Сводные таблицы

https://dfedorov.spb.ru

Сводная таблица (**pivot table)** — схожая операция, часто встречающаяся в электронных таблицах и других программах, работающих с табличными данными.
Сводная таблица получает на входе простые данные в виде столбцов и группирует записи в двумерную таблицу, обеспечивающую многомерное представление данных.
Различие между сводными таблицами и операцией GroupBy иногда неочевидно.
Вы выполняете операцию «разбить, применить, объединить», но как разбиение, так и объединение происходят не на одномерном индексе, а на двумерной координатной сетке.

Для примеров из этого раздела мы воспользуемся базой данных пассажиров парохода «Титаник», доступной через библиотеку Seaborn:

```
>>> import numpy as np
>>> import pandas as pd
>>> import seaborn as sns
>>> titanic = sns.load_dataset('titanic')
>>> titanic.head()
```

Этот набор данных содержит информацию о каждом пассажире злополучного рейса, включая пол, возраст, класс, стоимость билета и многое другое.

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Сводные таблицы «вручную»

Чтобы узнать о данных больше, можно начать с группировки пассажиров по полу, информации о том, выжил ли пассажир, или какой-то их комбинации.
Например, посмотрим на коэффициент выживаемости в зависимости от пола:

```
>>> titanic.groupby('sex')[['survived']].mean()
```

sex	survived
female	0.742038
male	0.188908

Это сразу же дает нам некоторое представление о наборе данных: в целом, три четверти находившихся на борту женщин выжило, в то время как из мужчин выжил только каждый пятый!

Однако хотелось бы заглянуть немного глубже и увидеть распределение выживших по полу и классу: сгруппировать по классу и полу, выбрать выживших, применить агрегирующую функцию среднего значения, объединить получившиеся группы, после чего выполнить операцию unstack иерархического индекса.

```
>>> titanic.groupby(['sex', 'class'])['survived'].aggregate('mean').unstack()
```

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

Это дает нам лучшее представление о том, как пол и класс влияли на выживаемость, но код начинает выглядеть несколько запутанным. Хотя каждый шаг этого конвейера представляется вполне осмысленным в свете ранее рассмотренных инструментов, такая длинная строка кода не особо удобна для чтения или использования. Двумерный GroupBy встречается настолько часто, что в состав библиотеки Pandas был включен удобный метод, pivot_table, позволяющий описывать более кратко данную разновидность многомерного агрегирования.

Синтаксис сводных таблиц

```
>>> titanic.pivot_table('survived', index='sex', columns='class')
```

class	First	Second	Third
sex			
female	0.968085	0.921053	0.500000
male	0.368852	0.157407	0.135447

Такая запись несравненно удобнее для чтения, чем подход с GroupBy , при том же результате.

Как и можно было ожидать от трансатлантического круиза начала XX века, судьба благоприятствовала женщинам и первому классу. Женщины из первого класса выжили практически все (привет, Роуз!), из мужчин третьего класса выжила только десятая часть (извини, Джек!).

Многоуровневые сводные таблицы

Группировку в сводных таблицах, как и при операции GroupBy , можно задавать на нескольких уровнях и с множеством параметров. Например, интересно взглянуть на возраст в качестве третьего измерения. Разобьем данные на интервалы по возрасту с помощью функции pd.cut:

```
>>> age = pd.cut(titanic['age'], [0, 18, 80])
>>> age

0      (18.0, 80.0]
1      (18.0, 80.0]
2      (18.0, 80.0]
3      (18.0, 80.0]
4      (18.0, 80.0]
           ...
886    (18.0, 80.0]
887    (18.0, 80.0]
888             NaN
889    (18.0, 80.0]
890    (18.0, 80.0]
Name: age, Length: 891, dtype: category
Categories (2, interval[int64]): [(0, 18] < (18, 80]]
```

```
>>> titanic.pivot_table('survived', ['sex', age], 'class')
```

	fare	(-0.001, 14.454]			(14.454, 512.329]		
	class	First	Second	Third	First	Second	Third
sex	age						
female	(0, 18]	NaN	1.000000	0.714286	0.909091	1.000000	0.318182
	(18, 80]	NaN	0.880000	0.444444	0.972973	0.914286	0.391304
male	(0, 18]	NaN	0.000000	0.260870	0.800000	0.818182	0.178571
	(18, 80]	0.0	0.098039	0.125000	0.391304	0.030303	0.192308

Дополнительные параметры сводных таблиц

pandas.pivot_table

```
>>> titanic.pivot_table(index='sex', columns='class', aggfunc={'survived':sum, 'fare':'mean'})
```

	fare			survived		
class	First	Second	Third	First	Second	Third
sex						
female	106.125798	21.970121	16.118810	91	70	72
male	67.226127	19.741782	12.661633	45	17	47

Ключевое слово aggfunc управляет тем, какой тип агрегирования применяется, по умолчанию это среднее значение.

Как и в GroupBy , спецификация агрегирующей функции может быть строкой с одним из нескольких обычных вариантов ('sum' , 'mean' , 'count' , 'min' , 'max' и т. д.) или функцией, реализующей агрегирование (np.sum() , min() , sum() и т. п.). Кроме того, агрегирование может быть задано в виде словаря, связывающего столбец с любым из вышеперечисленных вариантов.

Иногда бывает полезно вычислять итоги по каждой группе. Это можно сделать с помощью ключевого слова margins :

```
>>> titanic.pivot_table('survived', index='sex', columns='class', margins=True)
```

class	First	Second	Third	All
sex				
female	0.968085	0.921053	0.500000	0.742038
male	0.368852	0.157407	0.135447	0.188908
All	0.629630	0.472826	0.242363	0.383838

Такие итоги автоматически дают нам информацию о выживаемости вне зависимости от класса, коэффициенте выживаемости по классу вне зависимости от пола и общем коэффициенте выживаемости 38 %. Метки для этих итогов можно задать с помощью ключевого слова margins_name , по умолчанию имеющего значение "All" .