

Лабораторная работа №32 (2 часа)

Тема работы: «Разработка, отладка и испытание программ с пересекающимися названиями столбцов»

1 Цель работы

Закрепить навык анализа данных посредством «реляционной алгебры»

2 Задание

Создайте два объекта Series с данными. Необходимо с помощью библиотеки Pandas выполнить объединение двух наборами данных следующим образом:

- ✓ left;
- ✓ right.

3 Оснащение работы

Задание по варианту, ЭВМ, среда разработки **Python 3.7, IDLE**.

4 Основные теоретические сведения

Merge в pandas. По какой колонке?

Для использования merge библиотеке pandas нужны ключевые колонки, на основе которых будет проходить объединение (в случае с примером это колонка animal). Иногда pandas не сможет распознать их автоматически, и тогда нужно указать названия колонок. Для этого нужны параметры left_on и right_on.

Например, последний merge мог бы выглядеть следующим образом:

```
zoo.merge(zoo_eats, how = 'left', left_on='animal', right_on='animal')
```

Примечание: в примере pandas автоматически нашел ключевые колонки, но часто бывает так, что этого не происходит. Поэтому о left_on и right_on не стоит забывать.

Merge в pandas — довольно сложный метод, но остальные будут намного проще.

Сортировка в pandas

Сортировка необходима. Базовый метод сортировки в pandas совсем не сложный. Функция называется sort_values() и работает она следующим образом:

```
zoo.sort_values('water_need')
```

```
In [18]: zoo.sort_values('water_need')
```

```
Out[18]:
```

	animal	uniq_id	water_need
14	zebra	1015	80
13	zebra	1014	100
8	zebra	1009	200
9	zebra	1010	220
12	zebra	1013	220
11	zebra	1012	230
10	zebra	1011	240
6	tiger	1007	290
3	tiger	1004	300
7	tiger	1008	310
4	tiger	1005	320
5	tiger	1006	330
18	lion	1019	390
19	kangaroo	1020	410
21	kangaroo	1022	410
15	lion	1016	420
20	kangaroo	1021	430
17	lion	1018	500
0	elephant	1001	500
2	elephant	1003	550
16	lion	1017	600
1	elephant	1002	600

Примечание: в прошлых версиях pandas была функция `sort()`, работающая подобным образом. Но в новых версиях ее заменили на `sort_values()`, поэтому пользоваться нужно именно новым вариантом.

Единственный используемый параметр — название колонки, `water_need` в этом случае. Довольно часто приходится сортировать на основе нескольких колонок. В таком случае для них нужно использовать ключевое слово `by`:

```
zoo.sort_values(by=['animal', 'water_need'])
```

```
In [22]: zoo.sort_values(by = ['animal', 'water_need'])
```

```
Out[22]:
```

	animal	uniq_id	water_need
0	elephant	1001	500
2	elephant	1003	550
1	elephant	1002	600
19	kangaroo	1020	410
21	kangaroo	1022	410
20	kangaroo	1021	430
18	lion	1019	390
15	lion	1016	420
17	lion	1018	500
16	lion	1017	600
6	tiger	1007	290
3	tiger	1004	300
7	tiger	1008	310
4	tiger	1005	320
5	tiger	1006	330
14	zebra	1015	80
13	zebra	1014	100
8	zebra	1009	200
9	zebra	1010	220
12	zebra	1013	220
11	zebra	1012	230
10	zebra	1011	240

Примечание: ключевое слово `by` можно использовать и для одной колонки `zoo.sort_values(by = ['water_need'])`.

`sort_values` сортирует в порядке возрастания, но это можно поменять на убывание:

```
zoo.sort_values(by=['water_need'], ascending=False)
```

```
In [25]: zoo.sort_values(by = ['water_need'], ascending = False)
```

```
Out[25]:
```

	animal	uniq_id	water_need
1	elephant	1002	600
16	lion	1017	600
2	elephant	1003	550
0	elephant	1001	500
17	lion	1018	500
20	kangaroo	1021	430
15	lion	1016	420
19	kangaroo	1020	410
21	kangaroo	1022	410
18	lion	1019	390
5	tiger	1006	330
4	tiger	1005	320
7	tiger	1008	310
3	tiger	1004	300
6	tiger	1007	290
10	zebra	1011	240
11	zebra	1012	230
9	zebra	1010	220
12	zebra	1013	220
8	zebra	1009	200
13	zebra	1014	100
14	zebra	1015	80

reset_index()

Заметили ли вы, какой беспорядок теперь в нумерации после последней сортировки?

```
In [25]: zoo.sort_values(by = ['water_need'], ascending = False)
```

```
Out[25]:
```

	animal	uniq_id	water_need
1	elephant	1002	600
16	lion	1017	600
2	elephant	1003	550
0	elephant	1001	500
17	lion	1018	500
20	kangaroo	1021	430
15	lion	1016	420
19	kangaroo	1020	410
21	kangaroo	1022	410
18	lion	1019	390
5	tiger	1006	330
4	tiger	1005	320
7	tiger	1008	310
3	tiger	1004	300
6	tiger	1007	290
10	zebra	1011	240
11	zebra	1012	230
9	zebra	1010	220
12	zebra	1013	220
8	zebra	1009	200
13	zebra	1014	100
14	zebra	1015	80

Это не просто выглядит некрасиво... неправильная индексация может испортить визуализации или повлиять на то, как работают модели машинного обучения.

В случае изменения DataFrame нужно переиндексировать строки. Для этого можно использовать метод `reset_index()`. Например:

```
zoo.sort_values(by=['water_need'], ascending=False).reset_index()
```

```
In [18]: zoo.sort_values(by = ['water_need'], ascending = False).reset_index()
```

```
Out[18]:
```

	index	animal	uniq_id	water_need
0	1	elephant	1002	600
1	16	lion	1017	600
2	2	elephant	1003	550
3	0	elephant	1001	500
4	17	lion	1018	500
5	20	kangaroo	1021	430
6	15	lion	1016	420
7	19	kangaroo	1020	410
8	21	kangaroo	1022	410
9	18	lion	1019	390
10	5	tiger	1006	330
11	4	tiger	1005	320
12	7	tiger	1008	310
13	3	tiger	1004	300
14	6	tiger	1007	290
15	10	zebra	1011	240
16	11	zebra	1012	230
17	9	zebra	1010	220
18	12	zebra	1013	220
19	8	zebra	1009	200
20	13	zebra	1014	100
21	14	zebra	1015	80

Можно заметить, что новый DataFrame также хранит старые индексы. Если они не нужны, их можно удалить с помощью параметра `drop=True` в функции:

```
zoo.sort_values(by = ['water_need'], ascending = False).reset_index(drop = True)
```

```
In [19]: zoo.sort_values(by = ['water_need'], ascending = False).reset_index(drop = True)
```

```
Out[19]:
```

	animal	uniq_id	water_need
0	elephant	1002	600
1	lion	1017	600
2	elephant	1003	550
3	elephant	1001	500
4	lion	1018	500
5	kangaroo	1021	430
6	lion	1016	420
7	kangaroo	1020	410
8	kangaroo	1022	410
9	lion	1019	390
10	tiger	1006	330
11	tiger	1005	320
12	tiger	1008	310
13	tiger	1004	300
14	tiger	1007	290
15	zebra	1011	240
16	zebra	1012	230
17	zebra	1010	220
18	zebra	1013	220
19	zebra	1009	200
20	zebra	1014	100
21	zebra	1015	80

Fillna

Примечание: `fillna` — это слова `fill`(заполнить) и `na`(не доступно).

Запустим еще раз метод `left-merge`:

```
zoo.merge(zoo_eats, how='left')
```

Это все животные. Проблема только в том, что для львов есть значение `NaN`. Само по себе это значение может отвлекать, поэтому лучше заменять его на что-то более осмысленное. Иногда это может быть `0`, в других случаях — строка. Но в этот раз обойдемся `unknown`. Функция `fillna()` автоматически найдет и заменит все значения `NaN` в `DataFrame`:

```
zoo.merge(zoo_eats, how='left').fillna('unknown')
```

```
In [27]: zoo.merge(zoo_eats, how = 'left').fillna('unknown')
```

Out[27]:

	animal	uniq_id	water_need	food
0	elephant	1001	500	vegetables
1	elephant	1002	600	vegetables
2	elephant	1003	550	vegetables
3	tiger	1004	300	meat
4	tiger	1005	320	meat
5	tiger	1006	330	meat
6	tiger	1007	290	meat
7	tiger	1008	310	meat
8	zebra	1009	200	meat
9	zebra	1010	220	meat
10	zebra	1011	240	meat
11	zebra	1012	230	meat
12	zebra	1013	220	meat
13	zebra	1014	100	meat
14	zebra	1015	80	meat
15	lion	1016	420	unknown
16	lion	1017	600	unknown
17	lion	1018	500	unknown
18	lion	1019	390	unknown
19	kangaroo	1020	410	vegetables
20	kangaroo	1021	430	vegetables
21	kangaroo	1022	410	vegetables

Примечание: зная, что львы едят мясо, можно было также написать `zoo.merge(zoo_eats, how='left').fillna('meat')`.

Проверьте себя

Вернемся к набору данных `article_read`.

Примечание: в этом наборе хранятся данные из блога о путешествиях. Загрузить его можно здесь. Или пройти весь процесс загрузки, открытия и установки из первой части руководства `pandas`.

Скачайте еще один набор данных: `blog_buy`. Это можно сделать с помощью следующих двух строк в Jupyter Notebook:

```
!wget https://pythonru.com/downloads/pandas_tutorial_buy.csv
```

```
blog_buy = pd.read_csv('pandas_tutorial_buy.csv', delimiter=';',  
names=['my_date_time', 'event', 'user_id', 'amount'])
```

Набор `article_read` показывает всех пользователей, которые читают блог, а `blog_buy` — тех, купил что-то в этом блоге за период с 2018-01-01 по 2018-01-07.

Два вопроса:

Какой средний доход в период с 2018-01-01 по 2018-01-07 от пользователей из `article_read`?

Выведите топ-3 страны по общему уровню дохода за период с 2018-01-01 по 2018-01-07. (Пользователей из `article_read` здесь тоже нужно использовать).

Решение задания №1

Средний доход — 1,0852

Для вычисления использовался следующий код:

```
step_1 = article_read.merge(blog_buy, how='left', left_on='user_id',  
right_on='user_id')  
step_2=step_1.amount  
step_3=step_2.fillna(0)  
result=step_3.mean()  
result
```

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

```
In [2]: article_read = pd.read_csv('pandas_tutorial_read.csv', delimiter=';', names = ['my_datetime', 'event', 'user_id', 'amount'])  
blog_buy = pd.read_csv('pandas_tutorial_buy.csv', delimiter=';', names = ['my_date_time', 'event', 'user_id', 'amount'])
```

TASK #1

```
In [20]: step_1 = article_read.merge(blog_buy, how = 'left', left_on = 'user_id', right_on = 'user_id')  
step_2 = step_1.amount  
step_3 = step_2.fillna(0)  
result = step_3.mean()  
result
```

```
Out[20]: 1.0852367688022284
```

Примечание: шаги использовались, чтобы внести ясность. Описанные функции можно записать и в одну строку.

Краткое объяснение:

На скриншоте также есть две строки с импортом `pandas` и `numpy`, а также чтением файлов `csv` в Jupyter Notebook.

На шаге №1 объединены две таблицы (`article_read` и `blog_buy`) на основе колонки `user_id`. В таблице `article_read` хранятся все пользователи, даже если они ничего не покупают, потому что ноли (0) также должны учитываться при подсчете среднего дохода. Из таблицы удалены те, кто покупали, но кого нет в наборе `article_read`. Все вместе привело к `left-merge`.

Шаг №2 — удаление ненужных колонок с сохранением только `amount`.

На шаге №3 все значения `NaN` заменены на 0.

В конце концов проводится подсчет с помощью `.mean()`.

Решение задания №2

```
step_1 = article_read.merge(blog_buy, how = 'left', left_on = 'user_id', right_on = 'user_id')
step_2 = step_1.fillna(0)
step_3 = step_2.groupby('country').sum()
step_4 = step_3.amount
step_5 = step_4.sort_values(ascending = False)
step_5.head(3)
```

```
In [18]: import numpy as np
import pandas as pd
```

```
In [19]: article_read = pd.read_csv('pandas_tutorial_read.csv', delimiter=';', names = ['my_datetime', 'country', 'amount'])
blog_buy = pd.read_csv('pandas_tutorial_buy.csv', delimiter=';', names = ['my_date_time', 'event', 'country', 'amount'])
```

TASK #2

```
In [17]: step_1 = article_read.merge(blog_buy, how = 'left', left_on = 'user_id', right_on = 'user_id')
step_2 = step_1.fillna(0)
step_3 = step_2.groupby('country').sum()
step_4 = step_3.amount
step_5 = step_4.sort_values(ascending = False)
step_5.head(3)
```

```
Out[17]: country
country_4    1112.0
country_5     324.0
country_2     296.0
Name: amount, dtype: float64
```

Найдите топ-3 страны на скриншоте.

Краткое объяснение:

Тот же метод merge, что и в первом задании.

Замена всех NaN на 0.

Суммирование всех числовых значений по странам.

Удаление всех колонок кроме amount.

Сортировка результатов в убывающем порядке так, чтобы можно было видеть топ.

Вывод только первых 3 строк.

5 Порядок выполнения работы

1. Выделить ключевые моменты задачи.
2. Построить алгоритм решения задачи.
3. Запрограммировать полученный алгоритм.
4. Провести тестирование полученной программы.

6 Форма отчета о работе

Лабораторная работа № ____

Номер учебной группы _____
Фамилия, инициалы учащегося: _____
Дата выполнения работы: _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____

7 Контрольные вопросы и задания

1. Что такое объединение наборов данных?
2. Бывает ли пересечение наборов данных?
3. Для чего применяется объединение данных?

8 Рекомендуемая литература

Плас, Дж. В. Python для сложных задач. Наука о данных и машинное обучение / Дж.В. Плас. – СПб: Питер, 2018.

Прохоренок, Н.А. Python 3. Самое необходимое / Н.А Прохоренок, В.А. Дронов – СПб.: БВХ-Петербург, 2016.