k Open in Kaggle

The Detective's First Case: Getting to Know Your Data (Pandas 101)

Story: Imagine you've just been hired as a Data Analyst for a new tech startup. Your boss hands you a mysterious dataset containing user information and asks you to "figure out who our users are".

You assume the role of a **Data Detective**. Your mission is to explore this dataset, uncover its secrets, and report back. Your primary tool? The powerful Python library, **Pandas**.

Let's begin the investigation.

Step 1: Gathering the Tools

Every detective needs a toolkit. For data analysis in Python, our essential tools are `pandas` (for data manipulation) and `numpy` (for numerical operations).

Action: Import the libraries.

In [1]:
```
import numpy as np
import pandas as pd
```

Step 2: Retrieving the Evidence

We've received a tip-off that the user data is located at a specific URL. We need to fetch this data and bring it into our workspace.

Action: Load the dataset from the remote URL.

In [2]:
```
url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/u.user'
```

Step 3: Opening the Case File

Now that we have the address, let's load the data into a Pandas DataFrame. We'll call our dataframe `users`.

Also, we notice the data is separated by pipes `|` instead of commas, and the first column looks like a unique ID, so we'll set `user_id` as our index.

Action: Read the CSV and assign it to `users`.

In [3]:
```
users = pd.read_csv(url, sep='|', index_col='user_id')
users.head()
```

Out[3]:

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213

Step 4: First Glance at the Suspects

Let's take a quick look at the first 25 entries to get a feel for the data. This is like glancing at the first few pages of a dossier.

Action: specificy `n=25` to see the first 25 rows.

In [4]:
```
users.head(25)
```

Out[4]:

	age	gender	occupation	zip_code
user_id				
1	24	M	technician	85711
2	53	F	other	94043
3	23	M	writer	32067
4	24	M	technician	43537
5	33	F	other	15213
6	42	M	executive	98101
7	57	M	administrator	91344
8	36	M	administrator	05201
9	29	M	student	01002
10	53	M	lawyer	90703
11	39	F	other	30329
12	28	F	other	06405
13	47	M	educator	29206
14	45	M	scientist	55106
15	49	F	educator	97301
16	21	M	entertainment	10309
17	30	M	programmer	06355
18	35	F	other	37212
19	40	M	librarian	02138
20	42	F	homemaker	95660
21	26	M	writer	30068
22	25	M	writer	40206
23	30	F	artist	48197
24	21	F	artist	94533
25	39	M	engineer	55107

Step 5: Checking the Back of the File

Sometimes important information is hidden at the end. Let's check the last 10 entries to ensure the data looks consistent throughout.

Action: Use `.tail()` to see the last 10 rows.

```
In [5]: users.tail(10)
```

Out[5]:

user_id	age	gender	occupation	zip_code
934	61	M	engineer	22902
935	42	M	doctor	66221
936	24	M	other	32789
937	48	M	educator	98072
938	38	F	technician	55038
939	26	F	student	33319
940	32	M	administrator	02215
941	20	M	student	97229
942	48	F	librarian	78209
943	22	M	student	77841

Step 6: Assessing the Scale of the Investigation

How many users are we dealing with? Is this a small group or a massive population?

Action: Check the number of observations (rows).

```
In [6]: print(f'Number of observations: {len(users)}')
```

```
Number of observations: 943
```

Step 7: Inventorying the Features

What information do we have on each user? We need to count the columns.

Action: Check the number of columns.

```
In [7]: print(f'Number of columns: {len(users.columns)}')
```

```
Number of columns: 4
```

Step 8: Naming the Attributes

Let's list out exactly what these columns are named, so we know what we are working with.

Action: Print column names.

```
In [8]: print(users.columns)
```

```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

Step 9: Understanding the Index

How is our dossier organized? We set `user_id` as the index earlier, but let's verify how pandas sees it.

Action: Inspect the index.

```
In [9]: users.index
```

```
Out[9]: Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
               ...
               934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
              dtype='int64', name='user_id', length=943)
```

Step 10: Checking Data Types

As a detective, you need to know if you're working with numbers, text, or dates. This determines what kind of analysis is possible.

Action: Check the data type of each column.

```
In [10]: users.dtypes
```

```
Out[10]: age            int64
         gender        object
         occupation    object
         zip_code      object
         dtype: object
```

Step 11: Zoning in on Occupations

Your boss is particularly interested in what our users *do* for a living. Let's isolate the 'occupation' column.

Action: Select and print the 'occupation' column.

```
In [11]: users.occupation
```

```
Out[11]: user_id
         1         technician
         2              other
         3             writer
         4         technician
         5              other
                    ...
         939          student
         940    administrator
         941          student
         942        librarian
         943          student
         Name: occupation, Length: 943, dtype: object
```

Step 12: Counting Unique Occupations

Data indicates we have many users, but how many *distinct* jobs are represented? Are they all engineers, or is there variety?

Action: Count the number of unique occupations.

```
In [12]: # value_counts() gives the count of unique values, then we count those checks
         count = users['occupation'].value_counts().count()
         print(f'Number of different occupations: {count}')
         # Alternatively: users['occupation'].nunique()
```

```
Number of different occupations: 21
```

Step 13: Finding the Most Common Job

What is the most popular profession among our users?

Action: Find the most frequent occupation.

```
In [13]: most_common = users['occupation'].value_counts().index[0]
         print(f'Most frequent occupation: {most_common}')
```

```
Most frequent occupation: student
```

Step 14: The Statistical Summary

Now, let's get a statistical overview of the numeric data (like age). This gives us mean, min, max, and quartiles instantly.

Action: Use `.describe()`.

```
In [14]: users.describe()
```

Out[14]:

	age
count	943.000000
mean	34.051962
std	12.192740
min	7.000000
25%	25.000000
50%	31.000000
75%	43.000000
max	73.000000

Step 15: The Full Report

We want to summarize everything, including text columns (like gender and occupation).

Action: Use `.describe(include='all')`.

In [15]:
```
users.describe(include='all')
```

Out[15]:

	age	gender	occupation	zip_code
count	943.000000	943	943	943
unique	NaN	2	21	795
top	NaN	M	student	55414
freq	NaN	670	196	9
mean	34.051962	NaN	NaN	NaN
std	12.192740	NaN	NaN	NaN
min	7.000000	NaN	NaN	NaN
25%	25.000000	NaN	NaN	NaN
50%	31.000000	NaN	NaN	NaN
75%	43.000000	NaN	NaN	NaN
max	73.000000	NaN	NaN	NaN

Step 16: Profiling the Occupations

Let's focus just on the summary statistics for the Occupation column again to see the top job and its frequency clearly.

Action: Describe only the occupation column.

```
In [16]: users['occupation'].describe()
```

```
Out[16]: count         943
         unique         21
         top       student
         freq          196
         Name: occupation, dtype: object
```

Step 17: Average Age of Users

How old is our average user?

Action: Calculate the mean of the age column.

```
In [17]: mean_age = users['age'].mean()
         print(f'Average age: {mean_age:.2f}')
```

```
Average age: 34.05
```

Step 18: The Rarest Age

We found the average age, but which age is the least common?

Action: Check the tail of the value counts for age.

```
In [18]: users['age'].value_counts().tail()
```

```
Out[18]: age
         7     1
         66    1
         11    1
         10    1
         73    1
         Name: count, dtype: int64
```

Conclusion

Great work, Detective!

You've successfully loaded the data, inspected its structure, and uncovered key insights:

- We have 943 users.
- They hold 21 different jobs.
- 'Student' is the most common occupation.
- The average age is around 34 years old.

This simple exploration is the foundation of all machine learning and data analysis projects. You must know your data before you can model it!

Mission Accomplished.

Author Details

Author: Tassawar Abbas
Email: abbas829@gmail.com