inforcehub Documentation

Release 0.2.2

Matt Gosden

Nov 07, 2018

Contents

Utilities for data science and customer management

CHAPTER 1

Features

- **colors** module - to enable quick use of inforcehub branded colors
- **anonymize** module - to encrypt customer identifiable data from a dataset and (optionally) create a key

CHAPTER 2

Installation

To install the module in your virtual environment using pip:

```
pip install inforcehub
```

To upgrade your existing inforcehub module installation using pip:

```
pip install -U inforcehub
```

CHAPTER 3

Requirements

Python 3.5 or later

CHAPTER 4

Documentation

Full documentation can be found at http://inforcehub.rtfd.org

4.1 Contents:

4.1.1 Overview

The **inforcehub** package contains various utilities and tools we find useful in running data science tasks for customer management at insurers.

We all benefit from good open-source solutions. Therefore as a business we want to contribute some of our solutions back to the community for those that may find them useful.

Contributions are welcome - please see the *contributing section* for more info.

Sub-modules

Anon

A module to anonymize and pseudo-anonymize customer datasets. Strips out the confidential data from a pandas dataframe.

Find out more.

Colors

This is a module of shortcuts to easily import inforcehub branded colors. This makes consistent coloring easier in matplotlib and jupyter notebooks.

Find out more.

Examples

Jupyter notebooks using these modules can be found in the GitHub repository.

4.1.2 Anonymize

This module provides methods to **anonymize or pseudo-anonymize a dataset**.

Anonymizing the dataset means removing all the confidential data and **not retaining any key** to enable it to be reconstructed. Also the remaining patterns in the data should not enable the identity of individuals to be identified via other means.

Pseudo-anonymizing the dataset also removes all the confidential data from the dataset. But we also retain a key or separate dataset enabling us to decode the original entities if required at some point in the future.

The anonymizer does not delete the confidential data. Instead it **one-way encrypts (hashes) the data** so that the original text or numbers cannot be retrieved.

Encryption is better than deletion because the pattern of data is useful information we want our models to be able to use. For example it may be that one customer has multiple contracts. Deleting the customer number loses the information that these contracts are linked but encryption retains those relationships.

Encryption method

The method used is similar to password hashing.

- A **salt** is randomly created for this encryption run by the system. If desired for reproducability, the user can override the salt with a passphrase of their choice.
- The salt is prepended to each value to be encrypted
- The encryption algorithm converts the salted value into a long hexidecimal hash. This is our encrypted value.

Encryption is done using the **python hashlib package**. Using the **md5 algorithm**. More details can be found at https://docs.python.org/3/library/hashlib.html

The randomized salt is created using the **bcrypt package**. More details can be found at https://pypi.org/project/bcrypt/

Note: In a password setting, usually for each individual encryption, a new random salt value would be created. This way you get very strong encryption with a key (the salt) that changes for every password.

However in our algorithm we only set the salt once for the full dataset. This is on purpose so that we can retain the relationship structure between values whilst still providing a good level of encryption by using a hash algorithm and a salt value.

What this means is that the same customer occuring in multiple rows will have the same hashed value using our method.

Usage

Import the anonymizer class:

```
from inforcehub import Anonymize
```

On initializing, the object will use a new randomized salt passphrase:

```
anon = Anonymize()
```

You need to decide which columns in your Pandas dataframe should be anonymized.

To transform a dataframe use the **transform** method on the anon object. We pass in the dataframe to be transformed, and a list of the columns to be anonymized:

```
anon.transform(df, ['columnA', 'columnZ'])
```

The dataframe **df** itself will be anonymized to save improve memory usage and speed for large datasets. To retain a copy of the original make a deep copy of the dataframe before transforming it:

```
original_df = df.copy(deep=True)  # Do this first
```

If **pseudo-anonymization** is required instead of full anonymization the **lookup** dataframe of encrypted and unencrypted values is returned. This can be used later as a lookup to return to the confidential data:

```
lookup_df = anon.transform(df, ['columnA', 'columnZ'])
```

Module details

class inforcehub.anonymize.**Anonymize**(*salt=None*)

A class that will transform a Pandas dataframe into an anonymized dataframe using the python hmac package for encryption.

When instantiating an object of this class, the **salt** init attribute can be specified which enables encryption results to be reproduced. If none is supplied, a randomized password is created instead for security.

Param str salt: an optional passphase - if empty, salt will be randomized

encode(*text*)

Provides the encryption of a single number, text or date

Param text: the text, value or date to be encrypted

Returns the hexidecimal encrypted value

Return type str

transform(*df*, *columns*, *verbose=False*)

Encrypts the selected columns in a dataframe and returns an anonymized dataframe.

If required, also returns a dataframe of column pairs showing the encrypted and original data to be used as a pseudo-anonymization key.

Param pd.DataFrame() df: A Pandas dataframe to be transformed

Param list columns: A list of columns to be transformed

Param str verbose: (default=False) If true will print status

Returns a pseudo-anonymization lookup table

Return type pd.DataFrame()

4.1.3 InforcehubColors

This provides simple utilities to return **inforcehub branded colors** for use across applications such as matplotlib.

Usage

Import the color class and create an instance:

```
from inforcehub import InforcehubColors

ifh = InforcehubColors()
```

On this you can then get the specific colors by name:

```
ifh.pink                # Hex code for our pink
ifh.blue                # Hex code for our blue
```

Or you can see the list of colors available:

```
ifh.show()              # Names of all colors and neutrals
ifh.show('core')        # Names of core colors only
ifh.show('neutral')     # Names of neutral colors only
```

Or you can get multiple color hexcodes in a list:

```
ifh.list()              # Hex codes for everything
ifh.list('core')        # Hex codes for core colors
ifh.list('colors')      # Hex codes for all colors
ifh.list('neutral')     # Hex codes for neutral colors
```

Note that these methods are class methods on the **InforcehubColors** object. So you do not have to create an instance first. But we suggest doing it using an instance as above as it will keep your code neater.

Module details

class inforcehub.colors.**InforcehubColors**

A class that will return the inforcehub brand colors as single hex codes or lists for use in various packages such as matplotlib.

The objective of this package is to make it easier and quicker to create consistent-looking charts using branded colors

classmethod list(*sublist='all'*)

Returns a list of the color hex codes for use in Matplotlib or other tools which want a set of colors to choose from

Param str sublist: (default='all') options 'colors'/'core'/'neutral'/'all'

Returns a list of color hex codes

Return type list

classmethod show(*sublist='all'*)

Returns a list of the color names

Param str sublist: (default='all') options 'colors'/'core'/'neutral'/'all'

Returns a list of color names

Return type list

4.1.4 Installation

At the command line either via easy_install or pip:

```
$ easy_install inforcehub
$ pip install inforcehub
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv inforcehub
$ pip install inforcehub
```

Requirements

Package requires **Python 3.3 or later**

4.1.5 Usage

To use the **inforcehub** package in a project:

```
import inforcehub
```

Examples

Jupyter notebooks using these modules can be found in the GitHub repository.

4.1.6 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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/inforcehub/inforcehub/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.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/inforcehub/inforcehub/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 :)

Get Started!

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

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

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

3. Create a branch for local development:

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

Now you can make your changes locally.

4. When you're done making changes, check that your changes pass style and unit tests, including testing other Python versions with tox:

```
$ tox
```

To get tox, just pip install it.

5. Before committing ensure you run *nbstripout* to remove any additional data on the Jupyter notebook examples:

```
$ nbstripout examples/.
```

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.

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 Python 3.5 - 3.7. Check https://travis-ci.org/Inforcehub/inforcehub under pull requests for active pull requests or run the `tox` command and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test test/test_inforcehub.py
```

4.1.7 Credits

Development Lead

- Matt Gosden <matt.gosden@inforcehub.com>
- Gareth Emery <gareth.emery@inforcehub.com>

Contributors

- Adrian Horobet <adrian.horobet@inforcehub.com>
- Michel Abbink <michel.abbink@inforcehub.com>

4.1.8 License

This package is covered by the GNU General Public License v3.

4.1.9 History

0.2.2 (2018-11-07)

- Fixed document build on Read the Docs

0.2.1 (2018-11-02)

- Fixed dependencies in PyPI

0.2.0 (2018-10-19)

- Added Anonymize module to encrypt customer identifiable data in a dataset
- Removed PyPy support due to PyPy and numpy incompatibility

0.1.3 (2018-10-16)

- Minor bug fix

0.1.2 (2018-10-16)

- Colors module refactored and additional tests written
- Documentation updated
- Added support for Python 3.5+

0.1.1 (2018-10-15)

- Documentation added

0.1.0 (2018-10-15)

- First release on PyPI.
- Colors module providing inforcehub brand colors

4.2 Feedback

If you have any suggestions or questions about **inforcehub** feel free to email me at matt.gosden@inforcehub.com.

If you encounter any errors or problems with **inforcehub**, please let me know! Open an Issue at the GitHub http://github.com/inforcehub/inforcehub main repository.

Python Module Index

i

Index