bulbea

Release 0.1.0

May 09, 2017

Contents

> *"Deep Learning based Python library for Stock Market Prediction and Modelling."*

Release: v0.1.0 (*Installation*) **bulbea** is an Open Source Python module (released under the *Apache 2.0 License*) that consists a growing collection of statistical, visualization and modelling tools for financial data analysis and prediction using deep learning.

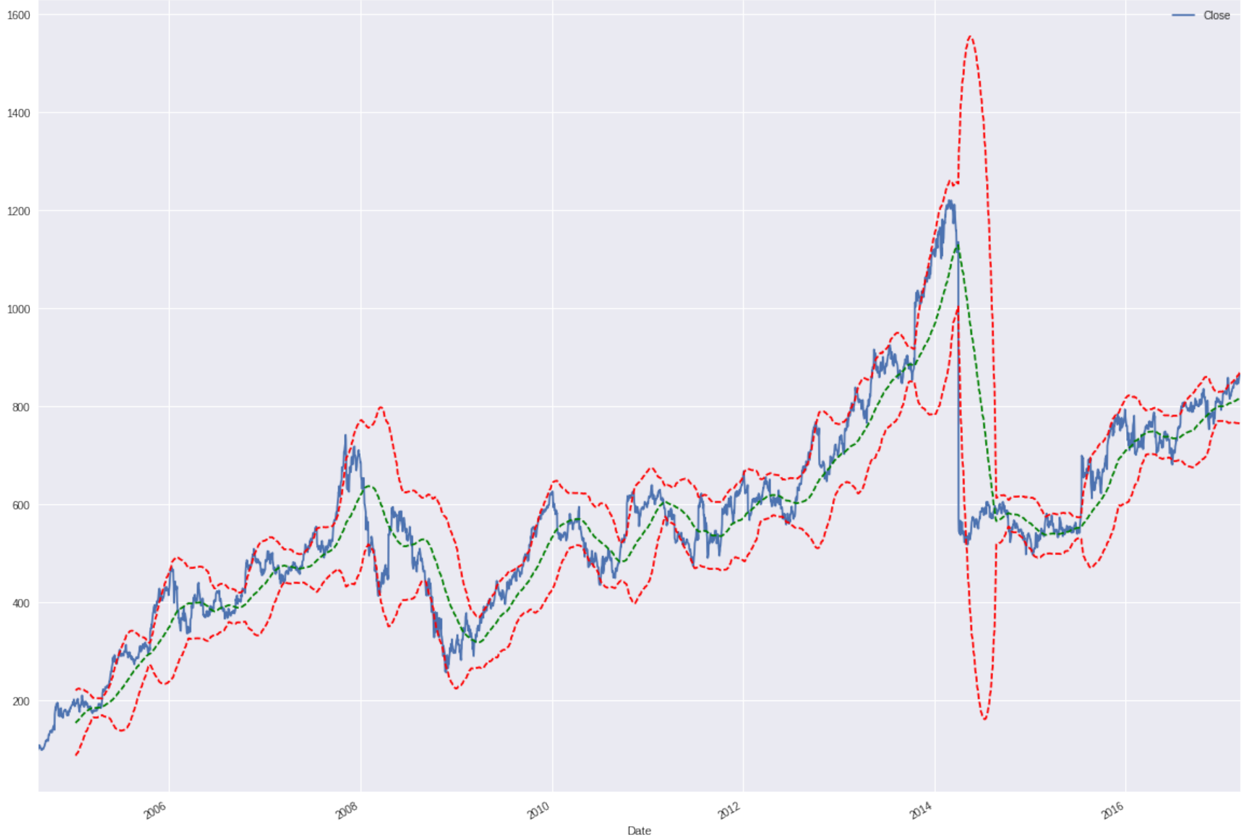
bulbea helps you with

Financial Data Loading

```
>>> import bulbea as bb
>>> share = bb.Share('YAHOO', 'GOOGL') # Get Google's historical data from Yahoo's␣
↪database
>>> share.data
                  Open        High         Low       Close      Volume  Adjusted Close
Date
2004-08-19   99.999999  104.059999   95.959998  100.339998  44659000.0       50.220219
2004-08-20  101.010005  109.079998  100.500002  108.310002  22834300.0       54.209210
2004-08-23  110.750003  113.479998  109.049999  109.399998  18256100.0       54.754754
...
```

Statistical Vizualization

```
>>> share.plot(bollinger_bands = True, period = 100, bandwidth = 2)
```



bulbea is created and currently maintained by Achilles Rasquinha.

bulbea officially supports Python 2.7 and 3.5.

CHAPTER 1

Guide - User

Introduction

What's in the name?

bulbea is a portmanteau of the very nature of a stock market - the bull and the bear. Hence, the name.

License

bulbea is released under the Apache 2.0 License.

```
Copyright 2017 Achilles Rasquinha <achillesrasquinha@gmail.com>

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

Installation

Building from source

bulbea is actively developed on GitHub and is always avaliable.

You can clone the base repository with `git` as follows:

```
$ git clone git@github.com:achillesrasquinha/bulbea.git
```

Optionally, you could download the tarball or zipball as follows:

For Linux Users

```
$ curl -OL https://github.com/achillesrasquinha/tarball/bulbea
```

For Windows Users

```
$ curl -OL https://github.com/achillesrasquinha/zipball/bulbea
```

Install necessary dependencies

```
$ pip install -r requirements.txt
```

bulbea depends on Keras which thereby depends on TensorFlow as a backend. You may have to manually install TensorFlow as follows:

```
$ pip install tensorflow # CPU-only
```

OR

```
$ pip install tensorflow-gpu # GPU-only, requires NVIDIA CUDA and cuDNN
```

Then, go ahead and install **bulbea** in your site-packages as follows:

```
$ python setup.py install
```

Check to see if you've installed **bulbea** correctly.

```
>>> import bulbea as bb
```

Quickstart

Waiting to make some money? We introduce you to a quick way of building your first prediction model.

Create a `Share` object

The canonical way of importing bulbea as follows:

```
>>> import bulbea as bb
```

Go ahead and create a share object.

```
>>> share = bb.Share(source = 'YAHOO', ticker = 'GOOGL')
```

CHAPTER 2

Guide - API

Developer Interface

Entities

class `bulbea.`**`Share`**(*source*, *ticker*, *start=None*, *end=None*, *latest=None*, *cache=False*)

A user-created *`Share`* object.

Parameters

- **`source`** (`str`) – *source* symbol for economic data
- **`ticker`** (`str`) – *ticker* symbol of a share
- **`start`** (`str`) – starting date string in the form YYYY-MM-DD for acquiring historical records, defaults to the earliest available records
- **`end`** (`str`) – ending date string in the form YYYY-MM-DD for acquiring historical records, defaults to the latest available records
- **`latest`** (`int`) – acquires the latest N records

Example

```
>>> import bulbea as bb
>>> share = bb.Share(source = 'YAHOO', ticker = 'GOOGL')
>>> share.data.sample(1)
            Open       High        Low  Close      Volume  Adjusted Close
Date
2003-05-15  18.6  18.849999  18.470001  18.73  71248800.0        1.213325
```

`bollinger_bands`(*attrs='Close'*, *period=50*, *bandwidth=1*)

Returns the Bollinger Bands (R) for each attribute.

Parameters

- **`attrs`** (`str`, `list`) – *str* or *list* of attribute name(s) of a share, defaults to *Close*

- **period** (int) – length of the window to compute moving averages, upper and lower bands
- **bandwidth** (int) – multiple of the standard deviation of upper and lower bands

Example

```
>>> import bulbea as bb
>>> share     = bb.Share(source = 'YAHOO', ticker = 'AAPL')
>>> bollinger = share.bollinger_bands()
>>> bollinger.tail()
            Lower (Close)  Mean (Close)  Upper (Close)
Date
2017-03-07     815.145883    831.694803     848.243724
2017-03-08     816.050821    832.574004     849.097187
2017-03-09     817.067353    833.574805     850.082257
2017-03-10     817.996674    834.604404     851.212135
2017-03-13     819.243360    835.804605     852.365849
```

plot (*attrs='Close'*, *global_mean=False*, *bollinger_bands=False*, *period=50*, *bandwidth=1*, *subplots=False*, **args*, ***kwargs*)

Parameters attrs – *str* or *list* of attribute names of a share to plot, defaults to *Close* attribute

Example

```
>>> import bulbea as bb
>>> share = bb.Share(source = 'YAHOO', ticker = 'AAPL')
>>> share.plot()
```

save (*format_='csv'*, *filename=None*)

Parameters format (str) – type of format to save the Share object, default 'csv'.

update (*start=None*, *end=None*, *latest=None*, *cache=False*)

Update the share with the latest available data.

Example

```
>>> import bulbea as bb
>>> share = bb.Share(source = 'YAHOO', ticker = 'AAPL')
>>> share.update()
```

class bulbea.**Stock**

Modelling

CHAPTER 3

Blog

Data, Data Everywhere

> *"In God we trust, all others must bring data."* - W. Edwards Deming

How data is stored

Data streams itself right from when the gates of a stock exchange open to when it closes. Such data contains vital information that is archived each day. Some of the many types of information recieved after trading hours are - *opening price*, *closing price*, *volumne of shares*, *highest price*, *lowest price*, etc. for each enterprise.

bulbea helps you access such information (both - archived and the latest). Simply create a *Share* with a known `source` and `ticker` as follows:

```
>>> import bulbea as bb
>>> share = bb.Share(source = 'YAHOO', ticker = 'GOOGL')
>>> share.data
                  Open        High         Low       Close      Volume  Adjusted Close
Date
2004-08-19   99.999999  104.059999   95.959998  100.339998  44659000.0       50.220219
2004-08-20  101.010005  109.079998  100.500002  108.310002  22834300.0       54.209210
2004-08-23  110.750003  113.479998  109.049999  109.399998  18256100.0       54.754754
...
```

Data is accessed through the `Quandl` API stored remotely at sources in the form of CSV (Comma-Seperated Values) files. Information retrieved from such a CSV file is then wrapped around a `pandas.DataFrame` object.

Comma, Seperated, Value?

CSV files store tabular data in simple plain text (well, fits the need). Each row containing values associated to each attribute of a table are stored in a single line, where each value is seperated by a delimiter, you guessed it right, a comma. For instance, a data set containing the weight (in kilograms) and height (in inches) of members of my family would look something like the following:

```
weight,height
87,6.2
51,5.8
68,5.9
...
```

Almost always, the top-most line (also called as *the header*) should denote the attribute names seperated by the delimiter.

You can save a share object in a CSV format as follows:

```
>>> share.save()
```

By default, the *`save`* method saves a share as a CSV file in the working directory with a file name of the format - `<source>_<ticker>_<start>_<end>.csv`. You could also name the file anything you like as follows:

```
>>> share.save(filename = 'mycsvfile.csv')
```

`pandas.DataFrame`

Vizualizing the Market

Artificial Neural Networks

> *"All models are wrong, but some are useful."* - George E. P. Box

Recurrent Neural Networks

A vanilla Recurrent Neural Network (hereby, RNN) is a kind of an Artificial Neural Network that considers a scenario - *at which time-step did you feed the input?*

Index