SkillFactory

4. Сводные таблицы

✍ Сводные таблицы — это распространённый инструмент для агрегации данных.

→ Сводная таблица принимает на вход данные из отдельных столбцов и группирует их. В результате получается новая таблица, которая позволяет увидеть многомерное обобщение данных. Таким образом, благодаря сводным таблицам мы можем оценить зависимость между двумя и более признаками данных.

Мы чаще сталкиваемся со сводными таблицами, чем с обычными, **в плоском виде**, так как сводные таблицы удобнее для анализа и быстрых выводов, а также позволяют увидеть более общие зависимости между признаками, нежели простая группировка данных.

Под «плоским видом» подразумевается, что индексами являются номера строк, а столбцами — имена столбцов.

Инструмент сводных таблиц также широко популярен среди тех, кто использует *Excel* или какие-либо *BI*-системы.

МЕТОД GROUPBY КАК СПОСОБ ПОСТРОЕНИЯ СВОДНЫХ ТАБЛИЦ

На самом деле мы с вами уже строили простейшие одномерные сводные

таблицы с помощью метода groupby — мы рассматривали сводную таблицу в контексте группировки по одному признаку.

Например, мы уже умеем строить таблицу, которая показывает зависимость медианной цены и площади здания от числа комнат:

```
melb_df.groupby('Rooms')[['Price', 'BuildingArea']].median()
```

Rooms	Price	BuildingArea
1	385000.0	107.0
2	690000.0	126.0
3	950000.0	126.0
4	1285000.0	142.0
5	1660000.0	176.0
6	1800000.0	126.0
7	1496000.0	216.5
8	1515000.0	126.0
10	900000.0	126.0

Также можно построить таблицу, в которой мы будем учитывать не только число комнат, но и тип здания (Type). Для этого в параметрах метода groupby() укажем список из нескольких интересующих нас столбцов.

```
melb_df.groupby(['Rooms', 'Type'])['Price'].mean()
```

```
Rooms  Type
1      house        8.668655e+05
       townhouse    5.927045e+05
       unit         3.899289e+05
2      house        1.017238e+06
       townhouse    7.101585e+05
       unit         6.104905e+05
3      house        1.109233e+06
       townhouse    9.847087e+05
       unit         8.505963e+05
4      house        1.462283e+06
       townhouse    1.217092e+06
       unit         1.037476e+06
5      house        1.877327e+06
       townhouse    1.035000e+06
       unit                 NaN
6      house        1.869508e+06
       townhouse            NaN
       unit         5.200000e+05
7      house        1.920700e+06
       townhouse            NaN
       unit                 NaN
8      house        1.510286e+06
       townhouse            NaN
       unit         2.250000e+06
10     house        9.000000e+05
       townhouse            NaN
       unit                 NaN
Name: Price, dtype: float64
```

В результате выполнения такого кода мы получаем *Series*, которая обладает несколькими уровнями индексов: первый уровень — число комнат, второй уровень — тип здания. Такая организация индексов называется **иерархической**. Вычисление параметра (средней цены) происходит во всех возможных комбинациях признаков.

Для того, чтобы финальный результат был представлен в виде сводной таблицы (первый группировочный признак по строкам, а второй — по столбцам), а не в виде *Series* с иерархическими индексами, к результату чаще всего применяют метод unstack(), который позволяет переопределить вложенный индекс в виде столбцов таблицы:

```
melb_df.groupby(['Rooms', 'Type'])['Price'].mean().unstack()
```

Type	house	townhouse	unit
Rooms			
1	8.668655e+05	5.927045e+05	3.899289e+05
2	1.017238e+06	7.101585e+05	6.104905e+05
3	1.109233e+06	9.847087e+05	8.505963e+05
4	1.462283e+06	1.217092e+06	1.037476e+06
5	1.877327e+06	1.035000e+06	NaN
6	1.869508e+06	NaN	5.200000e+05
7	1.920700e+06	NaN	NaN
8	1.510286e+06	NaN	2.250000e+06
10	9.000000e+05	NaN	NaN

В результате мы получаем сводную таблицу, столбцы в которой представляют типы домов (*house*, *townhouse*, *unit*), строки — число комнат, а на пересечении строк и столбцов находится средняя стоимость объекта с такими показателями.

? Какие интересные выводы можно сделать из этой таблицы?

1 Пропуски в сводной таблице (*NaN*) говорят о том, что в наших данных нет соответствующих комбинаций признаков. Например, у нас нет информации о ценах на таунхаусы, где количество комнат больше пяти.

2 Наибольшей средней стоимостью (2,25 млн. австралийских долларов) обладают объекты типа *unit* с восемью жилыми комнатами. Наименьшая средняя стоимость — у однокомнатных домов типа *unit* (чуть меньше 400 тыс. австралийских долларов).

3 Сколько бы комнат ни было в доме, цена на объекты типа *unit* всегда ниже других (за исключением восьмикомнатных объектов).

МЕТОД PIVOT_TABLE ДЛЯ ПОСТРОЕНИЯ СВОДНЫХ ТАБЛИЦ

На самом деле метод groupby редко используется при двух параметрах, так как для построения сводных таблиц существует специальный и более простой метод — pivot_table().

Кликните на плашку, чтобы увидеть информацию ↓

Основные параметры метода pivot_table()

- values — имя столбца, по которому необходимо получить сводные данные, применяя агрегирующую функцию;
- index — имя столбца, значения которого станут строками сводной таблицы;
- columns — имя столбца, значения которого станут столбцами сводной таблицы;
- aggfunc — имя или список имён агрегирующих функций (по умолчанию — подсчёт среднего, 'mean');
- fill_value — значение, которым необходимо заполнить пропуски (по умолчанию пропуски не заполняются).

Давайте построим ту же самую таблицу, но уже с использованием метода pivot_table. В качестве параметра values укажем столбец *Price*, в качестве индексов сводной таблицы возьмём *Rooms*, а в качестве столбцов — *Type*. Агрегирующую функцию оставим по умолчанию (среднее). Дополнительно заменим пропуски в таблице на значение 0. Финальный результат для наглядности вывода округлим с помощью метода round() до целых.

```
melb_df.pivot_table(
  values='Price',
  index='Rooms',
  columns='Type',
  fill_value=0
).round()
```

Type	house	townhouse	unit
Rooms			
1	866866.0	592705.0	389929.0
2	1017238.0	710158.0	610491.0
3	1109233.0	984709.0	850596.0
4	1462283.0	1217092.0	1037476.0
5	1877327.0	1035000.0	0.0
6	1869508.0	0.0	520000.0
7	1920700.0	0.0	0.0
8	1510286.0	0.0	2250000.0
10	900000.0	0.0	0.0

Несложно понять, что метод pivot_table() имеет преимущество перед группировкой по нескольким критериям. Оно заключается в наличии специальных аргументов для строк и столбцов сводной таблицы, благодаря чему уменьшается вероятность запутаться при построении более сложных (многомерных) сводных таблиц, о которых мы поговорим далее.

А теперь давайте проанализируем продажи в каждом из регионов в зависимости от того, будний был день или выходной. Для этого построим сводную таблицу, в которой строками будут являться названия регионов (*Regionname*), а в столбцах будет располагаться наш «признак-мигалка» выходного дня (*Weekend*), который равен 1, если день был выходным, и 0 — в противном случае. В качестве значений сводной таблицы возьмём количество продаж.

```
melb_df.pivot_table(
    values='Price',
    index='Regionname',
    columns='Weekend',
    aggfunc='count',
)
```

Weekend	0	1
Regionname		
Eastern Metropolitan	447	1024
Eastern Victoria	13	40
Northern Metropolitan	1258	2632
Northern Victoria	11	30
South-Eastern Metropolitan	123	327
Southern Metropolitan	1534	3161
Western Metropolitan	960	1988
Western Victoria	8	24

Из результирующей таблицы можно сделать два вывода:

1. Число продаж резко возрастает в выходные вне зависимости от региона (приблизительно в 2-3 раза). То есть вероятность того, что дом продадут в выходные, гораздо выше вероятности, что его продадут в будний день.

2. В отдалённых регионах (*Victoria*) коэффициент роста числа продаж выше, чем в центральных. Если в центральных регионах *Metropolitan* продажи по выходным в 2-2.5 раза выше, чем по будням, то в регионах *Victoria* число продаж в выходные вырастает примерно в 3 раза.

   Такой рост можно даже попытаться объяснить логически: в выходные дни у людей появляется свободное время, чтобы доехать до отдалённых пригородов с целью покупки дома.

Разберём ещё один пример: найдём, как зависит средняя и медианная площадь участка (*Landsize*) от типа объекта (*Type*) и его региона (*Regionname*). Чтобы посмотреть несколько статистических параметров, нужно передать в аргумент aggfunc список из агрегирующих функций. Построим такую сводную таблицу, где пропущенные значения заменим на 0:

```
melb_df.pivot_table(
    values='Landsize',
    index='Regionname',
    columns='Type',
    aggfunc=['median', 'mean'],
    fill_value=0
)
```

	median			mean		
Type	house	townhouse	unit	house	townhouse	unit
Regionname						
Eastern Metropolitan	674.0	233.5	203	717.422847	269.440678	330.444444
Eastern Victoria	843.0	0.0	230	3108.960000	0.000000	295.333333
Northern Metropolitan	459.5	134.0	0	619.249092	317.325733	495.026538
Northern Victoria	724.0	0.0	0	3355.463415	0.000000	0.000000
South-Eastern Metropolitan	630.5	240.0	199	664.306701	212.160000	357.864865
Southern Metropolitan	586.0	246.0	0	569.643881	278.858824	466.380245
Western Metropolitan	531.0	198.0	62	507.883406	244.560669	557.637232
Western Victoria	599.5	0.0	0	655.500000	0.000000	0.000000

Обратите внимание на добавление дополнительных индексов столбцов *median* и *mean*. Здесь медианное и среднее значения рассчитаны отдельно для каждой комбинации признаков.

Здесь в глаза бросаются объекты типа *house* в регионах *Eastern Victoria* и *Northern Victoria* — в них среднее и медиана отличаются более чем в три раза. Вероятно, это связано с тем, что в этих районах очень большой разброс цен: есть несколько объектов с гигантской площадью, а остальные объекты имеют небольшую площадь. Из-за этого среднее значение искажается, в то время как медиана нечувствительна к такому разбросу и не искажает результат.

МНОГОМЕРНЫЕ СВОДНЫЕ ТАБЛИЦЫ

До этого мы рассматривали, как некоторый статистический показатель может зависеть от двух признаков. Однако, как уже упоминалось, сводные таблицы позволяют наблюдать зависимость и от большего числа признаков. Такие сводные таблицы называются **многомерными**.

Для того чтобы исследовать зависимость от большего числа признаков, можно передать список признаков в параметр index или параметр columns.

Давайте построим таблицу, в которой по индексам будут располагаться признаки метода продажи (*Method*) и типа объекта (*Type*), по столбцам — наименование региона (*Regionname*), а на пересечении строк и столбцов будет стоять медианная цена объекта (*Price*):

```
melb_df.pivot_table(
    values='Price',
    index=['Method','Type'],
    columns='Regionname',
    aggfunc='median',
    fill_value=0
)
```

	Regionname	Eastern Metropolitan	Eastern Victoria	Northern Metropolitan	Northern Victoria	South-Eastern Metropolitan	Southern Metropolitan	Western Metropolitan	Western Victoria
Method	Type								
PI	house	1244000	780000	900000	500000	865000	1725000	870000	630000
	townhouse	760000	0	632500	0	1190000	1055000	670000	0
	unit	650000	0	410000	0	525000	571250	360000	0
S	house	1127000	675000	920000	555000	883300	1611000	870000	397500
	townhouse	828000	0	750000	0	875000	1135000	729000	0
	unit	645750	492000	525500	0	606000	655000	489000	0
SA	house	932500	950000	817500	540000	880000	1390000	772500	0
	townhouse	807500	0	425000	0	0	1141000	467500	0
	unit	0	0	616000	0	0	580000	571000	0
SP	house	1050000	672500	900000	521000	770000	1521750	865000	360000
	townhouse	910000	0	690000	0	800000	1162500	702500	0
	unit	515000	400000	470000	0	601000	550000	460000	0
VB	house	1100000	712500	1050000	690000	850000	1800000	880000	0
	townhouse	892500	0	640000	0	0	1250000	689500	0
	unit	500000	0	450000	0	700000	500000	420000	0

Первым индексом в таблице идёт метод продажи здания, далее для метода указывается тип недвижимости. По столбцам расположены регионы. В ячейках таблицы указана медианная цена для каждой такой комбинации.

Такие таблицы уже сложнее читать, однако с помощью них можно более глубоко исследовать закономерности. Например, можно видеть, что вне зависимости от метода продажи и региона цена на объекты типа *house* практически всегда выше, чем на объекты другого типа.

ДОСТУП К ДАННЫМ В СВОДНОЙ ТАБЛИЦЕ

Как получить доступ к данным или произвести фильтрацию в сложной сводной

? таблице, где есть дополнительные индексы?

Давайте рассмотрим, что собой представляют столбцы сложной сводной таблицы.

Запишем сводную таблицу, которую мы создавали ранее в переменную pivot:

```
pivot = melb_df.pivot_table(
  values='Landsize',
  index='Regionname',
  columns='Type',
  aggfunc=['median', 'mean'],
  fill_value=0
)
```

Выведем её столбцы с помощью атрибута columns:

```
pivot.columns
```

```
MultiIndex([('median',     'house'),
            ('median', 'townhouse'),
            ('median',      'unit'),
            (  'mean',     'house'),
            (  'mean', 'townhouse'),
            (  'mean',      'unit')],
           names=[None, 'Type'])
```

В результате мы получаем объект *MultiIndex*. Этот объект хранит в себе шесть комбинаций пар столбцов (два статистических параметра и три типа здания), то есть есть шесть возможных вариантов обращения к столбцам таблицы.

Мультииндексы раскрываются подобно вложенным словарям — по очереди, как матрёшка. Чтобы получить доступ к определённому столбцу, вы должны сначала обратиться к столбцу, который находится уровнем выше.

Так, из таблицы pivot мы можем получить средние значения площадей участков для типа здания *unit*, просто последовательно обратившись по имени столбц

```
display(pivot['mean']['unit'])
```

```
Regionname
Eastern Metropolitan          330.444444
Eastern Victoria              295.333333
Northern Metropolitan         495.026538
Northern Victoria               0.000000
South-Eastern Metropolitan    357.864865
Southern Metropolitan         466.380245
Western Metropolitan          557.637232
Western Victoria                0.000000
Name: unit, dtype: float64
```

Аналогично производится и фильтрация данных. Например, если нам нужны регионы, в которых средняя площадь здания для домов типа *house* меньше их медианной площади, то мы можем найти их следующим образом:

```
mask = pivot['mean']['house'] < pivot['median']['house']
filtered_pivot = pivot[mask]
display(filtered_pivot)
```

	median			mean		
Type	house	townhouse	unit	house	townhouse	unit
Regionname						
Southern Metropolitan	586.0	246.0	0	569.643881	278.858824	466.380245
Western Metropolitan	531.0	198.0	62	507.883406	244.560669	557.637232

Чтобы получить индексы отфильтрованной таблицы, можно воспользоваться атрибутом index и обернуть результат в список:

```
print(list(filtered_pivot.index))
# ['Southern Metropolitan', 'Western Metropolitan']
```

✍ Таким образом, сводные таблицы изначально кажутся сложной структурой, но на самом деле это обычные *DataFrame* со вложенными индексами строк или столбцов.

Умение читать и анализировать сложные сводные таблицы — это важный навык, который помогает проводить углублённый анализ данных.

Примечание. На самом деле мультииндексные таблицы можно создавать

вручную. Давайте посмотрим на синтаксис данной конструкции:

```
import numpy as np
mser = pd.Series(
   np.random.rand(8),
      index=[['white','white','white','blue','blue','red','red','red'],
        ['up','down','right','up','down','up','down','left']])
display(mser)
```

```
white  up       0.708366
       down     0.364817
       right    0.615734
blue   up       0.766448
       down     0.734864
red    up       0.148334
       down     0.317675
       left     0.646968
dtype: float64
```

В данном примере мы создаём объект *Series* со вложенными индексами. Мы передаём в качестве индексов *Series* вложенный список, где первый список задаёт внешний уровень вложенности, а второй список — внутренний уровень вложенности. Значения *Series* — случайные числа от 0 до 1, сгенерированные функцией np.random.rand() (ваши значения могут отличаться).

Если посмотреть на индексы *Series*, можно увидеть, что они являются мультииндексами:

```
print(mser.index)
```

```
MultiIndex([('white',    'up'),
            ('white',  'down'),
            ('white', 'right'),
            ( 'blue',    'up'),
            ( 'blue',  'down'),
            (  'red',    'up'),
            (  'red',  'down'),
            (  'red',  'left')],
           )
```

Аналогично создаются *DataFrame* со вложенными признаками (вложенными столбцами) — для этого вложенный список передаётся в параметр columns при инициализации таблицы:

```
mframe = pd.DataFrame(
    np.random.randn(16).reshape(4,4),
    index=[['white','white','red','red'], ['up','down','up','down']],
    columns=[['pen','pen','paper','paper'],[1,2,1,2]]
)
display(mframe)
```

		pen		paper	
		1	2	1	2
white	up	-1.505083	-0.470798	-0.694832	0.791902
	down	1.186443	0.050301	1.766683	-0.419802
red	up	-0.714390	1.550414	-0.387972	-0.455019
	down	1.275397	-0.621602	-1.089852	0.442110

Давайте немного **потренируемся в составлении и чтении сводных таблиц** ↓

Задание 4.1

1 point possible (graded)

Какой параметр метода `pivot_table()` отвечает за признак, по которому будут рассчитаны агрегирующие функции?

- values
- index
- aggfunc
- columns

Отправить

Задание 4.2

1 point possible (graded)

Составьте сводную таблицу, которая показывает зависимость медианной площади (*BuildingArea*) здания от типа объекта недвижимости (*Type*) и количества жилых комнат в доме (*Rooms*). **Для какой комбинации признаков площадь здания наибольшая?**

В качестве ответа запишите эту комбинацию (тип здания, число комнат) **через запятую, без пробелов**.

Отправить

Задание 4.3

1 point possible (graded)

Составьте сводную таблицу, которая показывает зависимость средней цены объекта недвижимости (*Price*) от риелторского агентства (*SellerG*) и типа здания (*Type*).

Во вновь созданной таблице найдите агентство, у которого средняя цена для зданий типа *unit* максимальна. В качестве ответа **запишите название этого агентства**.

Отправить