clean$_{df}$ *Documentation*

Release 0.1.0

Nael Aqel

Aug 22, 2023

CONTENTS:

CHAPTER

ONE

CLEAN_DF

Python module to report, clean, and optimize **Pandas Dataframes** effectively.

Full Documentation Here.

1.1 Description and Features

The first step of any data analysis project is to check and clean the data, in this module I implemented a very effiecint code that can:

- Automatically drop columns that have a unique value (these columns are useless, so it will be dropped).
- Report your **Pandas DataFrame** to decide for actions, this report will show:
    1. Duplicated rows report.
    2. Columns' Datatype to optimize memory report.
    3. Columns to convert to categorical report.
    4. Outliers report.
    5. Missing values report.
- Clean the dataframe by dropping columns that have a high ratio of missing values, rows with missing values, and duplicated rows in the dataframe.
- Optimize the dataframe by converting columns to the desired data type and converting categorical columns to 'category' data type.

1.2 Installation

To install `clean_df`, run this command in your terminal:

```
$ pip install clean_df
```

For more information on installation details for this project, please see *Installation*.

1.3 Usage

This module is very easy to use, for a full detailed example please see *Usage*.

1.3.1 Importing the module

```
from clean_df import CleanDataFrame
```

1.3.2 Defining the class

Pass your pandas dataframe to `CleanDataFrame` class:

```
cdf = CleanDataFrame(
        df=df,              # the dataframe to be cleaned
        max_num_cat=5       # maximum number of unique values in a column to be
        )                   # converted to categorical datatype, default is 5
```

1.3.3 Reporting

Call `report` method to see a full report about the dataframe (duplications, columns to optimize its data types, categorical columns, outliers, and missing values:

```
cdf.report(
        show_matrix=True,   # show matrix missing values (from missingno package),␣
↪default is True
        show_heat=True,     # show heat missing values (from missingno package), default␣
↪is True
        matrix_kws={},      # if need to pass any arguments to matrix plot, default is {}
        heat_kws={}         # if need to pass any arguments to heat plot, default is {}
        )
```

1.3.4 Cleaning

Call `clean` method to drop single value columns, high number of missing value columns, duplicated rows, and rows with missing values:

```
cdf.clean(
        min_missing_ratio=0.05,     # the minimum ratio of missing values to drop a␣
↪column, default is 0.05
        drop_nan=True               # if True, drop the rows with missing values after␣
↪dropping columns
                                    # with missingsa above min_missing_ratio
        drop_kws={},                # if need to pass any arguments to pd.DataFrame.
↪drop(), default is {}
        drop_duplicates_kws={}      # same drop_kws, but for drop_duplicates function
        )
```

1.3.5 Optimizing

Call `optimize` method to optimize the dataframe by changing columns' data types based on its values for maximum memory savings:

```
cdf.optimize()
```

1.3.6 Accessing the Cleaned Data DataFrame

```
cdf.df
```

1.4 Contributing

See *Contributing* for contribution details. Feel free to contact me for any subject through my:

- Email
- LinkedIn
- WhatsApp

Also, you are welcomed to visit my personal blog.

1.5 License

Free software: MIT license.

1.6 Documentation

- The full documentation is hosted on my website, and on ReadTheDocs.
- The source code is available in GitHub.

1.7 Credits

- This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.
- Here are additional resources I got a lot from them.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install clean_df, run this command in your terminal:

```
$ pip install clean_df
```

This is the preferred method to install clean_df, 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 clean_df can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/naelaqel/clean_df
```

Or download the tarball:

```
$ curl -OJL https://github.com/naelaqel/clean_df/tarball/master
```

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

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

CHAPTER

THREE

USAGE

In this detailed example, we will use 2018 Airline Delay dataset, saved as `2018.csv`, the dataset is available in Kaggle.

3.1 Importing modules and dataset

```
>>> import pandas as pd
>>> from clean_df import CleanDataFrame
>>> df = pd.read_csv('2018.csv')
>>> df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7213446 entries, 0 to 7213445
Data columns (total 28 columns):
 #   Column               Dtype
---  ------               -----
 0   FL_DATE              object
 1   OP_CARRIER           object
 2   OP_CARRIER_FL_NUM    int64
 3   ORIGIN               object
 4   DEST                 object
 5   CRS_DEP_TIME         int64
 6   DEP_TIME             float64
 7   DEP_DELAY            float64
 8   TAXI_OUT             float64
 9   WHEELS_OFF           float64
 10  WHEELS_ON            float64
 11  TAXI_IN              float64
 12  CRS_ARR_TIME         int64
 13  ARR_TIME             float64
 14  ARR_DELAY            float64
 15  CANCELLED            float64
 16  CANCELLATION_CODE    object
 17  DIVERTED             float64
 18  CRS_ELAPSED_TIME     float64
 19  ACTUAL_ELAPSED_TIME  float64
 20  AIR_TIME             float64
 21  DISTANCE             float64
 22  CARRIER_DELAY        float64
 23  WEATHER_DELAY        float64
```

(continues on next page)

(continued from previous page)

```
 24  NAS_DELAY            float64
 25  SECURITY_DELAY       float64
 26  LATE_AIRCRAFT_DELAY  float64
 27  Unnamed: 27          float64
dtypes: float64(20), int64(3), object(5)
memory usage: 1.5+ GB
```

For our analysis, we will duplicate some rows:

```
df = pd.concat([df, df.iloc[[1, 2, 2, 3]]])
```

The module has only one class `CleanDataFrame`, we will assign it to variable `cdf`:

```
>>> cdf = CleanDataFrame(
                df=df,              # the dataframe to be cleaned
                max_num_cat=20      # maximum number of unique values in a column to be
                )                   # converted to categorical datatype, we will choose 20

Founded useless columns (with single value) ... [Unnamed: 27] columns dropped.
```

Our class dropped the column `Unnamed: 27` automatically because it has one value only (useless).

3.2 Reporting

Now, we can start apply the methods, first lets see the report about dataset:

```
>>> cdf.report(
        show_matrix=True,   # show matrix missing values (from missingno package),␣
↪default is True
        show_heat=True,     # show heat missing values (from missingno package), default␣
↪is True
        matrix_kws={},      # if need to pass any arguments to matrix plot, default is {}
        heat_kws={}         # if need to pass any arguments to heat plot, default is {}
        )

===============
Duplicated Rows
===============
The dataset has 7 duplicated rows, which is 0.0% from the dataset, duplicated rows are:

    FL_DATE     OP_CARRIER      OP_CARRIER_FL_NUM  ORIGIN    DEST      CRS_DEP_TIME   ␣
↪DEP_TIME    DEP_DELAY    TAXI_OUT    WHEELS_OFF    WHEELS_ON    TAXI_IN    CRS_ARR_
↪TIME    ARR_TIME    ARR_DELAY    CANCELLED    CANCELLATION_CODE    DIVERTED    CRS_
↪ELAPSED_TIME    ACTUAL_ELAPSED_TIME    AIR_TIME    DISTANCE    CARRIER_DELAY   ␣
↪WEATHER_DELAY    NAS_DELAY    SECURITY_DELAY    LATE_AIRCRAFT_DELAY
--  ----------  ------------  -------------------  --------  ------  --------------  ----
↪------  -----------  ----------  ------------  -----------  ---------  -------------- ␣
↪----------  -----------  -----------  -------------------  ----------  -----------------
↪--  ---------------------  ----------  ----------  ---------------  ---------------  --
↪---------  ----------------  ---------------------
 1  2018-01-01  UA                           2427  LAS       SFO               1115     ␣
```

(continues on next page)

(continued from previous page)

```
↪  1107           -8          11         1118         1223          7            1254 ␣
↪      1230          -24           0                 nan          0                 ␣
↪99                    83          65         414             nan              nan   ␣
↪      nan              nan                  nan
 2  2018-01-01  UA                          2426  SNA       DEN              1335     ␣
↪  1330           -5          15         1345         1631          5            1649 ␣
↪      1636          -13           0                 nan          0                 ␣
↪134                   126         106         846             nan              nan  ␣
↪       nan              nan                  nan
 3  2018-01-01  UA                          2425  RSW       ORD              1546     ␣
↪  1552            6          19         1611         1748          6            1756 ␣
↪      1754           -2           0                 nan          0                 ␣
↪190                   182         157        1120             nan              nan  ␣
↪       nan              nan                  nan
 1  2018-01-01  UA                          2427  LAS       SFO              1115     ␣
↪  1107           -8          11         1118         1223          7            1254 ␣
↪      1230          -24           0                 nan          0                 ␣
↪99                    83          65         414             nan              nan   ␣
↪      nan              nan                  nan
 2  2018-01-01  UA                          2426  SNA       DEN              1335     ␣
↪  1330           -5          15         1345         1631          5            1649 ␣
↪      1636          -13           0                 nan          0                 ␣
↪134                   126         106         846             nan              nan  ␣
↪       nan              nan                  nan
 2  2018-01-01  UA                          2426  SNA       DEN              1335     ␣
↪  1330           -5          15         1345         1631          5            1649 ␣
↪      1636          -13           0                 nan          0                 ␣
↪134                   126         106         846             nan              nan  ␣
↪       nan              nan                  nan
 3  2018-01-01  UA                          2425  RSW       ORD              1546     ␣
↪  1552            6          19         1611         1748          6            1756 ␣
↪      1754           -2           0                 nan          0                 ␣
↪190                   182         157        1120             nan              nan  ␣
↪       nan              nan                  nan


==============================
Numerical Columns Optimization
==============================
These numarical columns can be down graded:

        columns_to_convert
------  ------------------------------------------------------------------------------------
↪-----------------------------------------------------------------------------------------
↪-------------------------------------------------
uint16  OP_CARRIER_FL_NUM, CRS_DEP_TIME, DEP_TIME, WHEELS_OFF, WHEELS_ON, TAXI_IN, CRS_
↪ARR_TIME, ARR_TIME, ACTUAL_ELAPSED_TIME, AIR_TIME, DISTANCE, CARRIER_DELAY, WEATHER_
↪DELAY, NAS_DELAY, SECURITY_DELAY, LATE_AIRCRAFT_DELAY
int16   DEP_DELAY, ARR_DELAY, CRS_ELAPSED_TIME
uint8   TAXI_OUT, CANCELLED, DIVERTED
```

(continues on next page)

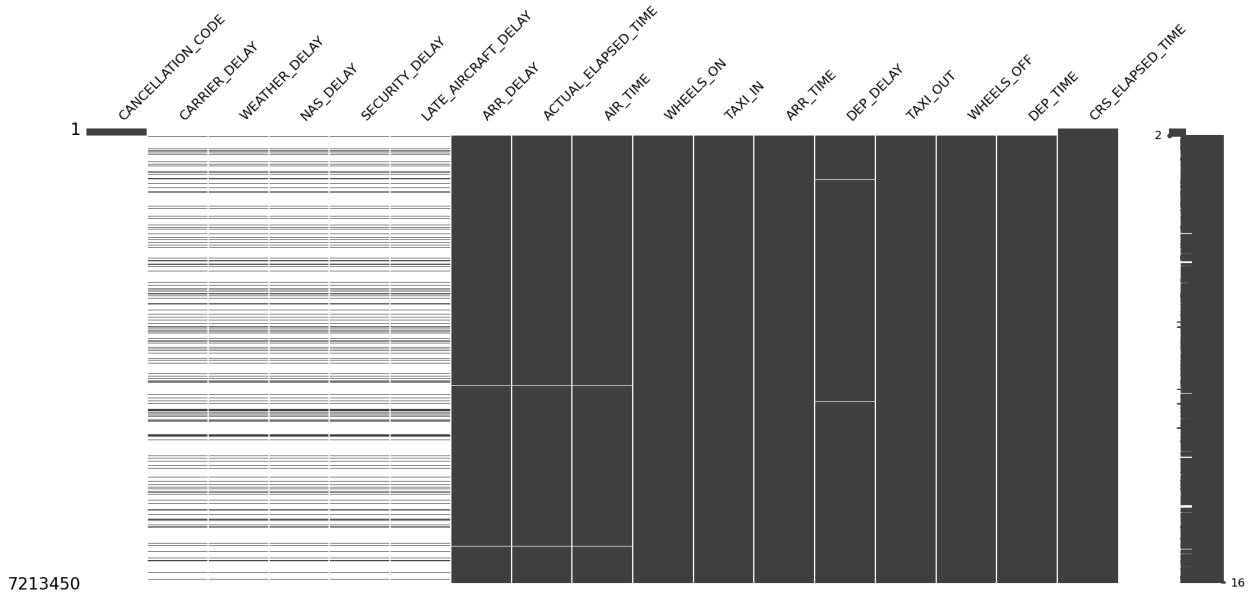
(continued from previous page)

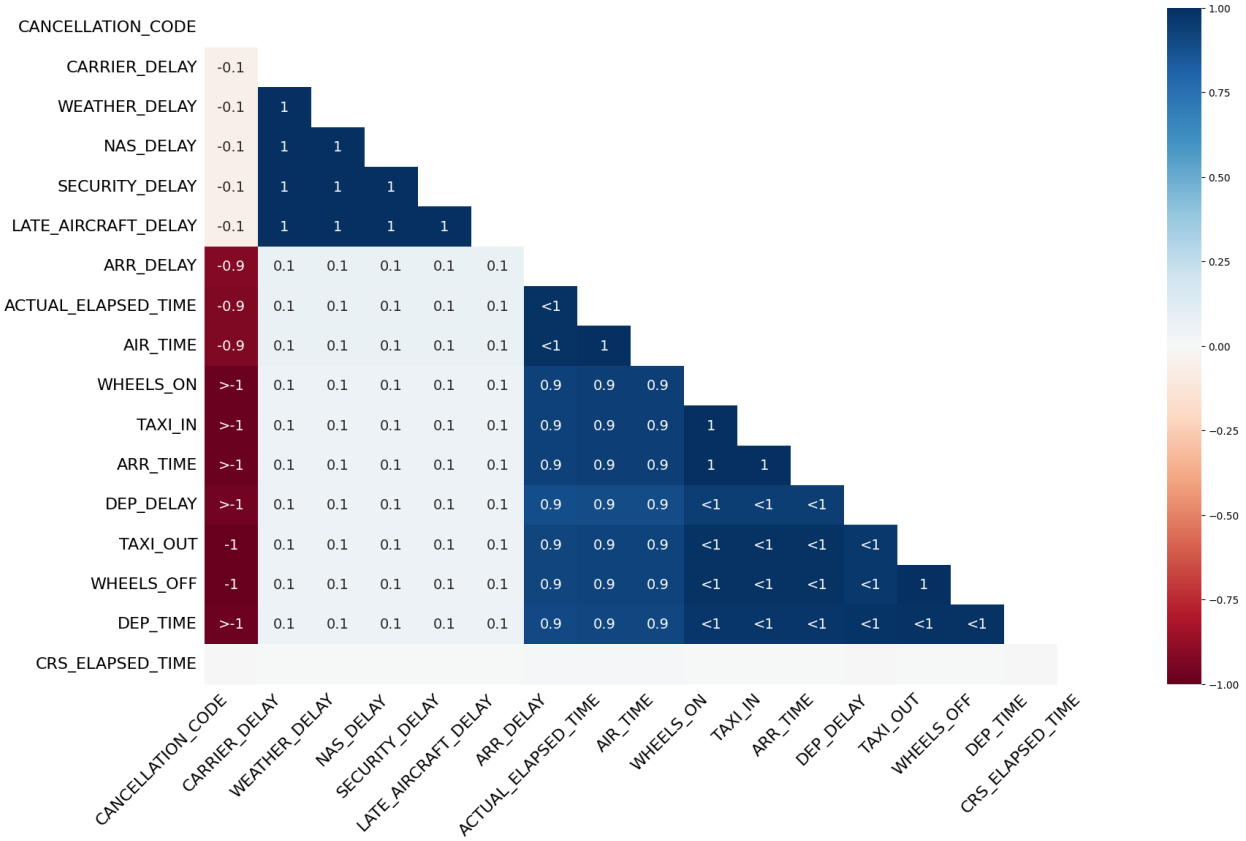
```
================================
Categorical Columns Optimization
================================
These columns can be converted to categorical:

                  unique_values
----------------  ------------------------------------------------------------------------
OP_CARRIER        UA, AS, 9E, B6, EV, F9, G4, HA, MQ, NK, OH, OO, VX, WN, YV, YX, AA, DL
CANCELLATION_CODE  B, A, C, D


========
Outliers
========
Outliers are:

                     outliers_lower    outliers_upper    outliers_total    outliers_
↪percentage
-------------------  ----------------  ----------------  ----------------  ----------
↪-------
DEP_DELAY                      3058            937650            940708              
↪   13.04
ARR_DELAY                      9874            642724            652598              
↪    9.05
DISTANCE                          0            432362            432362              
↪    5.99
TAXI_IN                           0            428981            428981              
↪    5.95
TAXI_OUT                          0            411112            411112              
↪    5.7
CRS_ELAPSED_TIME                  5            395338            395343              
↪    5.48
AIR_TIME                          0            391119            391119              
↪    5.42
ACTUAL_ELAPSED_TIME               0            371247            371247              
↪    5.15
CARRIER_DELAY                     0            155876            155876              
↪    2.16
LATE_AIRCRAFT_DELAY               0            132029            132029              
↪    1.83
CANCELLED                         0            116584            116584              
↪    1.62
NAS_DELAY                         0            100224            100224              
↪    1.39
WEATHER_DELAY                     0             85055             85055              
↪    1.18
DIVERTED                          0             17859             17859              
↪    0.25
SECURITY_DELAY                    0              4348              4348              
↪    0.06
```

(continues on next page)

(continued from previous page)

```
==============
Missing Values
==============
Missing details are:

                     missing_counts    missing_percentage
-------------------  ----------------  --------------------
CANCELLATION_CODE         7.09687e+06                 98.38
CARRIER_DELAY             5.86074e+06                 81.25
WEATHER_DELAY             5.86074e+06                 81.25
NAS_DELAY                 5.86074e+06                 81.25
SECURITY_DELAY            5.86074e+06                 81.25
LATE_AIRCRAFT_DELAY       5.86074e+06                 81.25
ARR_DELAY            137040                            1.9
ACTUAL_ELAPSED_TIME  134442                            1.86
AIR_TIME             134442                            1.86
WHEELS_ON            119246                            1.65
TAXI_IN              119246                            1.65
ARR_TIME             119245                            1.65
DEP_DELAY            117234                            1.63
TAXI_OUT             115830                            1.61
WHEELS_OFF           115829                            1.61
DEP_TIME             112317                            1.56
CRS_ELAPSED_TIME         10                            0
```





The report shows that:

1. 7 duplicated rows.
2. For optimization we can convert 16 columns to *uint16*, 3 columns to *uint8*, 3 columns to *int16*, and 2 columns to *categorical* datatypes.
3. 15 columns have outliers as detailed above.
4. 17 columns have missing values as detailed shown (6 of them have more than 80% of missing values).

3.3 Cleaning

To clean the dataframe (remove missing, unique value columns and duplication):

```
>>> cdf.clean(
        min_missing_ratio=0.05,     # the minimum ratio of missing values to drop a␣
↪column, default is 0.05
        drop_nan=True,              # if True, drop the rows with missing values after␣
↪dropping columns
                                    # with missingsa above min_missing_ratio
        drop_kws={},                # if need to pass any arguments to pd.DataFrame.
↪drop(), default is {}
        drop_duplicates_kws={}      # same drop_kws, but for drop_duplicates function
        )
>>> cdf.report()                    # to see the changes
```

(continues on next page)

(continued from previous page)

```
===============
Duplicated Rows
===============
No duplicated rows.


==============================
Numerical Columns Optimization
==============================
These numarical columns can be down graded:

        columns_to_convert
------  ------------------------------------------------------------------------------------
↪-------------------------------------------------------
uint16  OP_CARRIER_FL_NUM, CRS_DEP_TIME, DEP_TIME, WHEELS_OFF, WHEELS_ON, TAXI_IN, CRS_
↪ARR_TIME, ARR_TIME, ACTUAL_ELAPSED_TIME, AIR_TIME, DISTANCE
int16   DEP_DELAY, ARR_DELAY, CRS_ELAPSED_TIME
uint8   TAXI_OUT, CANCELLED, DIVERTED


================================
Categorical Columns Optimization
================================
These columns can be converted to categorical:

            unique_values
----------  ------------------------------------------------------------------------
OP_CARRIER  UA, AS, 9E, B6, EV, F9, G4, HA, MQ, NK, OH, OO, VX, WN, YV, YX, AA, DL


========
Outliers
========
Outliers are:

                     outliers_lower    outliers_upper    outliers_total    outliers_
↪percentage
-------------------  ----------------  ----------------  ----------------  ----------
↪-------
DEP_DELAY                      3048            931179            934227              ␣
↪   13.21
ARR_DELAY                      9869            642674            652543              ␣
↪    9.23
DISTANCE                          0            427251            427251              ␣
↪    6.04
TAXI_IN                           0            426694            426694              ␣
↪    6.03
TAXI_OUT                          0            408062            408062              ␣
↪    5.77
AIR_TIME                          0            391119            391119              ␣
```

(continues on next page)

(continued from previous page)

```
↪    5.53
CRS_ELAPSED_TIME                    1            390605            390606                ␣
↪    5.52
ACTUAL_ELAPSED_TIME                 0            371246            371246                ␣
↪    5.25


==============
Missing Values
==============
No missing values.
```

3.4 Optimizing

To optimize the dataframe (convert datatypes):

```
>>> cdf.optimize()
>>> cdf.report()                # to see the changes after optimization

===============
Duplicated Rows
===============
No duplicated rows.


==============================
Numerical Columns Optimization
==============================
No numerical columns to optimize.


===============================
Categorical Columns Optimization
===============================
No columns to optimize.


========
Outliers
========
Outliers are:

                     outliers_lower    outliers_upper    outliers_total    outliers_
↪percentage
-------------------  ----------------  ----------------  ----------------  ----------
↪-------
DEP_DELAY                      3048            931179            934227                ␣
↪   13.21
ARR_DELAY                      9869            642674            652543                ␣
↪    9.23
```

(continues on next page)

(continued from previous page)

```
DISTANCE                        0           427251          427251          ␣
↪    6.04
TAXI_IN                         0           426694          426694          ␣
↪    6.03
TAXI_OUT                        0           408062          408062          ␣
↪    5.77
AIR_TIME                        0           391119          391119          ␣
↪    5.53
CRS_ELAPSED_TIME                1           390605          390606          ␣
↪    5.52
ACTUAL_ELAPSED_TIME             0           371246          371246          ␣
↪    5.25


==============
Missing Values
==============
No missing values.
```

All is clear now, only we can see the outliers, the actions required with outliers is out of this module scope.

3.5 How much did we optimize?

Lets see our dataframe info after cleaning and optimizing:

```
>>> df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 7071817 entries, 0 to 7213445
Data columns (total 21 columns):
#   Column               Dtype
---  ------               -----
0   FL_DATE              object
1   OP_CARRIER           category
2   OP_CARRIER_FL_NUM    uint16
3   ORIGIN               object
4   DEST                 object
5   CRS_DEP_TIME         uint16
6   DEP_TIME             uint16
7   DEP_DELAY            int16
8   TAXI_OUT             uint8
9   WHEELS_OFF           uint16
10  WHEELS_ON            uint16
11  TAXI_IN              uint16
12  CRS_ARR_TIME         uint16
13  ARR_TIME             uint16
14  ARR_DELAY            int16
15  CANCELLED            uint8
16  DIVERTED             uint8
17  CRS_ELAPSED_TIME     int16
```

(continues on next page)

(continued from previous page)

```
18  ACTUAL_ELAPSED_TIME  uint16
19  AIR_TIME             uint16
20  DISTANCE             uint16
dtypes: category(1), int16(3), object(3), uint16(11), uint8(3)
memory usage: 431.6+ MB
```

The module reduces the dataframe size from **1.5 GB** to around **430 MB**.

CHAPTER

FOUR

CLEANDATAFRAME DOCSTRING

This class is used to clean and optimize a pandas dataframe for further analysis.

clean_df.CleanDataFrame.**df**

The dataframe to be cleaned and optimized or after cleaning and optimization.

Type
pandas.DataFrame

clean_df.CleanDataFrame.**max_num_cat**

The maximum number of unique values in a column for it to be considered categorical.

Type
int

clean_df.CleanDataFrame.**duplicate_inds**

Array of the indices of duplicated rows.

Type
numpy ndarray, readonly

clean_df.CleanDataFrame.**cols_to_optimize**

A dictionary of all numerical columns that can be memory optimized, it will be {column name: optimized data type}.

Type
dict, readonly

clean_df.CleanDataFrame.**outliers**

A dictionary for outliers details in descinding order in an array as {column name: outlier details} format, the list has:

- The number of lower outliers.
- The number of upper outliers.
- The total number of outliers.
- The percentage of the total values that are outliers.

Type
dict, readonly

clean_df.CleanDataFrame.**missing_cols**

A dictionary for missing details in descinding order in an array as {column name: missing details} format, the list has:

- The total number of missing values.

- The percentage of the total values that are missing.

Type
dict, readonly

clean_df.CleanDataFrame.**cat_cols**

A dictionary for columns that can convert to categiorical type as {column name: array of unique values} format.

Type
dict, readonly

clean_df.CleanDataFrame.**num_cols**

Array of numerical columns.

Type
numpy ndarray of str, readonly

clean_df.CleanDataFrame.**__init__**(*self*, *df*, *max_num_cat=10*) → None:

Constructor for CleanDataFrame.

Parameters

- **df** (*pandas.DataFrame*) – The dataframe to be cleaned and optimized.
- **max_num_cat** (*int, optional*) – The maximum number of unique values in a column for it to be considered categorical, defaults to 10.

clean_df.CleanDataFrame.**report**(*self*, *show_matrix=True*, *show_heat=True*, *matrix_kws={}*, *heat_kws={}*) → None:

Generate a summary report of the dataset, including:

1. Duplicated rows report.
2. Columns' Datatype to optimize memory report.
3. Columns to convert to categorical report.
4. Outliers report.
5. Missing values report.

Parameters

- **show_matrix** (*bool, optional*) – A flag to control whether to show the missing value matrix plot or not, defaults to True.
- **heat_matrix** (*bool, optional*) – A flag to control whether to show the missing value heatmap plot or not, defaults to True.
- **matrix_kws** (*dict, optional*) – Keyword arguments passed to the missing value matrix plot, defaults to {}.
- **heat_kws** (*dict, optional*) – Keyword arguments passed to the missing value heatmap plot, defaults to {}.

Raises
TypeError – If any parameter has the wrong type.

clean_df.CleanDataFrame.**clean**(*self*, *min_missing_ratio=0.05*, *drop_nan=True*, *drop_kws={}*, *drop_duplicates_kws={}*) → None:

Drops columns with a high ratio of missing values and duplicate rows.

Parameters

- **min_missing_ratio** (*float, optional*) – The minimum ratio of missing values for columns to drop. Value should be between 0 and 1. Defaults to 0.05.
- **drop_nan** (*bool, optional*) – A flag to decide whether to drop any rows that contain missing values after dropping the columns with above *min_missing_ratio* missing values. Defaults to True.
- **drop_kws** (*dict, optional*) – Keyword arguments passed to the *drop()* function. Defaults to {}.
- **drop_duplicates_kws** (*dict, optional*) – Keyword arguments passed to the *drop_duplicates()* function. Defaults to {}.

Raises

- **TypeError** – If *drop_nan* is not boolean, or if *drop_kws* or *drop_duplicates_kws* have the wrong types.
- **ValueError** – If *min_missing_ratio* is not between 0 and 1, or if *drop_kws* or *drop_duplicates_kws* have a key *inplace*.

clean_df.CleanDataFrame.**optimize**(*self*) → None:

Optimizes the dataframe by converting columns to the desired data type and converting categorical columns to 'category' data type. Note that numerical columns should not contain missing values.

Raises

Warning – If any numerical column contains missing values.

CHAPTER

FIVE

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/naelaqel/clean_df/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

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

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/clean_df.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 clean_df
$ cd clean_df/
$ 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 clean_df tests
$ python setup.py test or pytest
$ 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.

5.3 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 Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/naelaqel/clean_df/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_clean_df
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Nael Aqel <dev@naelaqel.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER

SEVEN

HISTORY

7.1 0.3.0 (2023-08-23)

- Improve the performance when calling `report` method.
- The `pytest` now is including the full methods in the module.

7.2 0.2.3 (2023-03-04)

- Improve memory consumption and module performance.

7.3 0.2.2 (2023-03-03)

- Fix a bug that made “dict_keys” error in some speical cases.

7.4 0.2.1 (2023-03-03)

- Improve module performance.

7.5 0.2.0 (2023-03-02)

- Add a new report for categorical columns.
- Make the module more efficient.

7.6 0.1.1 (2023-02-27)

- Rectify and organize documentation.
- Provide test to GitHub Actions.

7.7 0.1.0 (2023-02-27)

- First release on PyPI.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

C

INDEX

Symbols

C

D

M

N

O

R