HOW TO HANDLE BIG DATA SCIENCE TASKS AS A BEGINNER

Several Useful Tips from a Real Task

Sibo Ding
Apr 28, 2023

ABOUT ME

Prior to this task:

- Passed COMP1117: Introduction to Python
- Learned Numpy and Pandas without too much practice

ABOUT THE TASK

Transform a 3000*1500 data frame to 75000*40 within 1 minute

There are **only input and output files**

→ Understand the data structures in both files by myself

	A	B	C	D	E	F	G
1	id	name	Poc_Sut__12	Poc_Sut__9	Poc_Sut	Poc_Sut_7	Poc_Sut_10
2	156	kick	65412	86453	789456	98645	874651
3	237	abba	48653	86453	846511		

15 common columns

1400+ different columns

Input Data

	A	B	C	D
1	id	name	quarter	Poc_Sut
2	156	kick	-12	65412
3	156	kick	-9	86453
4	156	kick	0	789456
5	156	kick	7	98645
6	156	kick	10	874651
7	237	abba	-12	48653
8	237	abba	-9	86453
9	237	abba	0	846511
10	237	abba	7	
11	237	abba	10	

25 quarters for each row

25 different columns

Output Data

TIP 1: ANALYZE WORK FLOW

Decompose a big task into several small tasks

Either forwards or backwards

	A	B	C	D
1	id	name	quarter	Poc_Sut
2	156	kick	-12	65412
3	156	kick	-9	86453
4	156	kick	0	789456
5	156	kick	7	98645
6	156	kick	10	874651
7	237	abba	-12	48653
8	237	abba	-9	86453
9	237	abba	0	846511
10	237	abba	7	
11	237	abba	10	

1 2 3

1. **Duplicate 25 rows for the common columns**
2. **Add “quarter” column**
3. **Transform remaining columns**

No method is mentioned

→ Totally **generalizable**

TIP 2: SIMPLIFY TASKS, FIND PATTERNS

I manually **group** 1400+ columns using Excel (20 mins)

	A	B	C	D	E
1	Poc_Sut	Dfl_Sut	Poc_Sut_chg	Dfl_Sut_avg	Qer_Sut_log
2	Poc_MIT	Dfl_MIT	Poc_MIT_chg	Dfl_MIT_avg	Qer_MIT_log
3	Poc_Tr_Ca	Dfl_Tr_Ca	Poc_Tr_Ca_chg	Dfl_Tr_Ca_avg	Qer_Tr_Ca_log

Stupid, but intuitive and effective to **understand the structure**

Same beginning, same middle, etc.

Filter columns we need (in green for example)

TIP 3: USE SIMPLIFIED TASKS FOR TRIAL

Create small tasks with a **similar structure**

To verify big tasks with shorter time

	A	B	C	D	E	F	G
1	id	name	Poc_Sut__12	Poc_Sut__9	Poc_Sut	Poc_Sut_7	Poc_Sut_10
2	156	kick	65412	86453	789456	98645	874651
3	237	abba	48653	86453	846511		

Input Data

	A	B	C	D
1	id	name	quarter	Poc_Sut
2	156	kick	-12	65412
3	156	kick	-9	86453
4	156	kick	0	789456
5	156	kick	7	98645
6	156	kick	10	874651
7	237	abba	-12	48653
8	237	abba	-9	86453
9	237	abba	0	846511
10	237	abba	7	
11	237	abba	10	

Output Data

TIP 4: READ DOCUMENTATIONS

Same as “help” panel in R

My personal habit: Examples (specific) → Parameters (abstract)

Examples

Constructing DataFrame from a dictionary.

```
>>> d = {'col1': [1, 2], 'col2': [3, 4]}
>>> df = pd.DataFrame(data=d)
>>> df
   col1  col2
0     1     3
1     2     4
```

https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

pandas.DataFrame

class pandas.DataFrame(*data=None, index=None, columns=None, dtype=None, copy=None*) [source]

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Parameters: **data** : ***ndarray (structured or homogeneous), Iterable, dict, or DataFrame***

Dict can contain Series, arrays, constants, dataclass or list-like objects. If data is a dict, column order follows insertion-order. If a dict contains Series which have an index defined, it is aligned by its index.

Changed in version 0.25.0: If data is a list of dicts, column order follows insertion-order.

TIP 5: SPEED, READABILITY, EXTENDIBILITY

Sometimes conflict → **need balance**

High readability helps you (colleagues, clients) better **maintain** code

High extendibility helps you handle similar new cases by **modifying a small part** of code

THANKS

Tip 6: Keep improving

Review previous work after learning new knowledge