

The challenge of missing or duplicate data

- Video: Welcome to week 33 min
- Video: Methods for handling missing data8 min
- Reading: Data deduplication with Python20 min
- Reading: Follow-along instructions: Work with missing data in a Python notebook10 min
- Lab: Annotated follow-along guide: Work with missing data in a Python notebook20 min
- Video: Work with missing data in a Python notebook12 min
- Video: Remy: A day in the life of a data professional2 min
- Lab: Activity: Address missing data1h
- Lab: Exemplar: Address missing data20 min
- Practice Quiz: Test your knowledge: The challenge of missing or duplicate data4 questions

The ins and outs of data outliers

- Change categorical data to numerical data
- Input validation
- Review: Clean your data

Data deduplication with Python

As you've learned, the data cleaning and validating practices include several different steps, including handling missing data, outliers, and label encoding; checking for misspellings; and, handling duplicates. As a data professional, it will be your task to know how best to handle data values in those categories. In this reading, you'll learn more about handling duplicates. You will also learn to identify and decide whether deduplication is the right strategy for a dataset. In addition, you will learn some common Python functions for handling duplicates.

Identifying duplicates

Before we make any decisions on whether to remove duplicate values or not, we should first determine if duplicate values are present in our dataset.

A simple way to identify duplicates is to use the `pd.duplicated()` function from Pandas.

This function returns a series of "true/false" outputs, with "true" indicating the data value is a duplicate, and "false" indicating it is a unique value.

Here's an example of a five-row data frame:

Note: The following code block is not interactive.

```
1 df
2   brand   style  rating
3 0 Wouyow cistern  4.0
4 1 Wouyow cistern  4.0
5 2 Splaysh jug     5.5
6 3 Splaysh stock  3.3
7 4 Pipplee stock  3.0
```

Using the "duplicated" function, the result is that one has been marked "True," indicating it is a duplicate.

Note: The following code block is not interactive.

```
1 df
2   brand   style  rating
3 0 Wouyow cistern  4.0
4 1 Wouyow cistern  4.0
5 2 Splaysh jug     5.5
6 3 Splaysh stock  3.3
7 4 Pipplee stock  3.0
8 df.duplicated()
9 0 False
10 1 True
11 2 False
12 3 False
13 4 False
14
```

Identifying duplicates for an entire data frame will be different than a single column or index. Be sure when you use the `duplicated()` function for an entire dataframe. The `duplicated()` function will only return *entire rows* that have exactly matching values, not just individual matching values found within a column. If you wish to identify duplicates for only one column or a series of columns within a dataframe, you will need to include that in the "subset" portion of the argument field of the `duplicated()` function. Going further, if you'd like to specify which of the duplicates to keep as the "original" as opposed to the duplicate, you can specify that in the "keep" portion of the argument field.

Below is an example of identifying duplicates in only one column (subset) of values and labeling the last duplicates as "false," so that they are "kept":

```
1 df = pd.DataFrame({
2     'type': ['rinds', 'rinds', 'pellets', 'pellets', 'pellets'],
3     'color': ['olive', 'olive', 'gray', 'salmon', 'salmon'],
4     'rating': [9, 9, 4.5, 11, 7]
5 })
6 df
7
8 df.duplicated(subset=['type'])
9
10 df.duplicated(keep='last')
```

Decision time: To drop or not to drop?

As you've learned, every dataset is unique and you cannot treat every dataset the same. When you are making the decision on whether to eliminate duplicate values or not, think deeply about the **dataset itself** and about **the objective you wish to achieve**. What impact will dropping duplicates have on your dataset and your objective?

1. Deciding to drop

You should **drop or eliminate** duplicate values if duplicate values are clearly mistakes or will misrepresent the remaining unique values in the dataset.

House address	Latest price
567432 Pickled Puppeteer Pike	275,300
10009 AI B Kerxy St.	199,999
984 Fortitudnal Fort Rd	298,342
6743 Believed Blvd	245,654
14573 S Match Rd	203,778
32 South Uhdvabor Dur	299,444
27000 N Umberland St	270,008
14573 S Match Rd	203,778

For example, you can be reasonably sure that a data professional will (in most cases) eliminate duplicate values of a dataset containing house addresses and house prices. Counting the same house twice will (in most cases) misrepresent any conclusions drawn from the dataset as a whole, such as average house price, total house price, or even total number of houses. In a case like this, a data professional would almost certainly eliminate the duplicate data so as to fairly represent the remaining data during analysis and visualization.

2. Deciding to NOT drop

You should **keep** duplicated data in your dataset if the duplicate values are clearly **not** mistakes and should be taken into account when representing the dataset as a whole.

Throw / Attempt No.	Distance (m)
1	12.3
2	13.6
3	12.3
4	12.7
5	12.9
6	13.4
7	12.9
8	13.2

For example, a dataset marking the number of throws and distances of an Olympic shot-put athlete in training will likely include several duplicate distances; just by nature of number of attempts and the limits a person can have a weighted ball, there will be duplicate values—particularly if the distance measurements are labeled to only 1 or 2 decimal places. In a case like this, a data professional would almost certainly keep all of the data to fairly represent it as a whole during analysis and visualization.

Don't be duped — How to do deduplication

Before we get back into Python and learn how to eliminate duplicates, let's first define the term "deduplication":

- Deduplication:** The elimination or removal of matching data values in a dataset.

There are a number of different libraries, functions, and methods in Python you could use to remove matching data values.

One of the more common functions to use is in Pandas: `drop_duplicates()`

This function will help you create a new dataframe with all of the duplicate rows removed.

For example, use a data frame from earlier in this reading:

Note: The following code block is not interactive.

```
1 df
2   brand   style  rating
3 0 Wouyow cistern  4.0
4 1 Wouyow cistern  4.0
5 2 Splaysh jug     5.5
6 3 Splaysh stock  3.3
7 4 Pipplee stock  3.0
```

Now apply the drop duplicates function:

Note: The following code block is not interactive.

```
1 df.drop_duplicates()
2
3   brand   style  rating
4 0 Wouyow cistern  4.0
5 1 Splaysh jug     5.5
6 2 Splaysh stock  3.3
7 3 Pipplee stock  3.0
8
```

You'll notice in the resulting output that the duplicate row of data was removed, leaving the remaining unique values intact.

Note: Keep in mind that the `drop_duplicates()` function as written above will only drop duplicates of exact matches of **entire rows of data**. If you wish to drop duplicates within a single column, you will need to adjust the code accordingly.

Key Takeaways

Identifying duplicate data values in a dataset is an important part of EDA (or "Exploratory Data Analysis") practices, specifically cleaning and validating. After identifying duplicates, think about the impact to the dataset and your analysis objective when choosing to eliminate duplicates or not eliminate duplicates.

Additional Resources

Want to learn more about duplicates and deduplication? Check out the following additional links.

- Look at Pandas documentation to learn more about the parameters of the argument field
- Advanced deduplication walkthrough – partial vs exact duplicate handling
- Dedupe library: perform identity resolution and deduplication using machine learning

Mark as completed