csvinsight Documentation

Release 0.3.3

Michael Penkov

Jun 10, 2021

Contents

Contents:

CHAPTER 1

csvinsight

Fast & simple summary for large CSV files

- Free software: MIT license
- Documentation: https://csvinsight.readthedocs.io.

1.1 Features

- Calculates basic stats for each column: max, min, mean length; number of non-empty values
- Calculates exact number of unique values and the top 20 most frequent values
- Supports non-orthogonal data (list fields)
- Works with very large files: does not load the entire CSV into memory
- Fast splitting of CSVs into columns, one file per column
- Multiprocessing-enabled

1.2 Example Usage

Given a CSV file:

```
bash-3.2$ cat tests/sampledata.csv
name|age|fave_color
Alexey|33|red;yellow
```

(continues on next page)

(continued from previous page)

```
Boris|31|blue
Valentina|0|
```

you can obtain a CsvInsight report with:

```
bash-3.2$ csvi tests/sampledata.csv --list-fields fave_color
CSV Insight Report
Total # Rows: 3
Column counts:
        3  columns ->  3 rows

Report Format:
Column Number. Column Header -> Uniques: # ; Fills: # ; Fill Rate:
Field Length: min #, max #, average:
 Top n field values -> Dupe Counts


1. name -> Uniques: 3 ; Fills: 3 ; Fill Rate: 100.0%
    Field Length:  min 5, max 9, avg 6.67
        Counts      Percent  Field Value
        1           33.33 %  Valentina
        1           33.33 %  Boris
        1           33.33 %  Alexey

2. age -> Uniques: 3 ; Fills: 3 ; Fill Rate: 100.0%
    Field Length:  min 1, max 2, avg 1.67
        Counts      Percent  Field Value
        1           33.33 %  33
        1           33.33 %  31
        1           33.33 %  0

3. fave_color -> Uniques: 4 ; Fills: 3 ; Fill Rate: 75.0%
    Field Length:  min 0, max 6, avg 3.25
        Counts      Percent  Field Value
        1           25.00 %  yellow
        1           25.00 %  red
        1           25.00 %  blue
        1           25.00 %  NULL
```

Since CSV comes in different flavors, you may need to tweak the underlying CSV parser's parameters to read your file successfully. CSVInsight handles this via CSV dialects. For example, to read a comma-separated file, you would use the following command:

```
bash-3.2$ csvi your/file.csv --dialect delimiter=,
```

You may combine as many dialect parameters as needed:

```
bash-3.2$ csvi your/file.csv --dialect delimiter=, quoting=QUOTE_NONE
```

For a full list of dialect parameters, see the documentation for Python's csv module. Constant values like QUOTE_NONE are resolved automagically.

Once you've discovered the winning parameter combination for your file, save it to a YAML file:

```
list_fields:
  - fave_color
  - another_field_name
```

(continues on next page)

(continued from previous page)

```
list_separator: ;
dialect:
  - "delimiter=|"
  - "quoting=QUOTE_NONE"
```

You can then invoke CSVI as follows:

```
bash-3.2$ csvi your/file.csv --config your/config.yaml
```

1.3 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install csvinsight, run this command in your terminal:

```
$ pip install csvinsight
```

To install csvinsight with Jupyter notebook support, run the following command:

```
$ pip install csvinsight[notebook]
```

This is the preferred method to install csvinsight, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for csvinsight can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/ProfoundNetworks/csvinsight
```

Or download the tarball:

```
$ curl  -OL https://github.com/ProfoundNetworks/csvinsight/tarball/master
```

Once you have a copy of the source, you can install it with:

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

CHAPTER 3

Usage

CsvInsight is primarily a command-line application, but it can be used as a library.

To use csvinsight in a project:

```
import csvinsight
```

Todo: Describe the main entry points to the library.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/ProfoundNetworks/csvinsight/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

csvinsight could always use more documentation, whether as part of the official csvinsight docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ProfoundNetworks/csvinsight/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *csvinsight* for local development.

1. Fork the *csvinsight* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/csvinsight.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv csvinsight
$ cd csvinsight/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 csvinsight tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/ProfoundNetworks/csvinsight/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests
```

CHAPTER 5

Credits

5.1 Development Lead

- Michael Penkov <mpenkov@profound.net>

5.2 Contributors

- Oleg Pankov <opankov90@gmail.com>
- Artem Golubin <me@rushter.com>

CHAPTER 6

History

6.1 Unreleased

6.2 0.3.3 (2020-12-02)

- Handle numeric quoting parameter, e.g. "–dialect quoting=3"

6.3 0.3.2 (2019-07-01)

- Set the field size limit to sys.maxsize

6.4 0.3.1 (2019-06-26)

- Make Jupyter notebook an optional dependency

6.5 0.3.0 (2018-07-11)

- Added –most-common parameter (resolved Issue #14)
- Added –no-tiny parameter
- Refactored temporary file naming
- Improve error message when handling empty CSV files
- Fixed "Argument list too long" error (Issue #15)
- Added –json parameter

- Added –ipynb parameter to generate IPython notebook

6.6 0.2.3 (2017-12-09)

- Fix bug: Unicode column names now work under Py2

6.7 0.2.2 (2017-12-04)

- Fix bug: Unicode characters no longer break CsvInsight on Py2

6.8 0.2.1 (2017-11-27)

- Fix bug: opening gzipped files with Py3 now works

6.9 0.2.0 (2017-11-25)

- Split files using gsplit and process them in parallel for faster processing
- No longer work with streams; works exclusively with files
- Get rid of csvi_summarize and csvi_split entry points
- Integrated plumbum for cleaner pipelines
- Fixed issue #11: added support for more CSV parameters via the –dialect option
- Fixed issue #10: reading from empty files no longer raises StopIteration
- Fixed issue #8: use the correct link to the GitHub project in the documentation
- Fixed issue #2: implemented in-memory mode for smaller files

6.10 0.1.0 (2017-10-29)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search