datacamp.com

Pandas Profiling (ydata-profiling) in Python: A Guide for Beginners

Satyam Tripathi

12–15 minutes

When dealing with a dataset, such as one with 10,000 rows and 50 columns, gaining a quick overview of these datasets quickly can be challenging. This is where pandas Profiling comes in handy. It streamlines the process by generating a comprehensive report of your dataset, minimizing the time to explore these large datasets.

In this article, you will learn how to get started with what was formerly known as pandas Profiling. **The `pandas-profiling` package name was recently changed to** `ydata-profiling`. In this tutorial, you will learn about generating a profile report from the dataset, what is inside the profile report, how to read this profile report, and finally, how to save this report for further use.

What is pandas Profiling (ydata-profiling)?

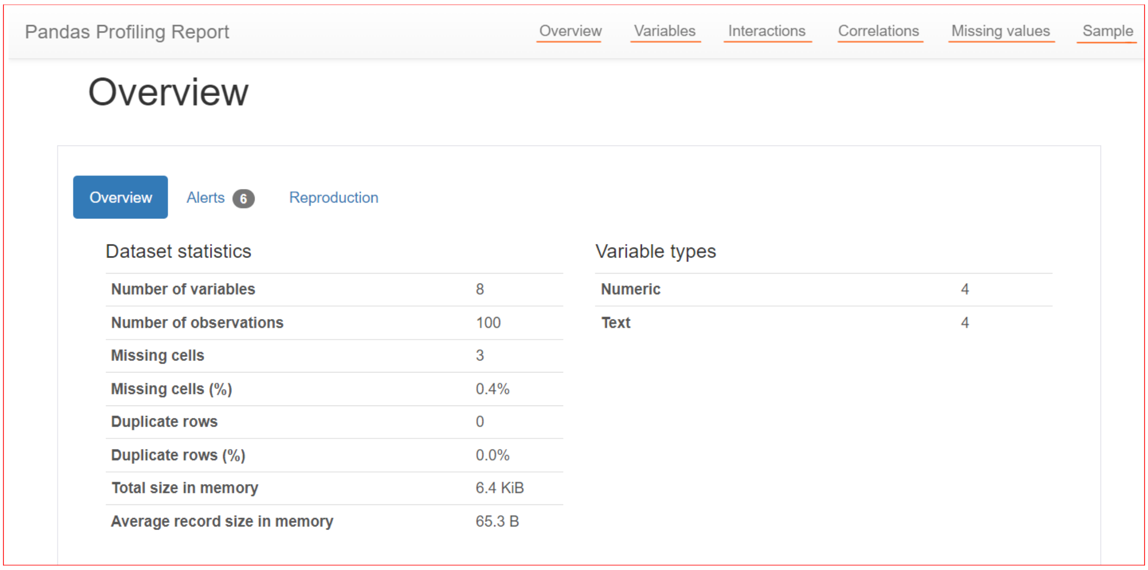
Pandas Profiling is used to generate a complete and exhaustive report for the dataset, with many features and customizations in the generated report. This report includes various pieces of information such as dataset statistics, distribution of values, missing values, memory usage, etc., which are very useful for exploring and analyzing data efficiently.

Pandas Profiling also helps a lot in Exploratory Data Analysis (EDA). EDA is used to understand the underlying structure of data, detect patterns, and generate insights in a visual format.

For EDA, we have to write many lines of code, which can sometimes be complex and time-consuming, but it can be automated using Pandas Profiling with just a few lines of code.

If you need a refresher on EDA, read out [Python Exploratory Data Analysis](#).

Here’s an example of a profile report:



Overview

Dataset statistics

Number of variables	8
Number of observations	100
Missing cells	3
Missing cells (%)	0.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	6.4 KiB
Average record size in memory	65.3 B

Variable types

Numeric	4
Text	4

Image by Author

Advantages and Disadvantages of pandas Profiling

Pandas profiling is widely used in EDA due to its ease of use, time efficiency, and interactive HTML reports. However, there are some potential drawbacks to using pandas profiling with large datasets.

Advantages

- **Ease of use**: Pandas profiling is very easy to use. You only need to write a couple of lines of code to generate a comprehensive report.
- **Time-saving**: Pandas profiling can create a comprehensive report with a wide range of information about a dataset with minimal effort. This makes it a great option for EDA.
- **Interactive HTML reports**: Pandas profiling generates interactive HTML reports that are easy to analyze and understand. The reports also allow you to dig deeper into specific variables and explore their distributions.

Disadvantages

- **Performance with large datasets**: The primary drawback of pandas profiling lies in its performance with large datasets. As the data volume grows, the report generation time increases considerably, making it less efficient for large-scale data analysis.

Installation of pandas Profiling

To install pandas Profiling, you can use either pip or conda, depending on your preference and environment.

Using Pip:

Open a command prompt or terminal and run the following command:

```
pip install ydata-profiling
```

Was this helpful?

Using Conda:

Open the Anaconda PowerShell Prompt and run the following command:

```
conda install -c conda-forge ydata-profiling
```

Was this helpful?

Importing Pandas Profiling

After the installation is successfully completed, import `ydata-profiling` using the following statement.

```
from ydata_profiling import ProfileReport
```

Was this helpful?

This will import the `ProfileReport` class from the `ydata_profiling` library. You can use this class to generate profile reports for your DataFrames.

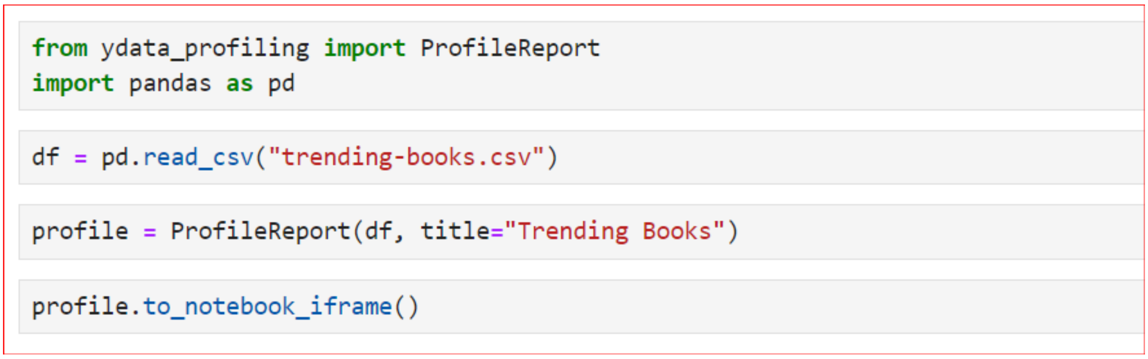
Generating Profile Report

To generate a profile report, follow the steps below:

1. Import **pandas**.
2. Import the **ProfileReport** class from the `ydata_profiling` library.
3. Create a DataFrame using your data.
4. Use the `ProfileReport()` class and pass the DataFrame.

Here's the straightforward code following the steps outlined above. First, we import the necessary libraries and then read the CSV file using the `read_csv()` function. In this case, we're using the CSV file of the Top 100 Bestselling Book Reviews. Next, we use the `ProfileReport` class and pass our DataFrame inside it.

Additionally, we're setting a new title, "Trending Books". By default, the title is something else, but if you want to customize it, use the `title` variable within the class. Finally, to generate and display the report, you can either use `profile` or `profile.to_notebook_iframe()`.



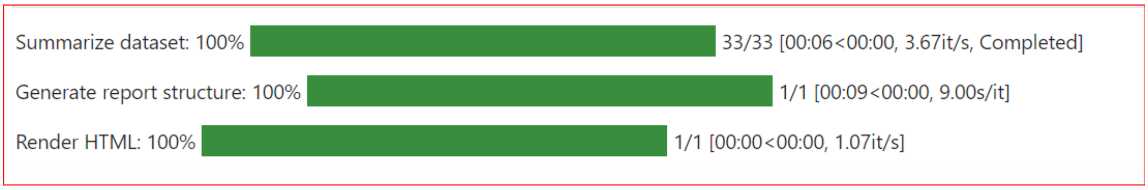
```
from ydata_profiling import ProfileReport
import pandas as pd
```

```
df = pd.read_csv("trending-books.csv")
```

```
profile = ProfileReport(df, title="Trending Books")
```

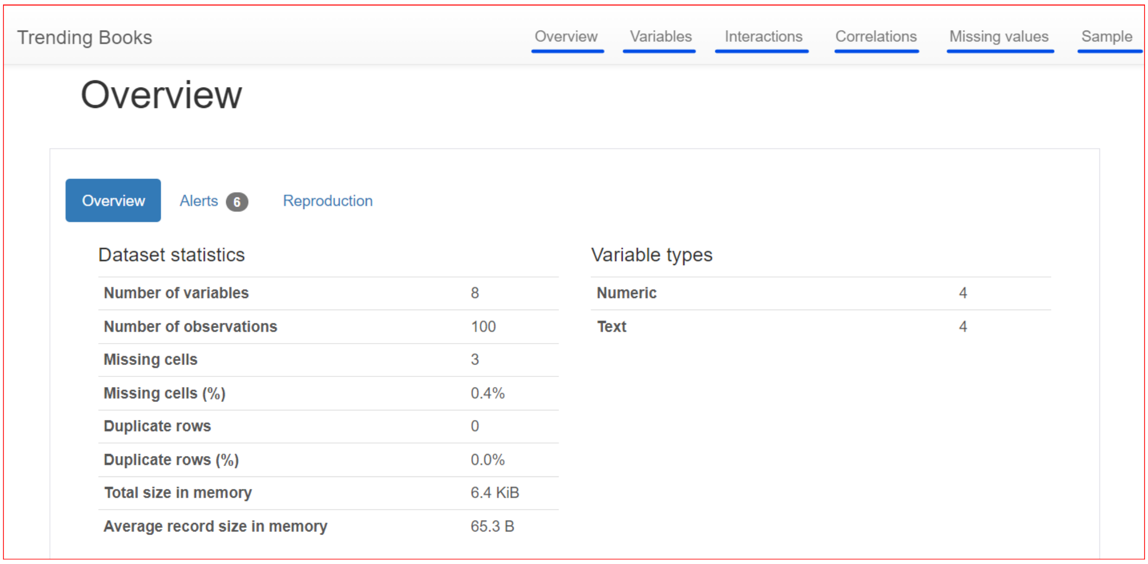
```
profile.to_notebook_iframe()
```

The report will be generated in the following sequence: First, the entire dataset will be summarized. Then, the report structure will be generated. Finally, it will display the report, which you can save as an HTML file and use for further analysis.



```
Summarize dataset: 100% 33/33 [00:06<00:00, 3.67it/s, Completed]
Generate report structure: 100% 1/1 [00:09<00:00, 9.00s/it]
Render HTML: 100% 1/1 [00:00<00:00, 1.07it/s]
```

Here is the generated report, which includes different sections such as **Overview**, **Variables**, **Interactions**, **Correlations**, **Missing Values**, and **Sample**.



Overview

Dataset statistics	
Number of variables	8
Number of observations	100
Missing cells	3
Missing cells (%)	0.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	6.4 KiB
Average record size in memory	65.3 B

Variable types	
Numeric	4
Text	4

If you are new to EDA and more specifically data profiling, read *out Exploratory Data Analysis of Craft Beers: Data Profiling.*

Exploring Profile Report Generated

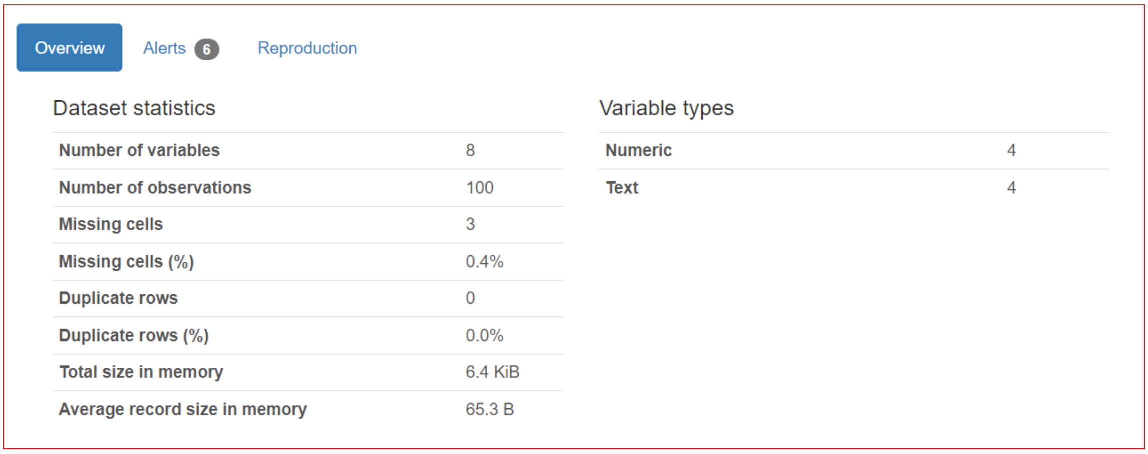
The report is generated in many sections, let’s explore all the sections one by one.

Overview

This section consists of 3 tabs: **Overview**, **Alerts**, and **Reproduction**.

The Overview tab includes dataset statistics, such as the number of variables (or the number of different columns), the number of cells that have missing values, duplicate rows, and the size of the dataset in memory.

In our dataset, there are a total of 8 variables or columns. Among the variables, four are numeric (rank, book price, rating, and year of publication), while the remaining four are text-based (book title, author, genre, and URL). There are no duplicate rows, as shown by a count of 0 for duplicates. Also, the 'rating' column has three missing values.



Dataset statistics

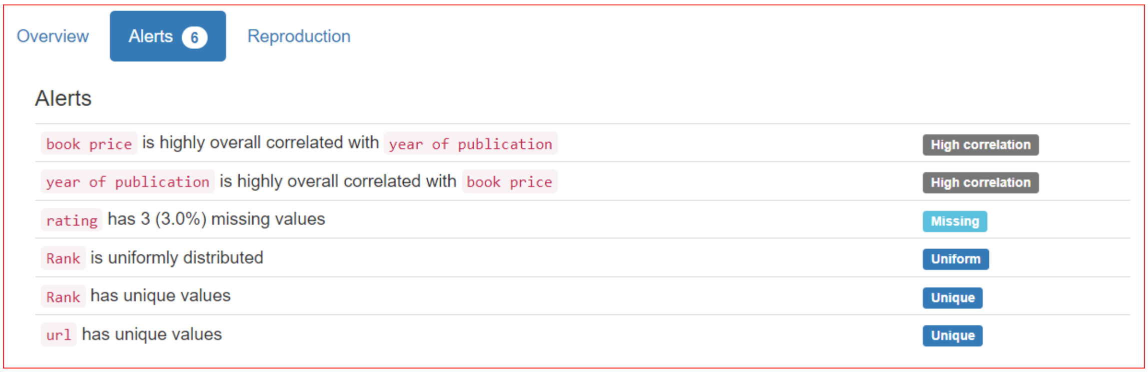
Number of variables	8
Number of observations	100
Missing cells	3
Missing cells (%)	0.4%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	6.4 KiB
Average record size in memory	65.3 B

Variable types

Numeric	4
Text	4

The **Alerts** tab consists of alerts related to correlations with other variables, missing values, unique values, zeroes, etc.

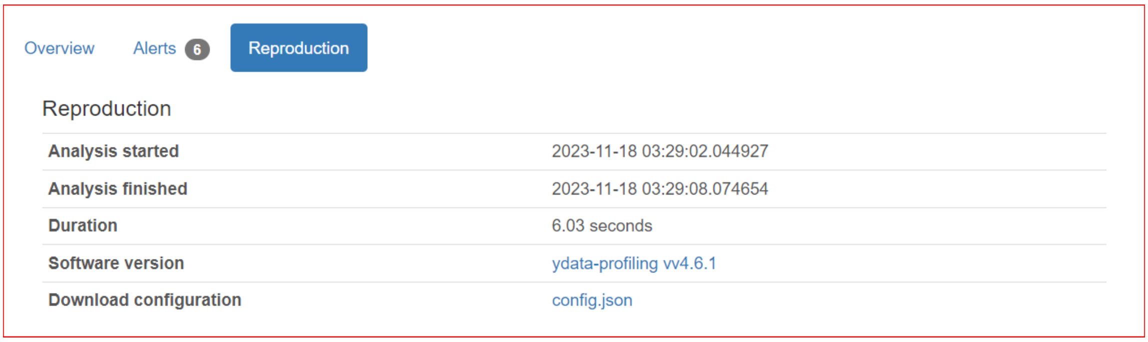
In our case, the URL and Rank columns have unique values, and the rating column has three missing values.



Alerts

`book price` is highly overall correlated with `year of publication`	High correlation
`year of publication` is highly overall correlated with `book price`	High correlation
`rating` has 3 (3.0%) missing values	Missing
`Rank` is uniformly distributed	Uniform
`Rank` has unique values	Unique
`url` has unique values	Unique

The **Reproduction** tab shows when the analysis started and when it ended. It displays the duration of the analysis, including the software version you are using (in my case, it's **ydata-profiling v4.6.1**).

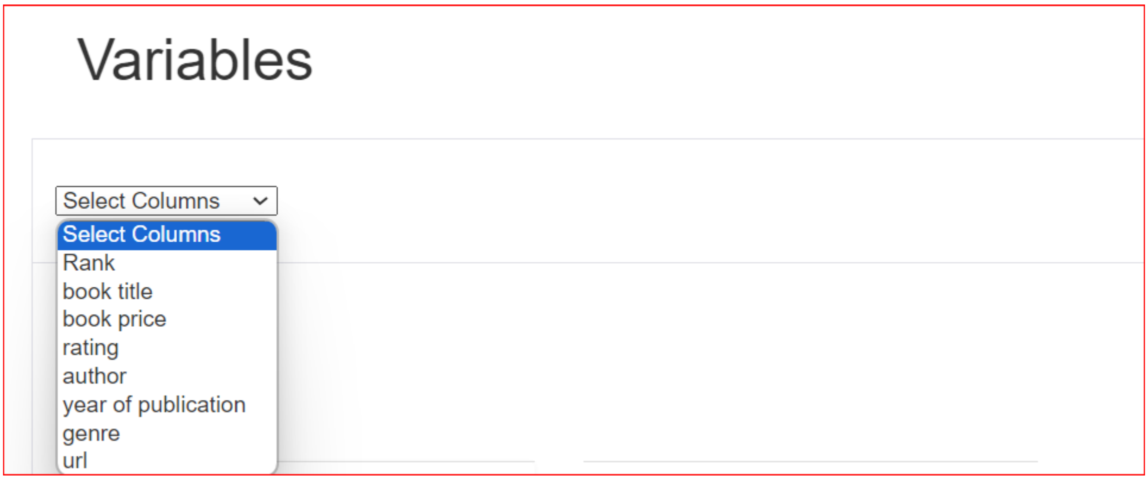


Reproduction

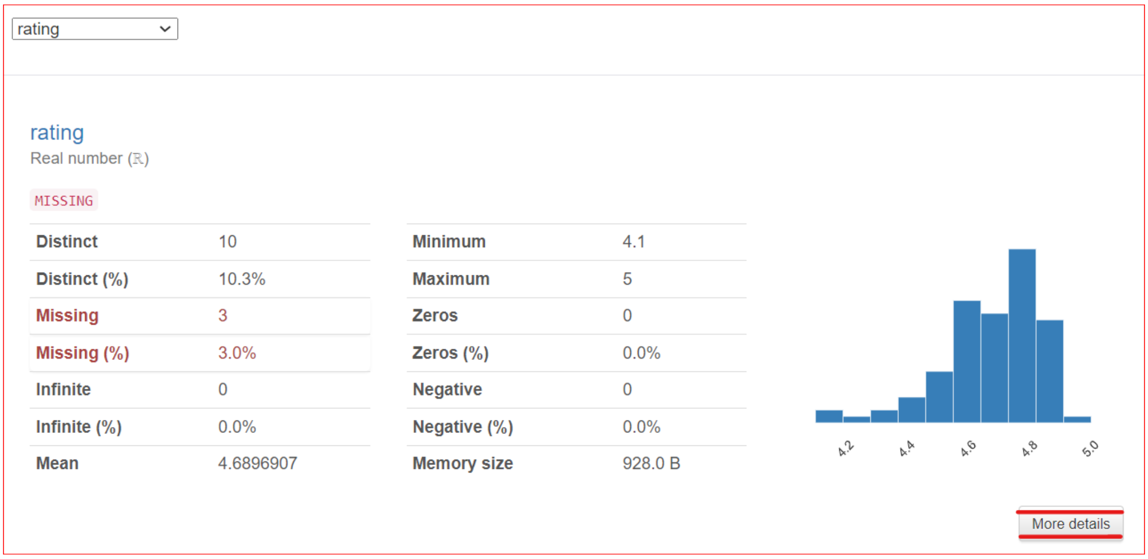
Analysis started	2023-11-18 03:29:02.044927
Analysis finished	2023-11-18 03:29:08.074654
Duration	6.03 seconds
Software version	ydata-profiling vv4.6.1
Download configuration	config.json

Variables

The **Variables** section includes all the columns of your dataset. You can click on the toggle arrow and select any column.



Assuming you have selected the **rating** column, the report shows that this column contains 10 unique values that are distributed across 100 rows. Additionally, three cells lack any value. The minimum value is 4.1, while the maximum is 5. The mean of all ratings is also displayed.



rating
Real number (ℝ)

MISSING

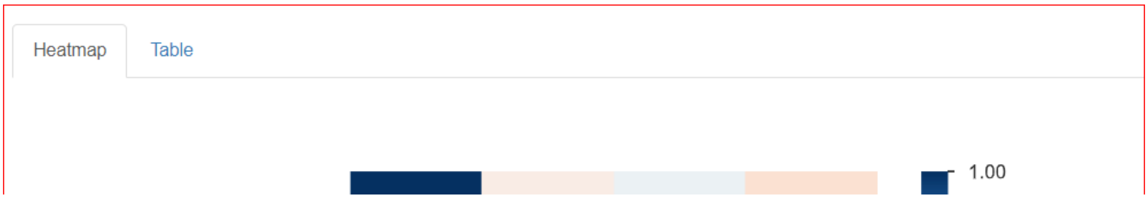
Distinct	10	Minimum	4.1
Distinct (%)	10.3%	Maximum	5
Missing	3	Zeros	0
Missing (%)	3.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	4.6896907	Memory size	928.0 B

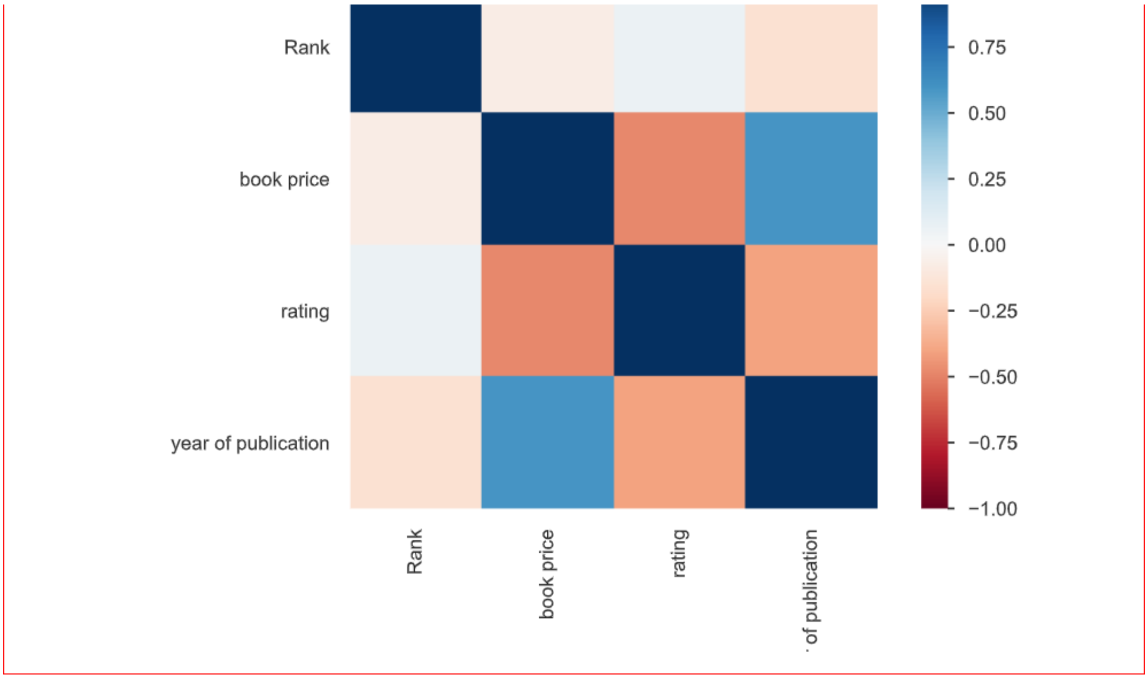
An important note: a **More Details** button is located at the bottom right corner. Clicking on this button provides access to more information about the rating column, such as the median, standard deviation, coefficient of variation, and various other characteristics associated with the column.

Correlations

It helps in the study of the relationship between two variables, which is known as correlation. The heatmap below shows the relationships among all the variables with each other. Rank is 100% related to Rank, and that is why it is represented by the dark blue square at the top left.

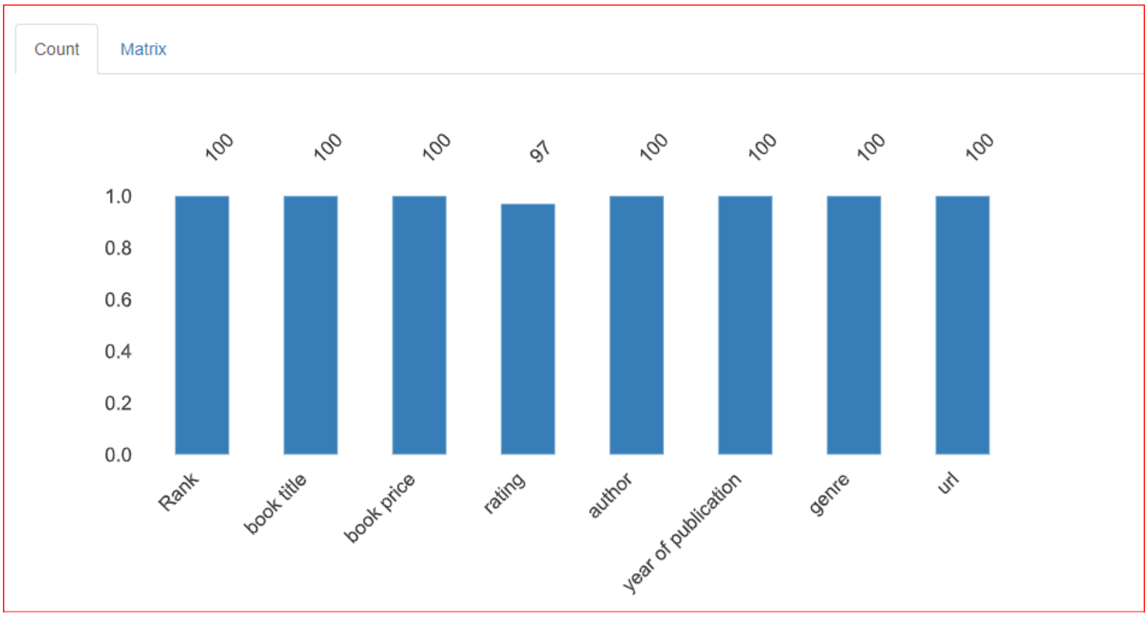
The year of publication is moderately related to the book price (around 0.75), which is represented by the light blue color because they are not fully related. For example, the book price is 20.93, and the year of publication is 2023, so these numbers are somewhat related to each other.



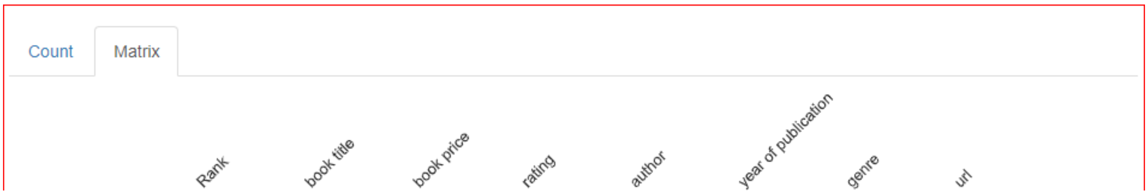


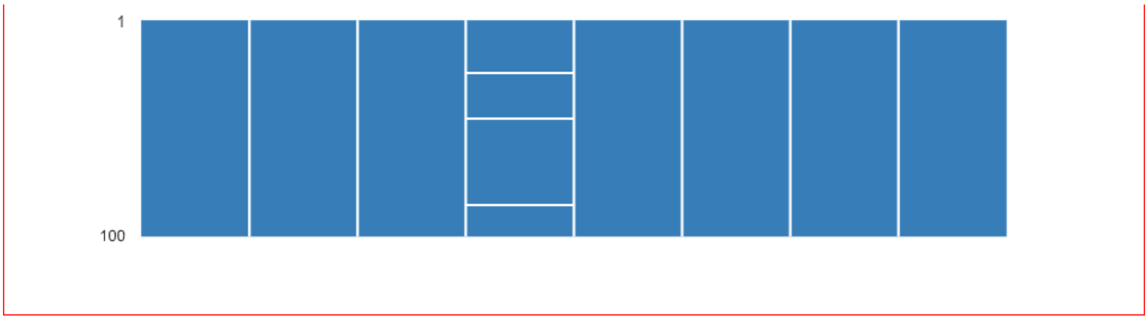
Missing Values

This section provides information about the missing values in the dataset. The **Count** tab of this section indicates that there are 3 missing values in the **rating** column.



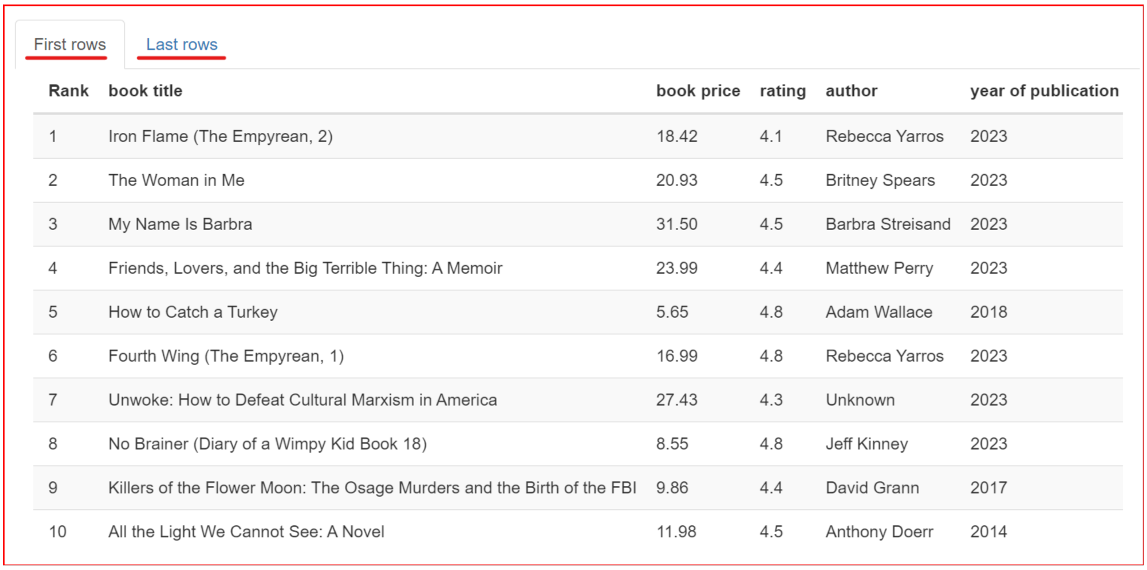
In the **Matrix** tab of the missing values section, three horizontal lines are present in the **Rating** column, which indicates that three values are missing in the column.





Sample

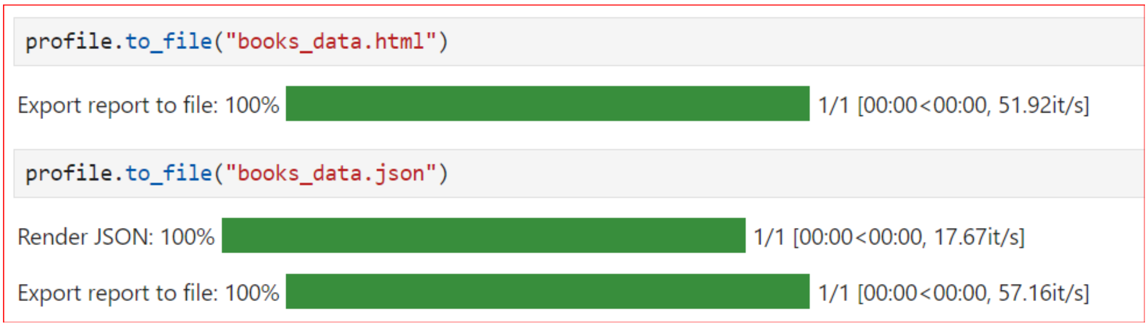
This section contains a sample of the dataset. It displays the first and last 10 rows of the dataset.


First rows
Last rows

Rank	book title	book price	rating	author	year of publication
1	Iron Flame (The Empyrean, 2)	18.42	4.1	Rebecca Yarros	2023
2	The Woman in Me	20.93	4.5	Britney Spears	2023
3	My Name Is Barbra	31.50	4.5	Barbra Streisand	2023
4	Friends, Lovers, and the Big Terrible Thing: A Memoir	23.99	4.4	Matthew Perry	2023
5	How to Catch a Turkey	5.65	4.8	Adam Wallace	2018
6	Fourth Wing (The Empyrean, 1)	16.99	4.8	Rebecca Yarros	2023
7	Unwoke: How to Defeat Cultural Marxism in America	27.43	4.3	Unknown	2023
8	No Brainer (Diary of a Wimpy Kid Book 18)	8.55	4.8	Jeff Kinney	2023
9	Killers of the Flower Moon: The Osage Murders and the Birth of the FBI	9.86	4.4	David Grann	2017
10	All the Light We Cannot See: A Novel	11.98	4.5	Anthony Doerr	2014

Saving the Profile Report

Your profile report is generated, and you may want to save it for further use, such as extracting useful data from the profile report or integrating it with other applications. You can save the report in HTML and JSON formats. The `to_file()` method will save the report outside the Jupyter Notebook.



```
profile.to_file("books_data.html")
```

```
Export report to file: 100% 1/1 [00:00<00:00, 51.92it/s]
```

```
profile.to_file("books_data.json")
```

```
Render JSON: 100% 1/1 [00:00<00:00, 17.67it/s]
Export report to file: 100% 1/1 [00:00<00:00, 57.16it/s]
```

Here is the complete code for the Pandas profiling:

```
from ydata_profiling import ProfileReport
import pandas as pd

df = pd.read_csv("trending-books.csv")
profile = ProfileReport(df, title="Trending Books")
profile.to_notebook_iframe()
profile.to_file("books_data.html")
```

Was this helpful?

Advanced Uses

For generating the report, we simply pass the CSV file and nothing else. We do not include any extra elements; only default values are used in the actions.

However, there might be sections you want to omit or include additional information. This is where the advanced uses of Pandas Profiling come into play. You can control various aspects of the report by changing the default configuration.

If you are interested in learning more about data analysis and visualization tools, read out 21 Essential Python Tools.

Dataset Description & Metadata

When sharing reports with coworkers or publishing online, it might be important to include metadata of the dataset, such as the author, copyright holder, or descriptions. `ydata-profiling` allows complementing a report with that information.

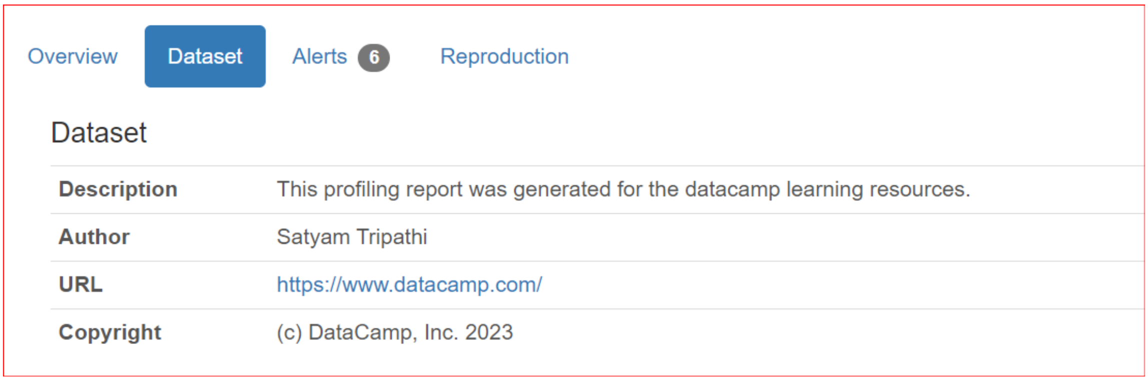
The currently supported properties are description, creator, author, url, copyright_year, and copyright_holder. By default, the datasets are presented in the **Overview** section of the report.

```
from ydata_profiling import ProfileReport
import pandas as pd

df = pd.read_csv("trending-books.csv")
report = ProfileReport(
    df,
    title="Trending Books",
    dataset={
        "description": "This profiling report
was generated for the datacamp learning
resources.",
        "author": "Satyam Tripathi",
        "copyright_holder": "DataCamp, Inc.",
        "copyright_year": 2023,
        "url": "<https://www.datacamp.com/>",
    },
)
report.to_notebook_iframe()
```

Was this helpful?

Here’s the code output:



Dataset

Description	This profiling report was generated for the datacamp learning resources.
Author	Satyam Tripathi
URL	https://www.datacamp.com/
Copyright	(c) DataCamp, Inc. 2023

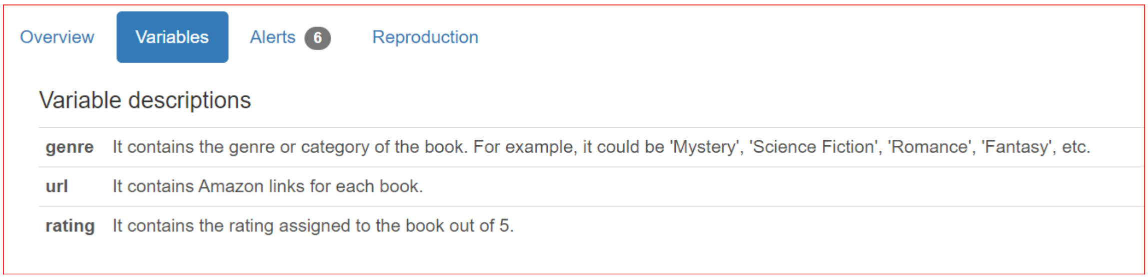
In addition to offering dataset details, users often want to include column-specific descriptions when sharing reports with team members and stakeholders. By default, these descriptions are presented in the **Overview** section of the report.

```
from ydata_profiling import ProfileReport
import pandas as pd

df = pd.read_csv("trending-books.csv")
report = ProfileReport(
    df,
    title="Trending Books",
    variables={
        "descriptions": {
            "genre": "It contains the genre or category of the book. For example, it could be 'Mystery', 'Science Fiction', 'Romance', 'Fantasy', etc.",
            "url": "It contains Amazon links for each book.",
            "rating": "It contains the rating assigned to the book out of 5.",
        }
    },
)
report.to_notebook_iframe()
```

Was this helpful?

Here’s the code output:



Variable descriptions

genre	It contains the genre or category of the book. For example, it could be 'Mystery', 'Science Fiction', 'Romance', 'Fantasy', etc.
url	It contains Amazon links for each book.
rating	It contains the rating assigned to the book out of 5.

Profiling Large Datasets

By default, `ydata-profiling` comprehensively summarizes the input dataset to provide the most insights for data analysis. For small datasets, these computations can be performed quickly. However, for larger datasets, it may become too unwieldy.

`ydata-profiling` includes a minimal configuration file where the most expensive computations are turned off by default. This configuration excludes time-consuming sections such as correlations, interactions, etc.

```
from ydata_profiling import ProfileReport
import pandas as pd

df = pd.read_csv("trending-books.csv")
report = ProfileReport(df, minimal=True)
report.to_notebook_iframe()
```

Was this helpful?

`ydata-profiling` also provides several alternatives to overcome the challenge of handling large datasets. Explore them here.

Conclusion

In the article, you learned about the unique library,`ydata-profiling,`formerly known "Pandas Profiling," for creating reports with just a couple of lines of code. You learned about how to generate the profile report and explore all the sections and tabs present in the profile report. Most importantly, you learned about the advanced uses of this library, which will take you one step ahead in your data science journey.

Pandas is the world's most popular Python library, used for everything from data manipulation to data analysis. To learn how to manipulate DataFrames, as you extract, filter, and transform real-world datasets for analysis, check out our course on Data Manipulation with pandas.