**Функции и работа с данными**

Темы на сегодня:

* Функции в Pandas
* Группировки в Pandas

ФУНКЦИИ В PANDAS

Часто при решении аналитических задач не хватает встроенных методов. По этой причине требуется писать собственные функции. Способ написания самой функции ничем не отличается от стандартного через ключевое слово **def**, либо через ключевое слово **lambda**. В параметрах функции нужно указать данные, которые нужно обработать, а также обязательно указывать return, иначе функция вернёт None.

**Пример:**

*import pandas as pd*

*df = pd.DataFrame({'user\_id': [1, 2, 3], 'clicks': [163, 130, 97], 'orders': [2, 4, 0]})*

*user\_id clicks orders*

*0 1 163 2*

*1 2 130 4*

*2 3 97 0*

*def func\_1(parameter):*

*"""Пример функции"""*

*return parameter == 0*

Вопрос в том, как применить функцию к датафрейму. Для этого существует специальный метод **.apply(функция)**.

**Пример:**

*df['bool'] = df['orders'].apply(func\_1)*

*df[['user\_id', 'clicks', 'orders', 'bool']]*

*user\_id clicks orders bool*

*0 1 163 2 False*

*1 2 130 4 False*

*2 3 97 0 True*

Что происходит? Мы обращаемся к столбцу “orders”. Каждое значение этого столбца помещается в функцию, которая указана в методе .apply. Функция возвращает значение, которое помещается в очередную ячейку столбца “bool”. Если необходимо передать иные параметры, то их нужно указать дополнительно в .apply.

**Пример:**

*df['val'] = df['orders'].apply(lambda x, n: x\*\*n, n = 2)*

*df[['user\_id', 'clicks', 'orders', 'val']]*

*user\_id clicks orders val*

*0 1 163 2 4*

*1 2 130 4 16*

*2 3 97 0 0*

Однако, такой подход применим, если задействуются значения лишь одного столбца. Что, если задействуются несколько столбцов?

**Пример:**

*df = pd.DataFrame({'user\_id': [1, 2, 3], 'clicks': [163, 130, 97], 'orders': [2, 4, 0]})*

*def func\_2(data):*

*return data[‘orders’] / data[‘clicks’]*

В таком случае вновь применяется метод .apply. Однако, теперь для него нужно указать дополнительный параметр **axis**, который указывает направление анализа. По умолчанию axis = 0, что означает, что анализируется каждый столбец. Применяя к единственному столбцу, мы ограничиваем применение функции. Параметр axis = 1 означает, что функция будет применяться построчно. В таком случае, нужно применить метод ко всему датафрейму. При необходимости можно полноценно преобразовывать таблицу.

**Пример:**

*df['conversion'] = df.apply(conversion, axis=1)*

*df[['user\_id', 'clicks', 'orders', 'conversion']]*

*user\_id clicks orders conversion*

*0 1 163 2 0.012270*

*1 2 130 4 0.030769*

*2 3 97 0 0.000000*

Часто приходится получать данные из ссылок:

*https://site.ru/utm\_source=yandex&utm\_medium=cpc&...*

**Пример:**

*stats = pd.read\_excel('ad\_campaigns.xlsx')*

*stats.columns = ['group', 'phrase', 'effect', 'ad\_id', 'link']*

*stats.head()*

*group phrase effect ad\_id link*

*0 мрт "!мрт !менделеевская" 4.5 https://site.ru/?utm...*

*1 мрт цао "мрт менделеевская" 4.5 https://site.ru/?utm...*

*2 мрт цао мрт менделеевская -шея 7.4 https://site.ru/?utm...*

*3 мрт цао "!мрт !цао" 5.0 https://site.ru/?utm...*

*4 мрт сао "мрт цао" 5.0 https://site.ru/?utm...*

*from urllib import parse*

*def campaign\_name(row):*

*# Метод .urlsplit возвращает объект, содержащий url, разбитый*

*# на именованные фрагменты.*

*parsed = parse.urlsplit(row['link'])*

*#* *UTM-метки содержатся в фрагменте “parsed.query”. Метод .parse\_qs*

*# преобразовывает строку в список: {'utm\_source': ['yandex'],'utm\_medium':*

*# ['cpc'],… }. Это удобно в том случае, если есть параметры с одним и*

*# тем же именем (…&a=1&a=2&a=3…)*

*params\_dict = parse.parse\_qs(parsed.query)*

*return params\_dict['utm\_campaign'][0]*

*stats['campaign'] = stats.apply(campaign\_name, axis=1)*

*stats.head()*

*group phrase effect ad\_id link campaign*

*0 мрт "!мрт !менделеевская" 4.5 https://site.ru/?utm... a825749b87*

*1 мрт цао "мрт менделеевская" 4.5 https://site.ru/?utm... a825749b87*

*2 мрт цао мрт менделеевская -шея 7.4 https://site.ru/?utm... a825749b87*

*3 мрт цао "!мрт !цао" 5.0 https://site.ru/?utm... 9cc0c5f6ca*

*4 мрт сао "мрт цао" 5.0 https://site.ru/?utm... 9cc0c5f6ca*

ГРУППИРОВКИ В PANDAS

Группировка является одной из самых важных операций над датафреймами. Алгоритм её работы следующий:

* В указанном столбце ищутся уникальные значения
* Для каждого уникального значения создаётся свой датафрейм
* Все последующие операции применяются к каждому датафрейму отдельно. В частности, метод .apply
* Датафреймы склеиваются

**Пример:**

*stats.groupby('campaign').count()[['group', 'effect']].head()*

*group effect*

*campaign*

*034ada41b6 3 3*

*03fcaecd1d 2 2*

*0697a81555 2 2*

*08cdcb57a3 3 3*

*0f0ba311fb 2 2*

Для того, чтобы столбец “campaign” перестал быть индексом, следует применить метод **.reset\_index()**.

**Пример:**

*stats.groupby('campaign').count()[['group', 'effect']].reset\_index().head()*

*campaign group effect*

*0 034ada41b6 3 3*

*1 03fcaecd1d 2 2*

*2 0697a81555 2 2*

*3 08cdcb57a3 3 3*

*4 0f0ba311fb 2 2*

Различные операции можно применять к группировке отдельно. Применить несколько функций к одним и тем же данным можно при помощи метода **.agg**.

**Пример:**

*stats['power\_up'] = stats['effect'].apply(lambda x: x\*\*2)*

*results = stats.groupby('campaign').agg({'effect': ['min', 'max'], 'power\_up': 'mean'})*

*results.head()*

*effect power\_up*

*min max mean*

*campaign*

*034ada41b6 4.1 9.0 56.686667*

*03fcaecd1d 4.0 6.5 29.125000*

*0697a81555 4.0 5.5 23.125000*

*08cdcb57a3 5.5 7.4 38.420000*

*0f0ba311fb 4.4 6.4 30.160000*

К вложенным столбцам можно обратиться через квадратные скобки.

**Пример:**

*results['effect']['max'].head()*

*campaign*

*034ada41b6 9.0*

*03fcaecd1d 6.5*

*0697a81555 5.5*

*08cdcb57a3 7.4*

*0f0ba311fb 6.4*

*Name: max, dtype: float64*

Групировать можно по нескольким столбцам сразу. Для этого их нужно указать как список в методе .groupby.

**Пример:**

*stats.groupby(['group', 'campaign']).count().head()*

*phrase effect ad\_id link power\_up*

*group campaign*

*1 5 москва мрт тесла 777160ba51 1 1 1 1 1*

*8e77a0e565 1 1 1 1 1*

*f0079349af 1 1 1 1 1*

*1 градская мрт 7f35591a28 3 3 3 3 3*

*1 мрт нагатинский 8e77a0e565 1 1 1 1 1*