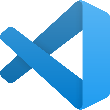
**Data Cleaning**

06/06/2020 Created by Apiwat Chuaphan

11/08/2020 Revised by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

 A close up of a screen

Description automatically generated A close up of a sign

Description automatically generated

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps might not be explained in the tutorial.  If you are not sure what to do:
  + Consult the resources listed below.
  + If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

* Learn how to use Pandas library to clean up unwanted and useless data.
  + Missing data
  + Duplicate data
* Learn how to use Pandas library to manipulate string.

**Resources**

* Pandas Documentation: <https://pandas.pydata.org/docs/user_guide/index.html>
* Data from Kaggle: <https://www.kaggle.com/datasets>

**Data Cleaning**

In real world, data comes from several sources before final data to be presented. The data from each source might not be complete. A data scientist needs to do data cleanup and manipulation that being said about 80 percent of any project is spent just getting, cleaning, and organizing the data before moving into a form ready for analysis.

**Dealing with missing data**

Open Jupyter Notebook:

1. Create a new file named **data\_cleaning.ipynb** under Module folder
2. Create data that contain None value by typing the following:

**A picture containing clock, drawing

Description automatically generated**

1. **Numpy** and **Pandas** data that contains
   1. **None** value will generate an error when you perform aggregations like sum() or min() on arrays.
   2. **NaN**, (missing floating-point values), no error pops up but the result of the operation is not useful.

A screen shot of a social media post

Description automatically generated

We will detect and manipulate null values by using the following methods from Pandas’s Series and DataFrame.

* isnull(): Generates a Boolean mask indicating missing valuesA picture containing clock

  Description automatically generated

As you see here, empty string is not treated as null in Pandas

* notnull(): Opposite of isnull()A picture containing drawing

  Description automatically generated

**ex1.notnull()** returns the opposite Boolean from **isnull()**

We displayed not-null values in the array ex1 with the filter of **notnull()**

* dropna(): Returns a filtered version of the dataA picture containing clock, meter

  Description automatically generated

This result looks similar to **notnull()** with indexing on the masked values from ex1 array, but **dropna()** removed missing values directly from ex1 array.

A screen shot of a smart phone

Description automatically generated

Notice that pandas actually dropped entire row that contain null values and you cannot drop a single value from DataFrame. The options you have is to drop either row or column. (Default is row).

You can specify the parameter as **ex3.dropna(axis=1)** for columns drop.

* fillna(): Returns a copy of the data with missing values filled or imputed

it can sometimes make more sense to fill null values with valid ones rather than drop them.

A close up of a screen

Description automatically generated

**Removing duplicate data**

1. We can use **duplicated** to spot duplicate values. The method returns Boolean mask saying whether value in DataFrame is duplicate. Let’s see how it does by typing the following:

A screenshot of a cell phone

Description automatically generated

As you can see here row 2 and row 4 are duplicate values of (A, 1) and (B, 3)

1. We can drop the duplicate values from DataFrame like the following. This will check values from all columns if all has the same values, drop it out.

A screenshot of a cell phone

Description automatically generated

1. What if we only want to drop duplicate values in particular column, not all. We can do that by specifying column like below.

A picture containing clock, monitor, screen

Description automatically generated

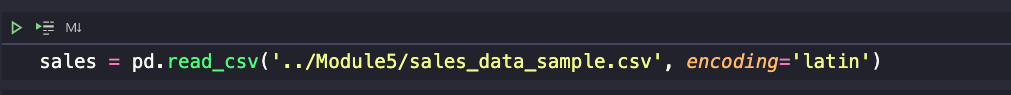
Notice the differences that only specified column was checked for duplicate values.

If you would like to check for yourself or see what other parameters the method has, run **ex3.dropna?,** for example**,**  in a code cell. This would do a similar job with **man** command in Linux.

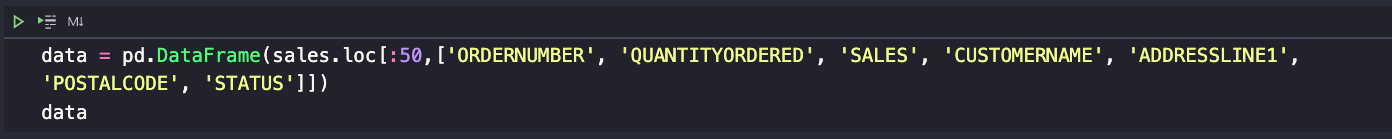
**String Manipulation**

1. We will use the CSV file from the previous module by specify the correct location of the CSV like below.

*Note: The path in the screenshot might be different from yours*



1. We will then select some columns that we need more than others.



We selected 7 columns out of 25 and 50 rows of out 2823 rows.

1. Changing the name of columns by the following method. This changes ORDERNUMBER to ORDER\_ID and so on as well as **inplace** option to make it change in-place.

A close up of a screen

Description automatically generated

1. Selecting all columns but not the one we do not need.

A screenshot of a cell phone

Description automatically generated

If you want to see the values from these columns, wrap it with **data[].**

**data[data.columns[data.columns != 'STATUS']]**

1. Let’s see how many sales are more than 5000 by using Boolean indexing.

A screenshot of a cell phone

Description automatically generated

1. We can also look for values that contain particular string in it.

A screenshot of a cell phone

Description automatically generated

1. Replacing values based on conditions. In this case STATUS column’s values changed depends on the values in SALES, if it is more than 5,000, STATUS will be ‘Canceled’

A screenshot of a cell phone

Description automatically generated

Learn more of what you can do with Pandas here <https://pandas.pydata.org/pandas-docs/stable/reference/index.html>