Конспект по теме «Работа с несколькими источниками данных»

Срез по данным из внешнего словаря

При работе со срезами, можно использовать внешние переменные не только числового или строкового типа, но и листы, словари, серии и даже датафреймы. Обращаться к ним можно, как к обычным внешним переменным. Если для среза используется лист, то проводится проверка, что значение в определённой колонке входит или не входит в лист:

```
our_list = [1, 2, 3]
print(data.query('column in @our_list'))
```

Аналогичная проверка осуществляется с помощью словаря, но проверяется, что значение в определённой колонке входит или не входит в ключи словаря:

```
our_dict = {0: 10, 1: 11, 2: 12}
print(data.query('column in @our_dict'))
```

Когда в переменной сохранён объект *Series,* конструкция `column in @our_series` проверит вхождение в список *значений*, а не *индексов:*

```
our_series = pd.Series([10,11,12])
print(data.query('column in @our_series'))
```

Если нужно проверить вхождение в индекс, это указывают явно, дописывая *index* через точку: `column in @our_series.index`:

```
our_series = pd.Series([10,11,12])
print(data.query('column in @our_series.index'))
```

Когда имеют дело с объектом *DataFrame*, вхождение в индекс проверяют так же, как в *Series* — приписав *index* через точку к имени датафрейма:

```
our_dataframe = pd.DataFrame ({
'column1': [2, 4, 6],
'column2': [3, 2, 2],
'column3': ['A', 'B', 'C'],
```

```
})
print(data.query('column in @our_dataframe.index'))
```

Для проверки вхождения в какой-либо столбец, передают его имя:

```
our_dataframe = pd.DataFrame ({
'column1': [2, 4, 6],
'column2': [3, 2, 2],
'column3': ['A', 'B', 'C'],
})
print(data.query('column in @our_dataframe.column2'))
```

Добавляем столбец

Несколько гистограмм можно отобразить на одном графике. Для этого можно применять следующую конструкцию:

```
ax = data1.plot(kind='hist', y='column1', histtype='step', range=(0, 500), bins=25,
                              linewidth=5, alpha=0.7, label='raw')
data2.plot(kind='hist', y='column1', histtype='step', range=(0, 500), bins=25,
                       linewidth=5, alpha=0.7, label='filtered', ax=ax, grid=True, legend=True)
```

Обратите внимание, что мы вызвали не метод `hist()`, а `plot()` с параметром **kind**, которому установили значение *kind='hist'*. Это та же гистограмма, только поддерживающая параметры, которые нельзя задействовать методом `hist()`:

- **histtype -** тип гистограммы, по умолчанию — это столбчатая (закрашенная). Значение *'step'* чертит только линию.
- **linewidth -** толщина линии графика в пикселях.
- **alpha -** густота закраски линии. 1 — это 100% закраска; 0 — прозрачная линия.
- **label -** Название линии.
- **ax**. Метод `plot()` возвращает оси, на которых был построен график. Чтобы обе гистограммы расположились на одном графике, сохраним оси первого графика в переменной `ax`, а затем передадим её значение параметру `ax` второго `plot()`. Так, сохранив оси одной гистограммы и построив вторую на осях первой, мы объединили два графика.
- **legend:** выводит легенду — список условных обозначений на графике.

При добавлении столбцов в датафрейм, нужно учесть несколько нюансов. Пусть у нас есть два датафрейма: `data1` и `data2`. Добавим в `data1` новую колонку, значения которой будут совпадать со значениями столбца `data2['column']`:

```
data1['new_column'] = df2['column']
```

Если бы столбец `new` уже был в `data1`, то все его элементы были бы удалены, а вместо них записаны новые.

Кажется просто: *Pandas* копирует столбец из `data2` и вставляет его в `data1`, однако всё сложнее. Для каждой строки первого датафрейма *Pandas* ищет «пару» — строку с таким же индексом во втором датафрейме. Находит и берёт значение из этой строки. В нашем случае индексы в `data1` и `data2` совпадали, и всё казалось простым копированием строк по порядку. Если же индексы не будут совпадать, например, в `data2` не будет некоторых значений индекса `data1` , то при копировании столбца на их месте будет значение *NaN*.

Число строк в `data2` не обязательно должно совпадать с числом строк `data1`. Если в `data2` не хватит значений, то будет *None*. А будут лишние — просто не попадут в обновлённый датафрейм. А вот повторяющиеся значения в индексе `data2` приведут к ошибке. *Pandas* не поймёт, какое из значений следует подставить в `data1`.

Отдельный столбец можно создать и без датафрейма, в *Series* — это будет набор значений с индексами. При попытке присвоить объект с индексами, *Pandas* подберёт соответствующие индексам строки. Если передавать столбцу список значений без индекса, такой как *list*, присвоение будет идти по порядку строк.

Объединяем данные из двух таблиц

При работе над задачей, нужно грамотно выбирать метод усреднения, так как он может повлиять на выводы. Где-то среднее арифметическое более точно опишет данные, а где-то может дать некорректный результат - тогда нужно вычислять медиану.

Метод `pivot_table` группирует данные, а что с ними делать, указывает значение параметра *aggfunc*. Среди таких значений есть *'first'* — первое значение из группы и *'last'* — последнее значение из группы.

В одном вызове `pivot_table` можно передать параметру *aggfunc* списком несколько функций — в результирующей таблице они будут в соседних столбцах.

Переименование столбцов

Когда в индексе не одно значение, а целый список, то получаются двухэтажные названия столбцов - **мультииндекс**. Вообще индексы можно представить как ещё один столбец в датафрейме. Выходит, мультииндекс — несколько столбцов. Это справедливо и для названий столбцов: они как дополнительные строки в датафрейме. В таком случае мультииндекс — это две и более строк с названиями.

Объединение столбцов методами merge() и join()

Когда нужно к существующему датафрейму добавить несколько новых столбцов, существует более лаконичный способ сделать это, по сравнению добавлением каждого столбца по очереди. Такая процедура называется объединение (join) или слияние (merge).

Для слияния используем метод `merge()`, параметр `how` - метод объединения:

```
data1.merge(data2, on='column', how='inner')
```

Если некоторые значения в колонке слияния совпадают, то остаются только одни, такой тип объединения называется `inner`, т.е. внутренняя общая область где есть и данные `data1` и `data2`. Существует тип слияния `outer`, т.е. внешняя общая область — область, где есть данные хотя бы в одном из df1 или df2; режим объединения `left`,когда в результате слияния обязательно будут все строки из левого DataFrame (в нашем случае `data1`); и есть полностью аналогичный режим `right`, когда сохраняются все строки из `data2`.

Если совпадут имена столбцов в `data1` и `data2`, то pandas переименует их, чтобы мы могли разобраться, где какое значение. К названию столбца из `data1` в таком случае припишется `_x`, а к столбцу из `data1` - `_y`. Вы можете самостоятельно задать, что приписать к названию столбцов, используя параметр `suffixes=('_x', '_y')` (заменив `'_x', '_y'` на нужные вам окончания имён столбцов).

Отметим так же, что если столбец индекса именованный, то можно передать его имя в параметр `on`. Объединять можно не только по одному столбцу, но и по совпадению значений в нескольких столбцах — достаточно только передать список в `on`.

Так же в pandas существует аналогичный метод `join()`, который ищет совпадения по индексам в `data1` и `data2`, если не указать параметр `on`. А в случае указания параметра `on` — он будет искать соответствующий столбец в `data1`, и сравнивать его с индексом `data2`. Кроме того, если в `merge()` по умолчанию `how='inner'`, то `join()` использует по умолчанию `how='left'`. Отличается и название параметра `suffixes` — они разделены на 2 независимых `lsuffix` и `rsuffix`. Ещё `join` позволяет объединить сразу более двух DataFrame — их набор можно передать списком вместо одного `data2`.