Data Cleaning - An introductory Overview

Introduction

How can we determine if a dataset is ready to be used? This article is intended to provide an introductory overview of basic techniques to clean your dataset. To note, data cleaning standards differ depending on the industry; however, I hope to provide some basic steps that are important to consider regardless of the field at hand. Before moving on, we will be heavily utilizing the Pandas module provided in Python although there are many other alternative methods and modules available.

**What is data cleaning?**

Data cleaning refers to the practice of removing erroneous data, which includes NA/NaN values, duplicates, etc. Erroneous data is mostly caused by human error such as bad sampling methods, misinputs, etc. Data cleaning is important the quality of your data set directly influences the validity of your conclusions.

The Data Cleaning Process

For demonstrative purposes I will showcase procedures of data cleaning using a dataset from my machine learning project provided by my internship at Data Glacier.

**Duplicate Data:**

As the name suggests, duplicate data is data that has been accounted for more than once. We usually do not want this. Below showcases how one would remove duplicated data from their data set.



In this code “df” refers to the data frame name that you wish to manipulate and “drop\_duplicates()” is the method name. I have left all the parameters as its default and the result is a data frame that keeps only the FIRST instances of duplicated data (dropping other instances if it exists). If you would like to delete ALL instances of duplicated data, simply change “keep = ‘last’”. 4

**Missing Values:**

Before we start, we should note that there is a difference between NA and NaN. NA stands for “not applicable” while NaN stands for “not a number”. NaN should be used carefully as several data frames are filled with a mixture of quantitative and qualitative data. Therefore, be careful if you are using NaN. Nevertheless, how do we detect and account for such values?  
  


In this code, the method “isnull()” applied to dataframe “df” is checking for missing values in each row and column by adding “.all()”. The result is a dictionary containing column names as keys and Booleans as values. Each key value pair indicates if there is a missing value in the respective column (in which case it will print “True”) or not (in which case it will print “False”). Couple this method with “sum()” to find the total missing values in every column and all of the data frame. My dataset does not have any missing values, but it does possess missing information.5

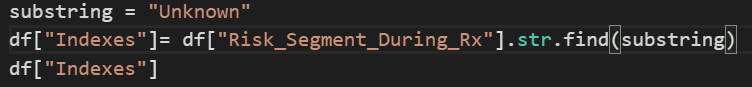


Here is a small snippet of my code for illustration:

Table

Description automatically generated

So, while I do not have to worry about NA nor NaN values, I do have to worry about missing information. In this case, I will use a method to detect string matches of “Unknown”. Here is the code I utilized:



I created a new column in my data frame names “Indexes” that contains every instance of the substring “Unknown” found in the “Risk\_Segment\_During\_Rx” column by returning a “0”. It is to be noted that this method is case sensitive which means that if by human error, “unknown” was inputted instead of “Unknown”, this method would not be able to detect it. Grammatical errors are worth considering but will not be covered in this article.7

**Wrong Date Formats:**

Luckily, there are no dates to consider in my data frame; however, this is not often the case. Dates are complicated because there are multiple ways to represent them (either by preference, accident, or cultural preference). There is also an issue with how some applications, like Excel, will “guess” dates that may or may not be correct. I will not cover this issue here as it can be quite extensive; however, here is a source that I believe will be immensely beneficial to look over: <https://datagy.io/pandas-datetime/>.

**Skewness:**

For skewness I recommend using the below method as well as visualizing your data. In this case, I applied the “skew()” method on my dataframe “df” over the index axis “0”. If you wish to check for every row, simply change the parameter “axis = 1”. The result produces a float data type indicating the measure of skewness. For general guidelines, a zero indicates perfect symmetry, a negative skew means the tail extends to the left side (negative values), and a positive skew means the tail extends to the right side (positive values).



Note that this method only considers columns containing quantitative data in the appropriate index. Categorical data analysis is a whole other subject that deserves its own article. Also, some quantitative data are measured in ranges rather than specific numbers. This method would skip such occurrences.6  
As I mentioned previously, it is very useful to visualize your data. I recommend importing “plotly.express as px” and using the “px.histogram()” method. Here is what a result would look like:

Chart, Teams

Description automatically generatedMatplotlib is also an excellent alternative.  
As an extension, you can measure how heavy-tailed or light-tailed a distribution is compared to the normal distribution by performing kurtosis. This article does not cover this, but I recommend looking into it.10

**Outliers:**

As previously mentioned, I would heavily recommend visualizing outliers in your data (in this case using box plots). I used the “px.box()” method included in the module earlier:



Here is what a result would look like that I applied to one column in my data frame “Count\_Of\_Risks”:

Chart, timeline

Description automatically generated

This method is useful in identifying outliers (in small instances) as I can hover my cursor over the dots (which represents outliers) and find its values. In cases where there are many outliers, this is not the most efficient method in obtaining outlier values. So, what do we do then?

Text

Description automatically generated

Above is a function created by Eric Kleppen that helps count and identify outliers.  


I set the function “find\_Outliers()” to my data frame column “df[‘Dexa\_Freq\_During\_Rx’]”. The result should return the value of the detected outlier and its position within the column. You can also use methods such as “str(len(outliers))” to find the total number of outliers. I heavily recommend checking out his article in “Bibliography” for more information.3  
An extension would be to find multivariate outliers. This is much more complicated and is not covered here but useful to further research. Here is a link to an article I found helpful: <https://towardsdatascience.com/detecting-and-treating-outliers-in-python-part-2-3a3319ec2c33>.

**An Example of a clean data set:**

The Fisher’s iris flower dataset is a popular dataset for beginner machine learning projects for a good reason. The dataset is a 150 data point set organized into five columns: sepal length, sepal width, petal length, petal width, and species. You can look at it here or its Wikipedia page linked in “Bibliography”: <https://www.randomservices.org/random/data/Fisher.html>.2

Conclusions

I hope that this article has helped provide an introductory overview of data cleaning. Usually, after this step, data will be cleaned and transformed. I recommend studying clean data sets and then cleaning messy data sets (Kaggle is a great source) for practice. Please note that this article is not meant to provide a thorough data cleaning process but rather a starting point for beginners. For more specific information about the steps and code mentioned above please refer to the bibliography below for further clarification.

Bibliography

1“How to Clean Your Data.” The Official Portal for European Data. European Commission, n.d. https://data.europa.eu/elearning/en/module11/#/id/co-01.

2“Iris Flower Data Set.” Wikipedia. Wikimedia Foundation, December 15, 2022. https://en.wikipedia.org/wiki/Iris\_flower\_data\_set#Data\_set.

3Kleppen, Eric. “How to Find Outliers Using Python [Step-by-Step Guide].” CareerFoundry, February 24, 2022. https://careerfoundry.com/en/blog/data-analytics/how-to-find-outliers/.

4“Pandas Drop Duplicate Rows in DataFrame.” Spark by {Examples}, December 9, 2022. https://sparkbyexamples.com/pandas/pandas-drop-duplicate-rows-from-dataframe/#:~:text=Remove%20All%20Duplicate%20Rows%20from,drop\_duplicates(keep%3DFalse)%20.

5“Pandas: Detect and Count Missing Values (Nan) with Isnull(), ISNA().” pandas: Detect and count missing values (NaN) with isnull(), isna(), n.d. https://note.nkmk.me/en/python-pandas-nan-judge-count/#:~:text=By%20calling%20all()%20from,it%20is%20applied%20to%20rows.

6“Python: Pandas Dataframe.skew().” GeeksforGeeks, July 15, 2022. https://www.geeksforgeeks.org/python-pandas-dataframe-skew/.

7“Python: Pandas Series.str.find().” GeeksforGeeks, June 23, 2021. https://www.geeksforgeeks.org/python-pandas-series-str-find/.

8Sherrer, Kara. “Data Cleaning: Definition, Methods & Steps.” TechnologyAdvice, June 30, 2022. https://technologyadvice.com/blog/information-technology/data-cleaning/.

9Tamboli, Nasima. “Tackling Missing Value in Dataset.” Analytics Vidhya, November 30, 2022. https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/.

10Zach. “How to Calculate Skewness & Kurtosis in Python.” Statology, October 2, 2020. https://www.statology.org/skewness-kurtosis-python/#:~:text=This%20value%20can%20be%20positive,extends%20towards%20more%20positive%20values.