

Pandas và Matplotlib

☑ Trắc nghiệm 29 ☑ Thực hành 1 ⬇ PDF bài

Dữ liệu của các cuộc thi thường là file CSV: `train.csv` có đáp án, `test.csv` là phần cần dự đoán, `sample_submission.csv` cho biết định dạng file nộp. pandas dùng để đọc, làm sạch và ghép các bảng đó; Matplotlib dùng để vẽ biểu đồ khi cần nhìn dữ liệu. Bài làm lần lượt các việc phải làm với một bộ dữ liệu như vậy: đọc file, xử lý ô trống, ghép bảng theo ID, rồi ghi file dự đoán và kiểm lại. Học sinh và điểm trong bài đều là dữ liệu giả lập.

MỤC TIÊU

- Đọc CSV, xem shape, kiểu từng cột và số ô trống.
- Lọc dòng, chọn cột, dùng groupby và merge.
- Vẽ biểu đồ hợp với câu hỏi cần trả lời.
- Ghi file nộp đúng thứ tự, đúng cột và kiểm lại.

Dữ liệu của đề

Đề dùng trong bài: dự đoán điểm Toán cuối kỳ của học sinh từ lớp, số giờ tự học mỗi tuần và điểm học kỳ 1. Mỗi dòng là một học sinh: `id` là mã học sinh, `diem_toan` là nhãn cần dự đoán, ba cột còn lại là dữ liệu đầu vào. `id` chỉ dùng để ghép bảng và ghép đáp án, không dùng để dự đoán.

Ô code đầu tự tạo dữ liệu giả cho đề: 400 học sinh có điểm (train) và 100 học sinh cần dự đoán (test), ghi ra một thư mục tạm. Không cần đọc kỹ ô này; từ ô sau trở đi ta làm như khi nhận đề thật, chỉ biết những gì có trong các file.

Python

```
import numpy as np
import pandas as pd
from pathlib import Path
from tempfile import TemporaryDirectory
```

```

demo_directory = TemporaryDirectory(prefix="phase0-demo-", dir=".")
data_dir = Path(demo_directory.name)

rng = np.random.default_rng(7)
n = 500
lop = rng.choice(["10A1", "10A2", "10A3", "10A4"], size=n)
gio = rng.gamma(2.0, 3.0, n).round(1)
hk1 = np.clip(rng.normal(6.5, 1.3, n), 0, 10).round(1)
bonus = pd.Series(lop).map({"10A1": 0.6, "10A2": 0.0, "10A3": -0.3, "10A4": 0.2}).to_numpy()
toan = np.clip(0.55 * hk1 + 0.12 * gio + bonus + rng.normal(0, 0.8, n) + 2.0, 0,
10).round(2)
gio[rng.random(n) < 0.06] = np.nan # một số học sinh không khai giừ
tự học

full = pd.DataFrame({"id": np.arange(1, n + 1), "lop": lop, "gio_tu_hoc": gio, "diem_hk1":
hk1, "diem_toan": toan})
full.iloc[:400].to_csv(data_dir / "train.csv", index=False, encoding="utf-8")
full.iloc[400:].drop(columns="diem_toan").to_csv(data_dir / "test.csv", index=False,
encoding="utf-8")
pd.DataFrame({"id": full["id"].iloc[400:], "diem_toan": 0.0}).to_csv(
data_dir / "sample_submission.csv", index=False, encoding="utf-8")
pd.DataFrame({"lop": ["10A1", "10A2", "10A3", "10A4"], "chuyen": [True, False, False,
False],
"si_so": [38, 42, 45, 40]}).to_csv(
data_dir / "lop.csv", index=False, encoding="utf-8")

```

Thư mục `data_dir` có bốn file: `train.csv` có nhãn, `test.csv` cần dự đoán, `sample_submission.csv` là mẫu file nộp, và `lop.csv` là bảng phụ ghi thông tin từng lớp. Đề quy định điểm nằm trong khoảng 0–10.

train.csv						lop.csv		
id	lop	gio	hk1	diem_toan		lop	chuyen	si_so
1	10A4	0.4	9.1	6.72	lop	10A1	True	38
...						...		

test.csv					sample_submission.csv	
id	lop	gio	hk1		id	diem_toan
401	10A2	NaN	7.1	id	401	0.0
...					...	

Bốn file của đề và hai khóa ghép: `lop` và `id`.

Đọc và nhìn dữ liệu

`pd.read_csv` đọc một file CSV (mỗi dòng một bản ghi, các giá trị ngăn bởi dấu phẩy) thành một DataFrame: một bảng có tên cột, mỗi cột có một kiểu dữ liệu riêng. Lấy riêng một cột thì được một Series. Vừa đọc xong, việc đầu tiên là xem bảng to cỡ nào và vài dòng đầu trông ra sao:

Python

```
train = pd.read_csv(data_dir / "train.csv", encoding="utf-8")
test = pd.read_csv(data_dir / "test.csv", encoding="utf-8")
print(train.shape, test.shape)
print(train.head(3))
```

KẾT QUẢ

(400, 5) (100, 4)

	id	lop	gio_tu_hoc	diem_hk1	diem_toan
0	1	10A4	0.4	9.1	6.72
1	2	10A3	0.6	5.9	6.84
2	3	10A3	2.8	4.9	4.79

Python

```
print(train.dtypes)
print(train.isna().sum())
```

KẾT QUẢ

```
id          int64
lop         object
gio_tu_hoc  float64
diem_hk1    float64
diem_toan   float64
dtype: object
id          0
lop         0
gio_tu_hoc  23
diem_hk1    0
diem_toan   0
dtype: int64
```

`dtypes` cho biết pandas đã đọc từng cột theo kiểu nào: các cột số là `int64` hoặc `float64`, cột chữ `lop` hiện là `object` (phiên bản pandas mới có thể ghi `str`). Bảng sau cho thấy 23

giá trị giờ tự học bị thiếu, được lưu là NaN. Kiểm ô thiếu bằng `isna()`, không bằng `value == np.nan`: theo chuẩn số thực, NaN không bằng chính nó.

index	Series				
	id	lop	gio_tu_hoc	diem_hk1	diem_toan
0	1	10A4	0.4	9.1	6.72
1	2	10A3	0.6	5.9	6.84
2	3	10A3	2.8	4.9	4.79
3	4	10A4	NaN	6.8	6.88
	int64	object	float64	float64	float64
					dtype

DataFrame: index, các cột Series, mỗi cột một dtype.

Tiếp theo, kiểm vài điều mà đề ngầm hứa: id không trùng, train không có nhãn trống, test có đủ các cột của train trừ nhãn, và không học sinh nào nằm ở cả hai tập. Viết thành `assert` thì sai chỗ nào sẽ báo ngay chỗ đó. (Nếu id có dạng "001", đọc bằng `dtype={"id": "string"}` để không mất số 0 ở đầu.)

Python

```
assert train["id"].is_unique and test["id"].is_unique
assert train["diem_toan"].notna().all()
assert set(test.columns) == set(train.columns) - {"diem_toan"}
assert not set(train["id"]) & set(test["id"])
print("schema: OK", "train/test ids disjoint: OK")
```

KẾT QUẢ

```
schema: OK train/test ids disjoint: OK
```

Các kiểm tra này chỉ kiểm định dạng. Chúng chưa bắt được trường hợp một học sinh có hai mã khác nhau; bài 1.3 sẽ quay lại chuyện đó.

Python

```
print(train.describe().round(2))
print(train["lop"].value_counts())
```

KẾT QUẢ

```
      id  gio_tu_hoc  diem_hk1  diem_toan
count  400.00      377.00    400.00    400.00
```

```

mean    200.50      5.43      6.55      6.44
std     115.61      4.13      1.39      1.16
min       1.00      0.30      2.90      3.51
25%     100.75      2.60      5.60      5.64
50%     200.50      4.40      6.50      6.43
75%     300.25      6.80      7.50      7.21
max     400.00     23.00     10.00      9.94
lop
10A4     110
10A3     103
10A1      98
10A2      89
Name: count, dtype: int64

```

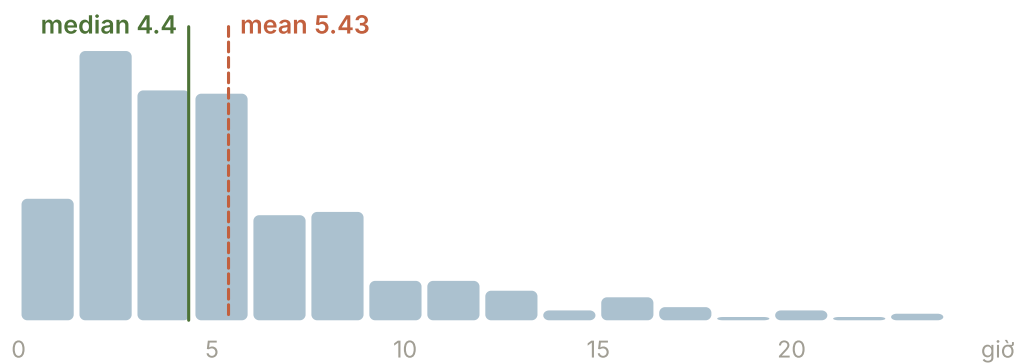
`describe` tóm tắt mọi cột số: số ô có giá trị (`count`), trung bình (`mean`), độ lệch chuẩn (`std`), nhỏ nhất, lớn nhất và các phân vị. Đọc bảng này là cách nhanh nhất để thấy điều lạ, như điểm âm, điểm trên 10 hay 100 giờ tự học mỗi tuần. Ở đây mọi cột đều hợp lý; riêng `gio_tu_hoc` có `count` là 377 thay vì 400, vì 23 ô trống. `value_counts` đếm số học sinh của từng lớp.

Vài khái niệm để đọc bảng `describe`:

- **Trung bình** (`mean`) là tổng chia cho số lượng. Biết trung bình và số lượng là biết tổng: 50 số có trung bình 40 thì tổng là 2000; thêm số 142 thì trung bình mới là $2142/51 = 42$.
- **Trung vị** (`median`, dòng 50%) là giá trị đứng giữa khi xếp các số theo thứ tự. **Yếu vị** (`mode`) là giá trị gặp nhiều nhất.
- **Phương sai** là trung bình của bình phương khoảng cách từ mỗi giá trị tới trung bình. **Độ lệch chuẩn** (`std`) là căn bậc hai của phương sai, cùng đơn vị với dữ liệu. Cả hai đo các giá trị trải xa trung bình đến đâu; cột mà mọi giá trị bằng nhau có phương sai 0. Bài 0.7 sẽ học kỹ.
- **Ngoại lai** (`outlier`) là giá trị nằm rất xa phần còn lại: có thể là lỗi nhập, như chiều cao 1700 cm, cũng có thể là trường hợp hiếm có thật. Ngoại lai kéo trung bình đi xa nhưng gần như không đổi trung vị: lương 8, 9, 10, 11 và 100 triệu có trung bình 27,6 nhưng trung vị 10.

Giờ tự học có vài bạn học rất nhiều, nên phân bố có đuôi dài bên phải (lệch phải). Khi đó trung bình bị kéo lên cao hơn trung vị, như hình dưới. Với phân bố lệch phải như thu

nhập hay giờ tự học, thường gặp trung bình > trung vị > yếu vị.



Giờ tự học lệch phải: đuôi dài kéo mean lên trên median.

HỎI NHANH

Bảng describe của cột `gio_tu_hoc` cho mean 5.43, median (dòng 50%) 4.40 và max 23.0. Phân bố giờ tự học nhiều khả năng có dạng nào?

- A Lệch phải, đuôi dài phía giá trị lớn kéo mean lên
- B Lệch trái, đuôi dài phía giá trị nhỏ kéo mean xuống
- C Gần đối xứng, vì mean và median chỉ chênh khoảng một giờ
- D Chưa nói được gì nếu chưa vẽ histogram

PHÒNG THEO NOAI 2026 Singapore · Vòng sơ loại · Câu 69

Chọn và lọc

Chọn cột bằng tên; chọn dòng bằng điều kiện. `loc` nhận nhãn dòng và tên cột, `iloc` nhận vị trí như mảng NumPy:

Python

```
print(train["diem_toan"].mean().round(3), train[["lop", "diem_toan"]].shape)
gioi = train[(train["diem_hk1"] >= 8) & (train["lop"] != "10A3")]
```

```
print(len(gioi), gioi["diem_toan"].mean().round(3))
print(train.loc[train["gio_tu_hoc"] > 20, ["id", "lop", "gio_tu_hoc"]])
print(train.iloc[:2, :3])
```

KẾT QUẢ

6.436 (400, 2)

52 7.416

	id	lop	gio_tu_hoc
220	221	10A3	22.6
252	253	10A1	20.1
287	288	10A4	23.0
399	400	10A3	21.7

	id	lop	gio_tu_hoc
0	1	10A4	0.4
1	2	10A3	0.6

Tên cột có dấu cách như "total price" chỉ chọn được bằng `df["total price"]`; cách viết `df.ten_cot` chỉ dùng khi tên cột là tên biến hợp lệ. Khi cắt dòng, `iloc[1:3]` theo vị trí và không lấy vị trí cuối như NumPy, còn `loc[20:40]` theo nhãn và lấy cả nhãn cuối.

Hai chi tiết cú pháp hay làm sai. Thứ nhất, kết hợp điều kiện dùng `&`, `|`, `~`, không dùng `and`, `or`, `not`; `and` hỏi giá trị đúng sai của cả một cột và báo lỗi. Thứ hai, mỗi điều kiện phải nằm trong ngoặc, vì `&` được ưu tiên trước `>=`.

Tạo cột mới bằng phép gán, và sắp xếp bằng `sort_values`:

Python

```
train["tien_bo"] = train["diem_toan"] - train["diem_hk1"]
print(train.sort_values("tien_bo", ascending=False)[["id", "diem_hk1", "diem_toan",
"tien_bo"]].head(3))
```

KẾT QUẢ

	id	diem_hk1	diem_toan	tien_bo
236	237	5.3	9.35	4.05
252	253	4.9	8.23	3.33
339	340	3.9	7.22	3.32

Khi muốn **sửa** một số ô theo điều kiện, dùng một lần `loc` với cả điều kiện dòng và tên cột. Viết hai lần chọn liên tiếp (gán nối chuỗi, chained assignment) thì phép gán rơi vào một bảng tạm do lần chọn đầu tạo ra, không vào bảng gốc. Đoạn sau sửa trên một bản sao để giữ nguyên nhãn thật:

Python

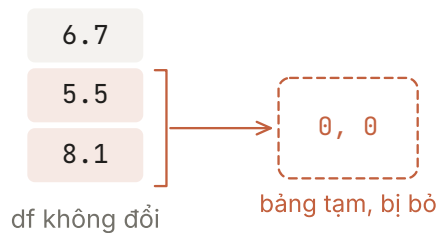
```
demo = train.copy()
demo[demo["lop"] == "10A1"]["diem_toan"] = 0          # SAI: sửa trên bảng tạm
print((demo["diem_toan"] == 0).sum())
demo.loc[demo["lop"] == "10A1", "diem_toan"] = 0      # ĐÚNG
print((demo["diem_toan"] == 0).sum())
```

KẾT QUẢ

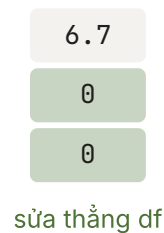
0
98

Dòng đầu không đổi gì trong demo; pandas chỉ in một cảnh báo chứ không dừng chương trình. Dòng sau đổi đúng 98 học sinh lớp 10A1. Vì vậy sau khi gán, nên in lại để chắc đã sửa đúng; không báo lỗi chưa có nghĩa là đã sửa.

`df[mask]["col"] = 0`



`df.loc[mask, "col"] = 0`



Gán nối chuỗi rơi vào bảng tạm; một lần loc sửa thẳng bảng gốc.

HỎI NHANH

Cần lọc các học sinh lớp 10A1 có điểm HK1 dưới 5. Viết thể nào cho đúng?

- A `train[train["lop"] == "10A1" and train["diem_hk1"] < 5]`
- B `train[train["lop"] == "10A1" & train["diem_hk1"] < 5]`
- C `train.loc["10A1", "diem_hk1" < 5]`
- D `train[(train["lop"] == "10A1") & (train["diem_hk1"] < 5)]`

Ô trống

Ô trống không có nghĩa là 0: “không khai giờ tự học” khác “không tự học giờ nào”. pandas có ba cách xử lý thường dùng: `dropna()` bỏ dòng có ô trống, `fillna(...)` điền một giá trị vào ô trống, `interpolate()` ước lượng từ các giá trị xung quanh. Bỏ dòng thì mất học sinh, mà học sinh ở test thì vẫn phải dự đoán, nên ở đây ta điền.

Giờ tự học lệch phải, nên điền bằng trung vị thay vì trung bình. Mỗi học sinh nhận trung vị giờ tự học của lớp mình; nếu gặp lớp không có trong train thì nhận trung vị chung. Cách này có tốt hơn điền một trung vị chung cho mọi người không thì chương 1 mới đo được.

Python

```
med_all = train["gio_tu_hoc"].median()
med_lop = train.groupby("lop")["gio_tu_hoc"].median()
print(round(med_all, 2), med_lop.round(2).to_dict())

def fill_hours(table):
    result = table.copy()
    group_value = result["lop"].map(med_lop)
    result["gio_tu_hoc"] = result["gio_tu_hoc"].fillna(group_value).fillna(med_all)
    return result

train = fill_hours(train)
test = fill_hours(test)
print(train["gio_tu_hoc"].isna().sum(), test["gio_tu_hoc"].isna().sum())
unseen = pd.DataFrame({"lop": ["10A5"], "gio_tu_hoc": [np.nan]})
print(fill_hours(unseen)["gio_tu_hoc"].tolist())
```

KẾT QUẢ

```
4.4 {'10A1': 4.2, '10A2': 4.4, '10A3': 4.7, '10A4': 4.45}
0 0
[4.4]
```

Output cho thấy: trung vị chung là 4.4, hai bảng không còn ô trống, và lớp lạ 10A5 nhận 4.4. Bước này nhờ `map`: nó tra trung vị theo tên lớp, lớp nào không có trong bảng tra thì cho ô trống, và `fillna(med_all)` điền tiếp ô đó.

10A4	NaN	—————>	4.45	
10A2	NaN	—————>	4.4	
10A1	5.2	- - - - ->	5.2	
10A1	NaN	—————>	4.2	
10A5	NaN	- - - - ->	4.4	median chung

median từ train

10A1	4.2
10A2	4.4
10A3	4.7
10A4	4.45

Ô thiếu nhận median của lớp mình; lớp lạ nhận median chung.

Để ý: các trung vị chỉ tính trên **train**, rồi dùng nguyên các số đó để điền cho test. Không tính lại trên test, vì khi dùng thật, dữ liệu mới đến từng phần và ta không được nhìn trước. Chương 1 sẽ gặp lại quy tắc này nhiều lần.

HỎI NHANH

Trên train, median giờ học của lớp A là 2, của lớp B là 4, của cả bảng là 3. Tức là `med_by_class = {"A": 2, "B": 4}` và `med_all = 3`. Ở test có một học sinh lớp C bị thiếu giờ học. Sau `test["lop"].map(med_by_class).fillna(med_all)`, ô của bạn ấy được điền số mấy?

A 3

B 4

C 0

D 2

groupby

`groupby` chia bảng thành từng nhóm theo giá trị của một cột, tính trên từng nhóm, rồi ghép kết quả lại. Đây là phép `GROUP BY` của SQL, và là công cụ dùng nhiều nhất khi tìm hiểu dữ liệu.

Python

```
print(train.groupby("lop")["diem_toan"].agg(["mean", "std", "count"]).round(2))
```

KẾT QUẢ

	mean	std	count
lop			
10A1	6.79	1.00	98
10A2	6.41	1.17	89
10A3	5.96	1.14	103
10A4	6.58	1.16	110

Lớp 10A1 có điểm trung bình 6.79, lớp 10A3 có 5.96, chênh khoảng 0.83 điểm. Như vậy biết lớp có thể giúp đoán điểm, nhưng chưa có nghĩa là học lớp 10A1 làm điểm cao lên.

Các hàm gộp như mean, count bỏ qua ô trống. Thử trên một bảng sáu học sinh, lớp B có một ô thiếu điểm:

Python

```
hand = pd.DataFrame({
    "lop": ["A", "A", "A", "B", "B", "B"],
    "diem": pd.Series([6, 8, 7, 5, np.nan, 9], dtype="float64"),
})
print(hand.groupby("lop")["diem"].mean().to_dict())
print(hand.groupby("lop")["diem"].agg(["count", "size"]).to_dict())
print((hand["diem"] >= 7).mean(), (hand["diem"].dropna() >= 7).mean())
```

KẾT QUẢ

```
{'A': 7.0, 'B': 7.0}
{'count': {'A': 3, 'B': 2}, 'size': {'A': 3, 'B': 3}}
0.5 0.6
```

Lớp B có trung bình $(5 + 9)/2 = 7$ vì ô trống bị bỏ qua. count không đếm ô trống, còn size đếm mọi dòng, nên lớp B có count là 2, size là 3. Phép so sánh thì khác: `NaN >= 7` cho False, nên tỉ lệ điểm từ 7 trở lên là $3/6 = 0.5$ khi tính trên mọi dòng, và $3/5 = 0.6$ khi chỉ tính học sinh đã có điểm. Hai con số trả lời hai câu hỏi khác nhau.

Có hai cách lấy kết quả theo nhóm. agg trả về một dòng cho mỗi nhóm, như bảng trên. transform trả về một giá trị cho mỗi dòng của bảng gốc, giữ nguyên số dòng, nên gán thẳng thành cột mới được:

Python

```
agg = train.groupby("lop")["diem_toan"].mean()
tf = train.groupby("lop")["diem_toan"].transform("mean")
print(agg.shape, tf.shape)
train["toan_tb_lop"] = tf
print(train[["lop", "diem_toan", "toan_tb_lop"]].head(3).round(2))
```

KẾT QUẢ

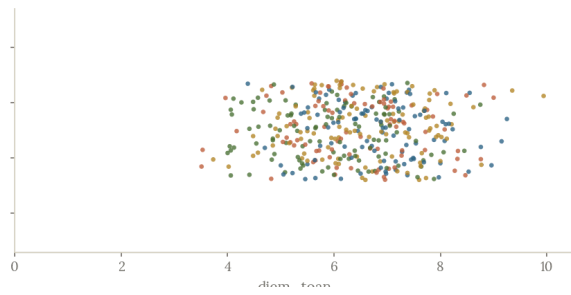
(4,) (400,)

	lop	diem_toan	toan_tb_lop
0	10A4	6.72	6.58
1	10A3	6.84	5.96
2	10A3	4.79	5.96

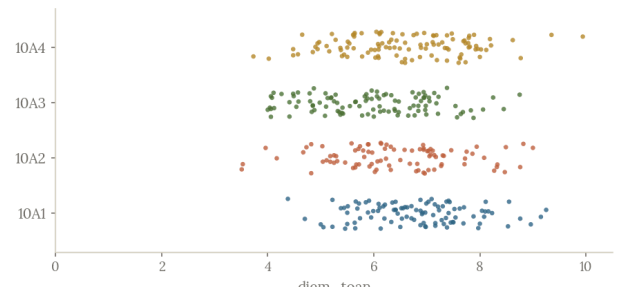
Mỗi dòng nhận điểm trung bình của lớp mình. Cột này chỉ để xem; không đưa vào làm đầu vào cho model, vì nó tính từ chính nhãn `diem_toan`, trong đó có điểm của chính học sinh đó. Bài 2.11 sẽ nói cách dùng loại cột này cho đúng.

transform: 400 dòng			agg: 4 dòng	
lop	điểm	TB lớp		
10A4	6.72	6.58	10A1	6.79
10A3	6.84	5.96	10A2	6.41
10A3	4.79	5.96	10A3	5.96
10A4	6.88	6.58	10A4	6.58
10A3	6.45	5.96		
10A4	8.02	6.58		
10A4	7.34	6.58		
10A1	5.50	6.79		

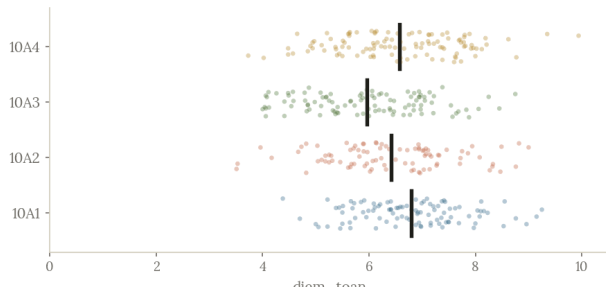
transform giữ mọi dòng; agg thu về một dòng mỗi lớp.



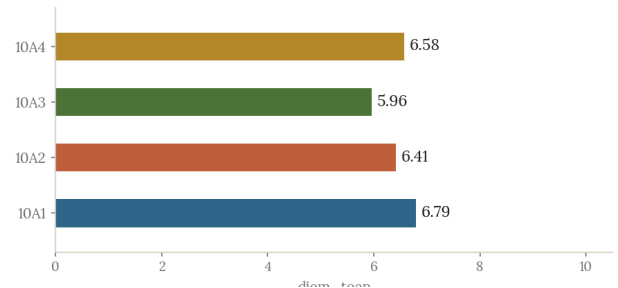
400 dòng của train.csv, mỗi chấm là một học sinh



split: tách các dòng theo cột lop



apply: mỗi nhóm lấy trung bình diem_toan (vạch đậm)



combine: df.groupby("lop")["diem_toan"].mean() thành một dòng mỗi lớp

groupby trên 400 học sinh của bài: tách dòng theo lớp, tính trung bình điểm toán trong từng nhóm, rồi gộp thành một bảng bốn dòng.

Đoạn trên gán `tf` thành cột mới mà các dòng vẫn khớp đúng học sinh. Đó là vì pandas ghép hai bảng theo **index**, tức nhãn dòng in ở cột ngoài cùng bên trái, chứ không theo vị trí như NumPy. Index có thể khác cột id. Ví dụ sau cố ý đảo thứ tự index:

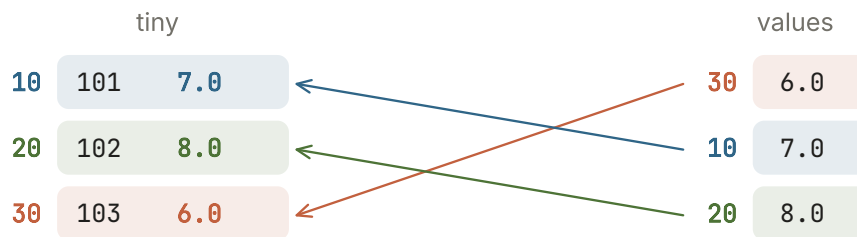
Python

```
tiny = pd.DataFrame({"id": [101, 102, 103]}, index=[10, 20, 30])
values = pd.Series([6.0, 7.0, 8.0], index=[30, 10, 20])
tiny["score"] = values
print(tiny["score"].tolist())
```

KẾT QUẢ

```
[7.0, 8.0, 6.0]
```

Dòng có index 10 lấy 7.0, không lấy giá trị đầu tiên là 6.0. Cách ghép này tiện khi index có nghĩa, nhưng dễ gây nhầm nếu ta tưởng pandas gán theo vị trí. Với mã học sinh, nên ghép rõ ràng theo cột id bằng merge ở mục sau.



pandas ghép theo nhãn index, không theo vị trí.

HỎI NHANH

train có 400 dòng, chia thành 4 lớp. `train.groupby("lop")`
`["diem_toan"].transform("max")` có bao nhiêu phần tử?

A 4

B 1

C 1600

D 400

merge

Thông tin về lớp (lớp chuyên hay không, sĩ số) nằm ở file khác. `merge` ghép hai bảng theo một cột khóa, giống JOIN của SQL:

Python

```
lop_info = pd.read_csv(data_dir / "lop.csv", encoding="utf-8")
merged = train.merge(lop_info, on="lop", how="left", validate="many_to_one")
assert len(merged) == len(train)
print(train.shape, merged.shape)
print(merged.groupby("chuyen")["diem_toan"].mean().round(2))
```

KẾT QUẢ

(400, 7) (400, 9)

chuyen

```
False    6.32
True     6.79
Name: diem_toan, dtype: float64
```

Ghép xong, số dòng phải giữ nguyên: mỗi học sinh khớp đúng một lớp. Điều đó chỉ đúng khi khóa ở bảng phụ không bị trùng. Nếu file lớp vô tình có một dòng lặp, merge sẽ nhân đôi mọi học sinh của lớp đó mà không báo gì:

Python

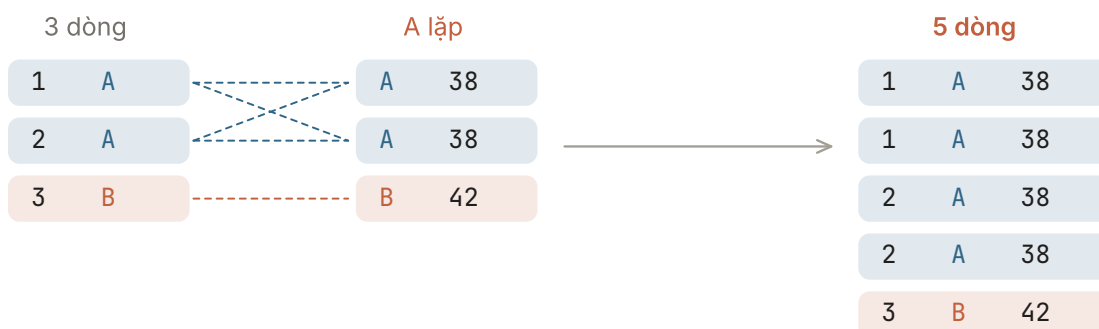
```
dup = pd.concat([lop_info, lop_info.iloc[[0]]])          # 10A1 xuất hiện hai lần
print(train.merge(dup, on="lop", how="left").shape)
try:
    train.merge(dup, on="lop", how="left", validate="many_to_one")
except pd.errors.MergeError as e:
    print("MergeError:", str(e).splitlines()[0])
```

KẾT QUẢ

(498, 9)

MergeError: Merge keys are not unique in right dataset; not a many-to-one merge

Tham số `validate="many_to_one"` nói với pandas rằng “nhiều học sinh ứng với một lớp”, và pandas báo lỗi ngay nếu điều đó sai. Nên luôn dùng nó, hoặc ít nhất in shape trước và sau khi ghép: bảng train bị lặp dòng thì mọi con số về sau đều sai theo.



Khóa lặp ở bảng phải làm left merge nhân bản dòng.

`how="left"` giữ mọi dòng của bảng bên trái, kể cả dòng không khớp được (khi đó các cột mới là NaN). `how="inner"` chỉ giữ dòng khớp được ở cả hai bên, và có thể làm mất mẫu một cách âm thầm.

HỎI NHANH

Bảng trái có ba học sinh, thuộc các lớp A, A, B. Bảng phải có ba dòng thông tin lớp, cũng là A, A, B, nghĩa là lớp A bị lặp. Ghép hai bảng bằng merge với how="left" theo cột lớp thì được bao nhiêu dòng?

A 3

B 4

C 5

D 9

Vẽ biểu đồ trả lời một câu hỏi

Một biểu đồ tốt trả lời một câu hỏi cụ thể. Chọn loại biểu đồ theo câu hỏi, không theo thói quen:

Câu hỏi	Biểu đồ	Matplotlib
Một biến phân bố thế nào?	Histogram	ax.hist
Hai biến số có liên quan không?	Scatter	ax.scatter
So sánh giá trị giữa các nhóm	Bar	ax.bar
Một đại lượng thay đổi theo thời gian	Line	ax.plot
So phân bố của nhiều nhóm	Boxplot	ax.boxplot

Boxplot vẽ mỗi nhóm thành một hộp: vạch giữa là trung vị, hộp chứa một nửa số giá trị ở giữa, hai râu kéo ra phần còn lại, chấm rời là ngoại lai.

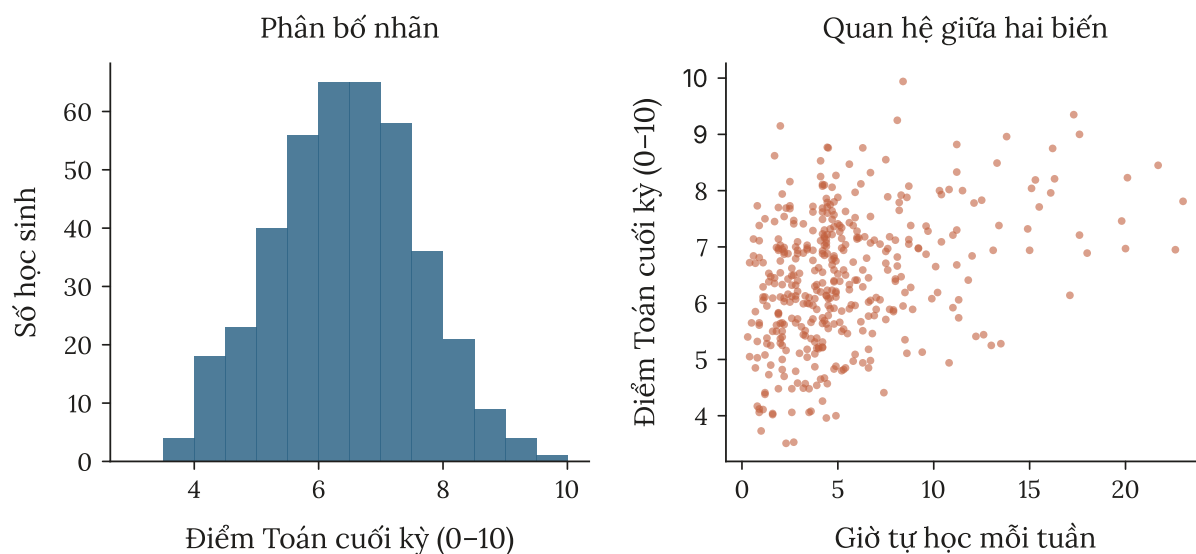
Matplotlib có hai cách viết; nên dùng cách hướng đối tượng: tạo fig (cả hình) và ax (từng ô vẽ), rồi gọi lệnh vẽ trên ax.

```
import matplotlib.pyplot as plt

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10, 4))
ax1.hist(train["diem_toan"], bins=20)
ax1.set_xlabel("điểm Toán cuối kỳ")
ax1.set_ylabel("số học sinh")
ax2.scatter(train["gio_tu_hoc"], train["diem_toan"], s=10, alpha=0.6)
ax2.set_xlabel("giờ tự học mỗi tuần")
ax2.set_ylabel("điểm Toán cuối kỳ")
fig.tight_layout()
fig.savefig(data_dir / "eda.png", dpi=120)
plt.show()
plt.close(fig)
print(train[["gio_tu_hoc", "diem_hk1", "diem_toan"]].corr().round(2))
```

KẾT QUẢ

	gio_tu_hoc	diem_hk1	diem_toan
gio_tu_hoc	1.00	0.00	0.36
diem_hk1	0.00	1.00	0.56
diem_toan	0.36	0.56	1.00

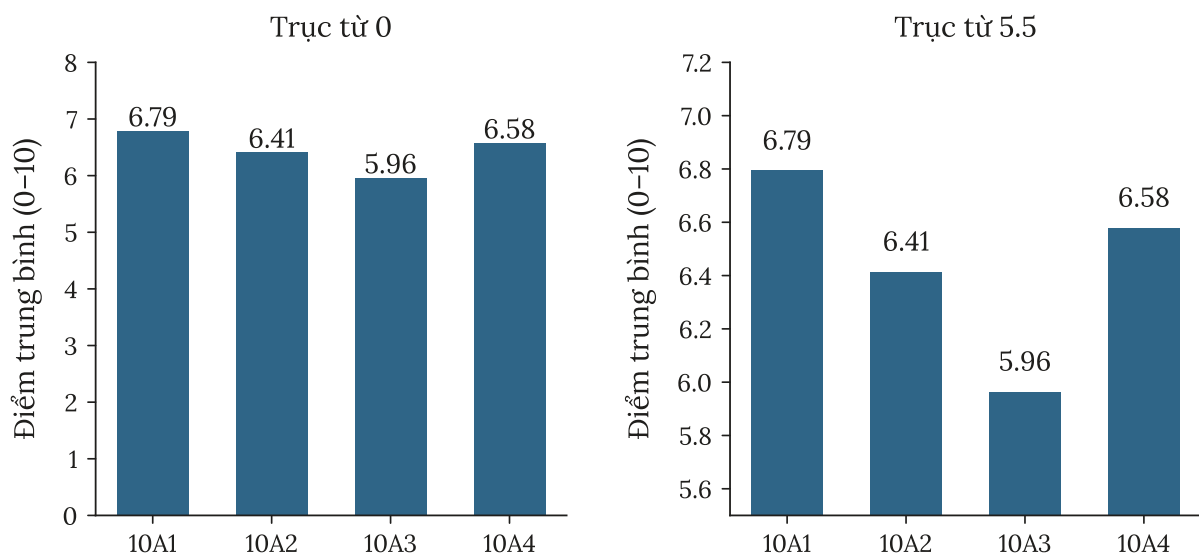


Phân bố điểm và quan hệ giữa giờ tự học với điểm.

Histogram cho thấy điểm tập trung quanh 6-7 và không có điểm nào ngoài 0-10. Scatter đi lên nhưng các chấm tản rộng: tự học nhiều thì điểm có xu hướng cao hơn, nhưng không chặt. Lưu ý 23 chấm trên scatter có giờ tự học là giá trị ta điền vào, không phải số học sinh khai.

Dòng `corr()` in **hệ số tương quan** giữa từng cặp cột, một số trong khoảng $[-1, 1]$: gần 1 là hai biến cùng tăng theo một đường thẳng, gần -1 là biến này tăng thì biến kia giảm, gần 0 là không có quan hệ kiểu đường thẳng. Điểm Toán tương quan 0.56 với điểm HK1 và 0.36 với giờ tự học. Hệ số này không phải “số điểm tăng thêm mỗi giờ”, và cũng không chứng minh cái này gây ra cái kia: ngày nóng bán được nhiều kem và cũng nhiều người đi bơi, nhưng ăn kem không làm người ta đi bơi.

Biểu đồ cột có một cái bẫy hay gặp: trục tung không bắt đầu từ 0. Chiều cao cột là thứ mắt người so sánh, nên cắt trục sẽ phóng đại khác biệt:



Cùng trung bình, cắt trục y làm chênh lệch trông lớn hơn.

Mọi biểu đồ nộp kèm bài hay đưa vào báo cáo cần có nhãn trục kèm đơn vị, và trục tung của biểu đồ cột bắt đầu từ 0.

HỎI NHANH

Muốn xem điểm HK1 có liên quan đến điểm Toán cuối kỳ không, ta nên vẽ biểu đồ gì?

- A Histogram điểm Toán, tô màu theo mức điểm HK1
- B Scatter, mỗi học sinh là một điểm (HK1, Toán)
- C Biểu đồ cột điểm Toán trung bình của từng lớp
- D Đường nối điểm Toán của các học sinh theo id

Tạo file nộp

Chưa cần model, ta dùng một cách đoán đơn giản để có file nộp đầu tiên: mỗi học sinh ở test được đoán bằng điểm Toán trung bình của lớp mình trên train; lớp lạ thì nhận trung bình chung. Cách đoán đơn giản như vậy gọi là **baseline**; nó tốt đến đâu thì chương 1 mới đo. Bảng dự đoán được cố ý đảo ngược thứ tự để kiểm rằng ghép theo id vẫn cho file nộp đúng thứ tự của mẫu.

Python

```
mean_by_lop = train.groupby("lop")["diem_toan"].mean()
pred_table = test[["id", "lop"]].iloc[:, :-1].copy()
pred_table["diem_toan"] =
pred_table["lop"].map(mean_by_lop).fillna(train["diem_toan"].mean())

sample = pd.read_csv(data_dir / "sample_submission.csv", encoding="utf-8")
sub = sample[["id"]].merge(
    pred_table[["id", "diem_toan"]], on="id", how="left", validate="one_to_one")
assert sub.shape == sample.shape, sub.shape
assert list(sub.columns) == list(sample.columns)
assert sub["id"].equals(sample["id"])
assert np.isfinite(sub["diem_toan"].to_numpy()).all()
assert sub["diem_toan"].between(0, 10).all()
sub.to_csv(data_dir / "submission.csv", index=False, encoding="utf-8")
read_back = pd.read_csv(data_dir / "submission.csv", encoding="utf-8")
```

```
assert read_back.columns.tolist() == sample.columns.tolist()
assert read_back["id"].equals(sample["id"])
assert np.isfinite(read_back["diem_toan"].to_numpy()).all()
print(read_back.head(3).round(3))
print("file:", read_back.shape, "OK")
```

KẾT QUẢ

```
   id  diem_toan
0  401      5.964
1  402      6.579
2  403      6.414
file: (100, 2) OK
```

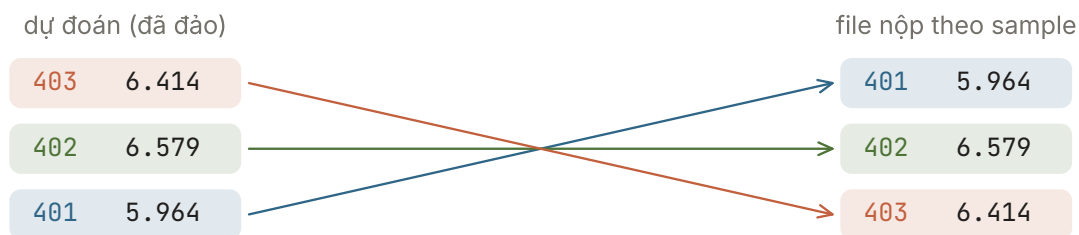
Nhìn ba ID đầu: 401, 402, 403 đúng thứ tự mẫu, dù bảng dự đoán đang đảo ngược.

HỎI NHANH

pred_table đang xếp ID theo thứ tự 500, 499, ..., 401, còn sample xếp 401, 402, ..., 500. Chạy `sub = sample[["id"]].merge(pred_table, on="id", how="left").ID` trong sub theo thứ tự nào?

- A 500, 499, ..., 401, theo bảng dự đoán
- B 401, 402, ..., 500, theo bảng bên trái
- C Tăng dần, vì merge luôn sắp theo khóa
- D Không biết trước, phải tự sort_values sau đó

Các dòng assert kiểm: đủ số dòng, đúng tên và thứ tự cột, đúng thứ tự id, không có ô trống hay vô cùng (`np.isfinite`; `notna()` không bắt được vô cùng), và mọi điểm nằm trong 0–10. Sau khi ghi, đọc lại file để kiểm chính file sẽ nộp chứ không chỉ bảng trong bộ nhớ.



Ghép theo id vào khung sample: đúng học sinh, đúng thứ tự.

Tham số `index=False` cũng cần được kiểm bằng một phản ví dụ:

Python

```
sub.to_csv(data_dir / "submission_bad.csv", encoding="utf-8") # quên index=False
print(pd.read_csv(data_dir / "submission_bad.csv").columns.tolist())
```

KẾT QUẢ

```
['Unnamed: 0', 'id', 'diem_toan']
```

Thiếu `index=False`, pandas ghi thêm một cột chỉ số không tên ở đầu file. Máy chấm sẽ thấy ba cột thay vì hai và từ chối bài, hoặc tệ hơn, đọc nhầm cột.

Baseline này tốt đến đâu thì phải đo trên dữ liệu có đáp án, bằng cách để riêng một phần tập train. Đó là nội dung của bài 1.1 và 1.3.

Kiểm xong thì dọn thư mục tạm. Khi chạy bài thật, chỉ dọn dữ liệu tạo để thử; giữ dữ liệu đề và file cần nộp:

Python

```
demo_directory.cleanup()
print(data_dir.exists())
```

KẾT QUẢ

```
False
```

Ngày giờ, chuỗi và dữ liệu không phải bảng

Cột thời gian thường là chuỗi, mã học sinh có thể lẫn khoảng trắng và chữ hoa thường. Đề ảnh, văn bản, âm thanh thì cho một thư mục tệp kèm bảng nhãn.

Python

```
log = pd.DataFrame({"thoi_diem": ["2026-03-02 07:45", "2026-03-02 21:10", "2026-03-07 09:30"],
                    "ma_hs": ["hs001", "HS002 ", "hs003"]})
log["thoi_diem"] = pd.to_datetime(log["thoi_diem"], format="%Y-%m-%d %H:%M")
log["gio"] = log["thoi_diem"].dt.hour
log["thu"] = log["thoi_diem"].dt.dayofweek          # 0 là thứ Hai, 6 là Chủ nhật
log["ma_hs"] = log["ma_hs"].str.strip().str.upper()
print(log)
print(log["thoi_diem"].max() - log["thoi_diem"].min())
```

KẾT QUẢ

	thoi_diem	ma_hs	gio	thu
0	2026-03-02 07:45:00	HS001	7	0
1	2026-03-02 21:10:00	HS002	21	0
2	2026-03-07 09:30:00	HS003	9	5
5	days 01:45:00			

- `pd.to_datetime` đổi chuỗi thành thời điểm. Nên ghi rõ format, vì 03-02 có thể là ngày 3 tháng 2 hoặc ngày 2 tháng 3.
- `.dt` lấy giờ, thứ, tháng của cả cột; trừ hai thời điểm được một khoảng thời gian.
- `.str` xử lý chuỗi trên cả cột. Chưa chuẩn hóa thì "hs001" và "HS001" là hai mã khác nhau, và merge theo mã sẽ bỏ sót dòng.
- Cột giá ghi "1.200.000 đ" là chuỗi nên `mean()` báo lỗi. `.str.replace(r"\D", "", regex=True)` bỏ mọi ký tự không phải chữ số, rồi `.astype(int)` đổi thành 1200000.

Với thư mục ảnh và tệp nhãn, đọc ảnh **theo thứ tự của bảng nhãn** để ảnh thứ *i* đi với nhãn thứ *i*:

Python

```
from PIL import Image
```

```

with TemporaryDirectory() as directory:
    root = Path(directory)
    (root / "images").mkdir()
    for name, value in [("c.png", 120), ("a.png", 40), ("b.png", 200)]:
        Image.new("L", (4, 3), color=value).save(root / "images" / name) # ảnh xám rộng 4,
cao 3
    pd.DataFrame({"file": ["a.png", "b.png", "c.png"], "label": ["toi", "sang",
"vua"]})).to_csv(
        root / "labels.csv", index=False)

    labels = pd.read_csv(root / "labels.csv")
    images = [np.asarray(Image.open(root / "images" / name).convert("L")) for name in
labels["file"]]
    X = np.stack(images).astype(np.float32) / 255
    print(X.shape, X.dtype, X.mean(axis=(1, 2)).round(3))
    print(labels["label"].tolist())

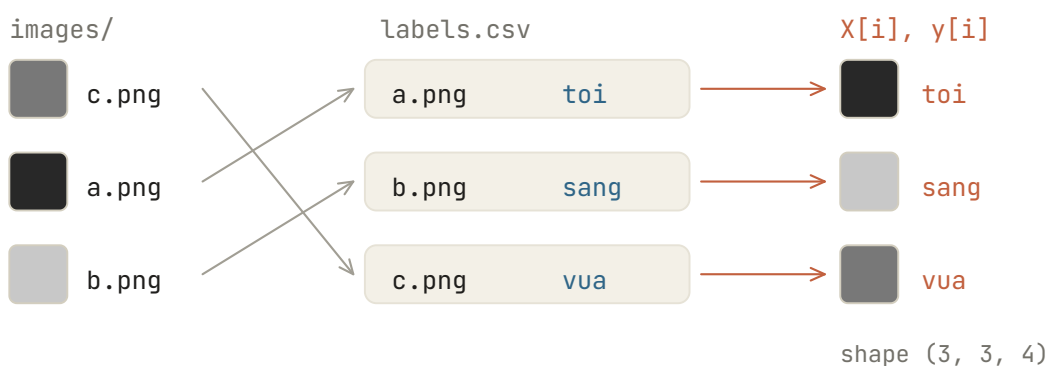
```

KẾT QUẢ

```

(3, 3, 4) float32 [0.157 0.784 0.471]
['toi', 'sang', 'vua']

```



Đọc ảnh theo cột file của labels.csv: ảnh thứ i luôn đi với nhãn thứ i .

PIL ghi kích thước theo (rộng, cao), NumPy theo (hàng, cột): ảnh rộng 4, cao 3 thành mảng (3, 4), ba ảnh thành (3, 3, 4). Độ sáng 0.157 là $40/255$, đúng ảnh tối của nhãn toi. Ảnh màu thì dùng `.convert("RGB")`, để ảnh có kênh trong suốt hay ảnh xám không làm lệch số kênh.

HỎI NHANH

Đọc ảnh bằng `sorted(Path("images").iterdir())` rồi ghép với cột `label` của `labels.csv` theo vị trí. Vì sao cách này nguy hiểm?

- A `iterdir` chỉ đọc được tối đa 1000 tệp
- B Ảnh đọc theo cách này luôn thành ảnh xám
- C Thứ tự tên tệp sau khi sắp chưa chắc trùng thứ tự dòng trong `labels.csv`
- D `sorted` làm đổi tên tệp trên đĩa

Văn bản hay ở dạng JSON Lines, mỗi dòng một đối tượng JSON. Âm thanh thường là WAV: một dãy số nguyên kèm tần số lấy mẫu.

Python

```
import json, wave

with TemporaryDirectory() as directory:
    root = Path(directory)
    rows = [{"id": 1, "text": "Học máy", "tags": ["ai"]}, {"id": 2, "text": "Dữ liệu bảng",
"tags": []}]
    (root / "train.jsonl").write_text("\n".join(json.dumps(r, ensure_ascii=False) for r in
rows), encoding="utf-8")
    print(pd.read_json(root / "train.jsonl", lines=True))

rate = 8000
t = np.arange(rate // 10) / rate # 0,1 giây
tone = (0.5 * np.sin(2 * np.pi * 440 * t) * 32767).astype(np.int16)
with wave.open(str(root / "tone.wav"), "wb") as f:
    f.setnchannels(1)
    f.setsampwidth(2) # 2 byte = int16
    f.setframerate(rate)
    f.writeframes(tone.tobytes())
with wave.open(str(root / "tone.wav"), "rb") as f:
    audio = np.frombuffer(f.readframes(f.getnframes()), dtype=np.int16)
    print(f.getframerate(), audio.shape, audio.dtype, round(float(np.abs(audio).max()) /
32768, 3))
```

KẾT QUẢ

```
   id      text  tags
0   1  Học máy  [ai]
1   2  Dữ liệu bảng  []
8000 (800,) int16 0.5
```

Cột `tags` giữ cả một list trong mỗi ô. Tập WAV ở đây có 8000 mẫu mỗi giây, nên 0,1 giây là 800 số `int16`; chia cho 32768 được biên độ trong $[-1, 1]$. Nên in tần số lấy mẫu mỗi khi đọc âm thanh: hai tệp khác tần số thì 800 mẫu không còn dài bằng nhau về thời gian. Chương 7 sẽ học kỹ dữ liệu âm thanh.

Khi đi thi

- Đọc xong thì xem ngay `shape`, `dtypes`, `isna().sum()`, `describe()` của cả train lẫn test.
- Ghép bảng với `validate`, in `shape` trước và sau khi ghép.
- Tạo file nộp từ `sample_submission`, ghép theo `id`, ghi với `index=False`, rồi đọc lại để kiểm.

Tóm tắt

- `DataFrame` là bảng có tên cột, mỗi cột một kiểu; đọc bằng `read_csv`, kiểm tra ngay `shape`, `dtypes`, `isna().sum()`, `describe()`.
- Lọc bằng điều kiện trong ngoặc nối bởi `&`, `|`, `~`; sửa ô bằng một lần `loc[mask, col]`.
- `groupby` + `agg` cho một dòng mỗi nhóm; `transform` cho một giá trị mỗi dòng gốc.
- `merge` theo khóa với `validate=` để kiểm quan hệ ghép; `left` vẫn có thể nhận dòng nếu khóa phải bị trùng.
- Chọn biểu đồ theo câu hỏi; ghi nhãn trục; trục cột bắt đầu từ 0.
- File nộp: lấy khung từ `sample_submission`, ghép theo `id`, kiểm tra, `index=False`.

Kiểm tra cuối bài

Mười câu dưới đây bao quát cả bài. Làm hết rồi xem điểm; câu nào sai thì đọc lại mục tương ứng.

CÂU 1

Vừa đọc xong `train.csv`. Việc đầu tiên nên làm là gì?

- A Xem `shape`, `dtypes`, `isna().sum()` và `describe()`
- B Huấn luyện ngay một mô hình nhỏ để có điểm so sánh
- C Xóa mọi dòng có ô trống cho bảng sạch
- D Vẽ biểu đồ đường của từng cột theo thứ tự dòng

CÂU 2

Cột `tuoi` có một ô ghi "không rõ". Sau `read_csv`, cột này có kiểu gì?

- A `int64`
- B `float64`, ô "không rõ" tự thành `NaN`
- C Kiểu chữ (object hoặc `str`)
- D `bool`

CÂU 3

Bảng có sáu dòng, cột diem kiểu float64 gồm các giá trị 5, 6, 7, 8, 9 và np.nan. `df["diem"].mean()` và `(df["diem"] >= 7).mean()` lần lượt bằng bao nhiêu?

A 5.83 và 0.6

B 7 và 0.6

C 7 và 0.5

D NaN và NaN

CÂU 4

Cần cho mọi học sinh lớp A trong df điểm 10. Dòng nào gán thẳng vào bảng gốc, không mắc bẫy gán nối chuỗi?

A `df.loc[:, "diem"] = 10`

B `df[df["lop"] == "A"]["diem"] = 10`

C `df.loc[df["lop"] == "A", "diem"] = 10`

D `df["diem"].iloc[:] = 10`

CÂU 5

Cột giờ học trên train là [1, 3, 5], trên test là [2, NaN, 100]. Ta điền ô thiếu bằng median của train. Ô NaN ở test được điền số mấy?

A 51

B 3

C 2

D 5

CÂU 6

Bảng df có một cột tên "diem toan", có dấu cách ở giữa. Ta chọn cột này thế nào?

A `df.diem toan`

B `df("diem toan")`

C `df.loc["diem toan"]`

D `df["diem toan"]`

ĐỀ THI NOAI 2026 Singapore · Vòng sơ loại · Câu 110

CÂU 7

`train` có 400 dòng. Ghép thêm thông tin lớp bằng `train.merge(info, on="lop", how="left")` thì thành 437 dòng. Nhiều khả năng là do đâu?

- A Một số lớp xuất hiện nhiều lần trong `info`
- B `how="left"` luôn thêm dòng cho khớp đủ hai bảng
- C Có học sinh không khớp lớp nào nên sinh dòng `NaN`
- D `merge` tạo thêm dòng khi sắp xếp lại theo khóa

CÂU 8

Điểm trung bình của bốn trường là 7.1, 7.2, 7.3 và 7.4. Muốn người xem thấy đúng các trường chênh nhau nhiều hay ít, ta nên vẽ thế nào?

- A Histogram của bốn điểm để xem trường nào cao nhất
- B Biểu đồ cột bắt đầu từ 0, ghi giá trị trên mỗi cột
- C Biểu đồ cột bắt đầu từ 7.0, không ghi chú gì
- D Đường nối bốn trường theo tên, đọc như xu hướng theo thời gian

CÂU 9

Hệ số tương quan giữa giờ tự học và điểm Toán là 0.36. Con số này nói lên điều gì?

- A Tự học thêm một giờ thì điểm tăng khoảng 0.36
- B Tự học nhiều là nguyên nhân làm điểm cao
- C Khoảng 36% học sinh tự học nhiều có điểm cao
- D Hai biến cùng tăng theo đường thẳng, ở mức vừa phải

CÂU 10

Đọc lại file CSV sắp nộp, ta thấy thừa một cột Unnamed: 0, còn cột id và cột dự đoán vẫn đúng. Cột thừa là index của pandas bị ghi ra. Sửa bước ghi file thế nào?

- A Gọi `reset_index()` rồi ghi như cũ
- B Ghi với `index=False`, rồi đọc lại để kiểm
- C Sắp xếp theo dự đoán trước khi ghi để bỏ index
- D Đọc file lên rồi đổi tên Unnamed: 0 thành id

Đọc thêm

- pandas — [10 minutes to pandas](#). Hướng dẫn chính thức, đi qua đúng các thao tác của bài: đọc, chọn, lọc, gộp nhóm, ghép bảng.
- McKinney — [Python for Data Analysis, 3rd edition](#). Sách miễn phí của chính tác giả pandas; chương 5, 7, 8, 10 bổ sung cho bài này.
- VanderPlas — [Python Data Science Handbook](#), chương 3 và 4. pandas và Matplotlib với nhiều ví dụ ngắn chạy được.
- Wilke — [Fundamentals of Data Visualization](#). Sách miễn phí về chọn biểu đồ cho đúng câu hỏi và tránh biểu đồ gây hiểu lầm, như cất trục ở biểu đồ cột.
- Matplotlib — [Quick start guide](#). Cách viết `fig`, `ax`, các loại biểu đồ cơ bản và cách lưu hình.
- Kaggle Learn — [Pandas](#). Khóa ngắn có bài tập thao tác bảng trong notebook để luyện thêm công cụ.
- pandas — [Merge, join, concatenate and compare](#) và [Copy-on-Write](#). Đọc quan hệ one-to-one/many-to-one và cách cập nhật bảng qua một lần loc.
- pandas — [Working with missing data](#). Đọc sự khác nhau giữa `NaN`, `pd.NA`, `count`, `size` và các phép gộp có bỏ qua giá trị thiếu.