3.1 `model`

This is the Machine Learning model generated by RapidML which is saved after being serialized via `Dill`.

3.7.3.2 `API.py`

This is the actual `Flask` API used by the server for accepting user inputs, making predictions on the basis of the inputs and returning the predictions.

CHAPTER 4

Examples

Note that all the IPython notebooks used in the example are available here: https://github.com/ritabratamaiti/RapidML/tree/master/Examples

4.1 ASD (Autism Spectrum Disorder) Detection

GitHub: https://github.com/ritabratamaiti/Autism-Detection-API

This project utilizes RapidML to detect ASD cases in adults. The training data consists of responses provided by the patients on the AQ-10 questionnaire. RapidML is utilized for selecting, training, serializing and packaging a high accuracy classifier. The files directory generated by RapidML, containing the packaged model is then uploaded to a WSGI server (See various deployment options here: http://flask.pocoo.org/docs/1.0/deploying/).

PythonAnywhere (https://www.pythonanywhere.com) was used in this project. The `builder_script.py` utilizes RapidML.

```
import RapidML
import os
import pandas as pd


# This Autism Screening Adult Data Set is from UCI Machine Learning Repository and is␣
↪available here: https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult

df = pd.read_csv('out.csv')
df = df.drop(columns = ['Unnamed: 0'])
df.head()

ml_model = RapidML.rapid_classifier(df,name='ASDapi')
```

Note: The training data is an Autism Screening Adult DataSet from UCI Machine Learning Repository and is available here: https://archive.ics.uci.edu/ml/datasets/Autism+Screening+Adult

The code generates the following output.

```
RapidML, Version: 0.1, Author: Ritabrata Maiti


       .---.        .-----------
      /     \  __  /    ------
     / /     \(  )/    -----
    //////   ' \/ `   ---
   //// / // :    : ---
  // /   /  /`    '--
 //          //..\
        ====UU====UU====
            '//||\\`
              ''``

Warning: xgboost.XGBClassifier is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.

Using the RapidML Classifier; Experimental, For Issues Contact Author:␣
↪ritabratamaiti@hiretrex.com
Label Encoding is being done....

Training....

Generation 1 - Current best internal CV score: 1.0
Generation 2 - Current best internal CV score: 1.0
Generation 3 - Current best internal CV score: 1.0
Generation 4 - Current best internal CV score: 1.0
Generation 5 - Current best internal CV score: 1.0

Best pipeline: DecisionTreeClassifier(input_matrix, criterion=entropy, max_depth=2,␣
↪min_samples_leaf=4, min_samples_split=6)

Sample Output from input dataframe:
1,1,0,1,0,0,1,1,0,1,6,35.0,f,White-European,no,yes,United States,no,Self,NO
```

The generated model, scripts and serialized files are stored in the directory: `ASDapi`.This directory is uploaded to a WSGI server, for making cloud predictions.

Note: This is a complete project, and some parts (such as the creation of the Android application) is outside the scope of RapidML documentation. Please visit the project on Github for more details.

4.2 Boston House Prices

Let's say we are building a machine learning model to run on the cloud and predict housing prices in an area, using parameters such as crime rates, business development, pollution metrics etc. We will be using the Boston House Prices dataset, due to its wide availability and usage within machine learning academia. Dataset description here: https://www.kaggle.com/c/boston-housing

Note: We will be using `sklearn.datasets` for easy loading of the Boston-housing dataset within Python.

Since we are predicting prices, it is clearly a regression problem. We will be using `RapidML.rapid_regressor_arr` for this task.

```python
# coding: utf-8

from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
import RapidML

housing = load_boston()
X_train, X_test, y_train, y_test = train_test_split(housing.data, housing.target,
                                                    train_size=0.75, test_size=0.25)

model = RapidML.rapid_regressor_arr(X_train, y_train)

print(model.m_tpot.score(X_test, y_test))
```

The following output is generated.

```
RapidML, Version: 0.1, Author: Ritabrata Maiti


       .---.        .------------
      /     \  __  /    ------
     / /     \(  )/    -----
    //////   ' \/ `   ---
   //// / // :    : ---
  // /   /  /`    '--
 //          //..\\
        ====UU====UU====
            '//||\\\`
              ''``

Warning: xgboost.XGBClassifier is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.
Warning: xgboost.XGBRegressor is not available and will not be used by TPOT.

Using RapidML Regressor with arrays, Inputs will not be label encoded.; Experimental,␣
↪For Issues Contact Author: ritabratamaiti@hiretrex.com

Training....

Generation 1 - Current best internal CV score: -11.913707598413463
Generation 2 - Current best internal CV score: -11.913707598413463
Generation 3 - Current best internal CV score: -11.913707598413463
Generation 4 - Current best internal CV score: -11.913707598413463
Generation 5 - Current best internal CV score: -11.404014702360742

Best pipeline: GradientBoostingRegressor(input_matrix, alpha=0.75, learning_rate=0.1,␣
↪loss=huber, max_depth=3, max_features=1.0, min_samples_leaf=5, min_samples_split=4,␣
↪n_estimators=100, subsample=0.6000000000000001)
-10.908425630183695
```

As we can see in this example, a score of `-10.908425630183695` has been achieved. Do note that different models may be generated on a separate program run and hence the scores may fluctuate by a small margin (approximately 1% or so).

In the directory `RapidML_files`, the model file and `API.py` script has been generated which can be uploaded to a WSGI server (with `Flask` support) to perform cloud predictions.

4.3 Using RapidML to build a neural network (For recognizing hand-written digits)

This example serves to demonstrate the versatility of RapidMl, by using udm(User Defined Models). Do note that we will be using matplotlib to visualise the digits' images. In this example, we use `RapidML.rapid_udm_arr` in order to feed a neural network classifier (`sklearn.neural_network.MLPClassifier`) as the machine learning model. We use the `digits` dataset from `sklearn.datasets`, and train the neural network on half the data. The other half is used for testing and visualization.

The following are Jupyter Notebook cells and their corresponding output.

```
Using RapidML with User Defined Models and Arrays, Inputs will not be label encoded;␣
↪note that the model provided by the user should be a Scikit_learn model and not a␣
↪TPOT object.; Experimental, For Issues Contact Author: ritabratamaiti@hiretrex.com

Training....
```

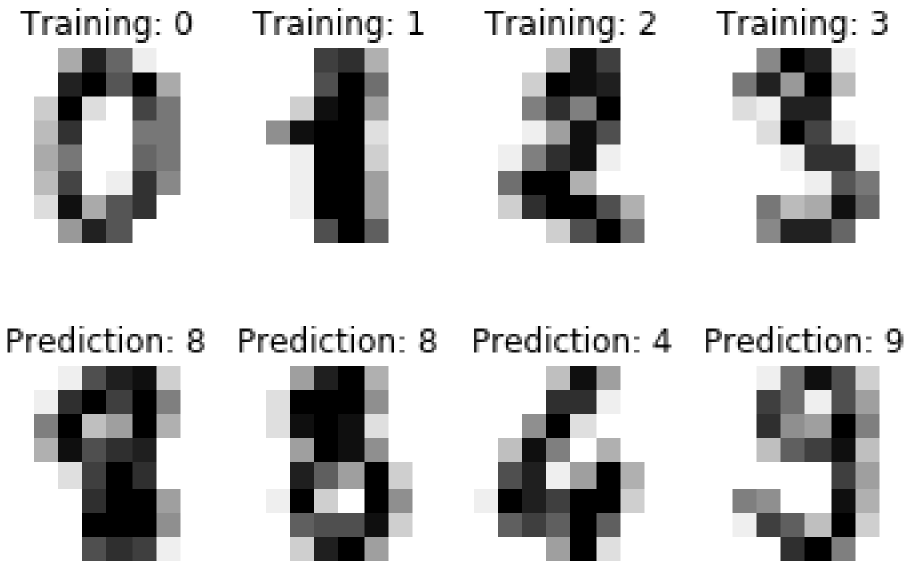
```
C:UsersRitabrata MaitiAnaconda3libsite-packagessklearnneural_
↪networkmultilayer_perceptron.py:564: ConvergenceWarning: Stochastic␣
↪Optimizer: Maximum iterations (200) reached and the optimization hasn't␣
↪converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Classification report for classifier MLPClassifier(activation='relu', alpha=1, batch_
↪size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False):
             precision    recall  f1-score   support

          0       0.99      0.97      0.98        88
          1       0.95      0.92      0.94        91
          2       0.99      0.98      0.98        86
          3       0.96      0.85      0.90        91
          4       0.99      0.89      0.94        92
          5       0.93      0.96      0.94        91
          6       0.91      0.99      0.95        91
          7       0.95      0.99      0.97        89
          8       0.93      0.94      0.94        88
          9       0.86      0.96      0.91        92

avg / total       0.95      0.94      0.94       899


Confusion matrix:
[[85  0  0  0  1  0  2  0  0  0]
 [ 0 84  0  1  0  1  0  0  0  5]
 [ 1  0 84  1  0  0  0  0  0  0]
 [ 0  0  1 77  0  3  0  4  6  0]
 [ 0  0  0  0 82  0  6  0  0  4]
 [ 0  0  0  0  0 87  1  0  0  3]
 [ 0  1  0  0  0  0 90  0  0  0]
 [ 0  0  0  0  0  1  0 88  0  0]
```

(continues on next page)

(continued from previous page)

```
 [ 0  3  0  0  0  0  0  0 83  2]
 [ 0  0  0  1  0  2  0  1  0 88]]
```



Note: If you wish to use the model as a flask API, do remember to flatten the image, to turn the data in a (samples, feature) matrix, and then convert to URL argument. However, this method hasn't undergone complete testing and is not guaranteed to work. However, it is possible to modify the `API.py` file to say, accept an image and then flatten it within the script itself.

CHAPTER 5

Support

The easiest way for you to get support is by opening a new issue on Github at https://github.com/ritabratamaiti/RapidML/issues or by contacting the author at ritabratamaiti@hiretrex.com

Github: https://github.com/ritabratamaiti/RapidML/issues

Email: ritabratamaiti@hiretrex.com

CHAPTER 6

Contributing

Please feel free to improve RapidML's code on GitHub

The preferred way to modify RapidML source is to fork the repository and submit the pull request. However, before doing so please update the documentation and release notes, available in the `docs` directory.

Areas of improvement:

- Implementing an elegant way of building reStructured Text documentation using Sphinx. The current method involves building the HTML files in the `docsource` directory and then using a redirection index to point to the actual index.
- Improve helper tips within RapidML functions.

For any questions, submit a question to: https://github.com/ritabratamaiti/RapidML/issues

Or contact: ritabratamaiti@hiretrex.com

CHAPTER 7

Indices and tables

- genindex
- modindex
- search