3.2 Конспект третьего урока

Шаг 1

Оглавление урока

Ссылки на шаги с кусочками конспекта из этого урока:

Шаг 2

Уникальные значения

`unique` – метод, возвращающий уникальные значения в колонке.

```
data.fam_sp.unique()
```

```
array(['PASTA ALIMENTICIA SE'], dtype=object)
```

Уникальные значения возвращаются в форме array – о них будет сказано позже. Для прост

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.unique.html

Шаг 3

Число уникальных значений

`nunique` – метод, который считает число уникальных значений в колонке.

```
data.fam_sp.nunique()
```

```
1
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 4

Медиана

Чтобы посчитать медиану колонки, используйте метод `median`

```
users_data.orders.median()
```

```
4.0
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Среднее

Для расчёта среднего значения используйте метод `mean`

```
users_data.orders.mean()
```

```
5.487290227048371
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 5

split

`split` – метод, который разбивает строку на куски и помещает фрагменты в список. По ум символам (пробел, табы, перенос строки).

```
brand_info = 'MARAVILLA 500 G Store_Brand'
brand_info.split()
```

```
['MARAVILLA', '500', 'G', 'Store_Brand']
```

Больше информации (https://www.w3schools.com/python/ref_string_split.asp)

Шаг 6

Анонимные функции

Обычно используются, когда нужно куда-то быстро поместить нечасто используемый функц
использовать анонимную функцию больше одного раза, лучше напишите обычную функцию

```
lambda x: do something
```

- `labmda` – ключевое слово, задающее анонимную функцию (не имеющую имени)
- `x` – то, как мы назвали аргумент, принимаемый функцией
- `:` – разделяет заголовок и тело безымянной функции
- `do something` – тело функции, должно помещаться в одну строчку, будет автоматиче

```
# Take 1 argument and add 3 to it
lambda x: x + 3
```

Один из примеров использования лямбда-функции – переименование колонок в датафрэйм
заглавными и заменяем дефисы на нижние подчёркивания.

```
# df is a dataframe as usual
df = df.rename(columns=lambda c: c.upper().replace('-', '_'))
```

Больше информации (https://realpython.com/python-lambda/)

Шаг 7

Серии

`pd.Series` – более примитивный тип данных в `pandas`, соответствует колонке датафрэйм
типа (числа, строки и т.п.). Работая с колонкой, мы работаем именно с серией. Часть методо
датафрэйма совпадают.

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

Шаг 8

Применение функций к датафрэйму

`apply` – применяет переданную в него функцию ко всем колонкам вызванного датафрэйм
одной колонке датафрэйма, можно выбрать её перед применением `apply`, например:

```
data.art_sp.apply(lambda c: c.split()[-1])
```

```
0        Store_Brand
1        Store_Brand
2            Brand_1
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 9

Объединение датафрэймов

Зачастую называется джойном. Очень частая операция, которую можно сделать с помощью
них – `merge`. Обязательным аргументом является другой датафрэйм, с которым планирует
идёт по общей колонке, у которой имеется одинаковый смысл и общие значения в обоих да
различные типы джойнов, они будут рассмотрены в курсе по SQL. Пожалуй, самый частый и

Здесь мы объединяем датафрэйм `users_data` с датафрэймом `users_lovely_brand_data`
джойна:

```
users_data.merge(users_lovely_brand_data, how='inner', on='tc')
```

	tc	orders	unique_brands	lovely_brand	max_orders
0	1031	6	2	Store_Brand	5
1	4241	5	2	Brand_4	3
2	17311	2	1	Brand_4	2
3	17312	2	2	Brand_1	1

В результате получается один датафрэйм, в котором колонки из двух таблиц, относящиеся к
объединяются в строку. Звучит сложно, поэтому для практики стоит попробовать сделать н

Дополнительные аргументы функции merge

- `how` – как объединять датафрэймы, возможные варианты: inner, outer, left, right

- `on` – общая колонка, по которой будет происходить объединение

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 10

Индекс и имена колонок

Индекс – это лэйбл строки в таблице, по умолчанию является её номером. А имена колонок лэйблы, по которым мы можем обращаться к каждому из столбцов.

У датафрэйма есть два атрибута – `index` и `columns`. Они позволяют получить доступ к со виде array (на самом деле не совсем array).

	journey_id	driver_id
easy	135	99
executive	22737	19484
group	239	143

```
df.index
Index(['easy', 'executive', 'group'], dtype='object')
```

```
df.columns
Index(['journey_id', 'driver_id'], dtype='object')
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Index.html)

Шаг 11

Сброс индекса

Иногда вам может потребоваться перевести индекс датафрэйма в колонку. Для этого суще Индексом становится дефолтная последовательность чисел от 0 до N-1, где N – число стро

```
df
```

event	click	view
date_day		
2019-04-01	881	41857
2019-04-02	1612	165174

2019-04-03	1733	224843
2019-04-04	1447	107098
2019-04-05	581790	2050279

```
df.reset_index()
```

	event	date_day	click	view
0	2019-04-01	881	41857
1	2019-04-02	1612	165174
2	2019-04-03	1733	224843
3	2019-04-04	1447	107098
4	2019-04-05	581790	2050279

Удаление индекса

Аргумент `drop` отвечает за то, будет ли индекс переведён в колонку или же убран из табли

```
df.reset_index(drop=True)
```

	event	click	view
0	881	41857
1	1612	165174
2	1733	224843
3	1447	107098
4	581790	2050279

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 12

Поиск пустых значений

`isna` – это чудо-метод, с помощью которого можно быстро найти пропущенные значения в

```
df.head()
```

	Id	sum
0	1	150.0
1	2	230.0
2	3	NaN
3	4	143.0

4	5	NaN

Применив его, на выходе мы получаем датафрэйм той же размерности, где в каждой ячейке зависимости от того, было ли значение пропущено.

```
df.isna()
```

	id	sum
0	False	False
1	False	False
2	False	True
3	False	False
4	False	True

В связке с ним можно использовать, например, метод `sum`, чтобы посмотреть на число `NA`

```
df.isna().sum()
```

```
id     0
sum    4
dtype: int64
```

Документация (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.

Шаг 13

Графики

Графики – важная часть анализа данных, так как они наглядно представляют данные (если позволяют быстро разобраться в их сути.

Чтобы графики отображались в юпитер ноутбуке, необходимо выполнить следующую строч

```
%matplotlib inline
```

В Python существуют разные способы создания графиков. Популярные библиотеки для визу

- `pandas`
- `seaborn`
- `matplotlib`

Давайте начнём постепенно в них разбираться.

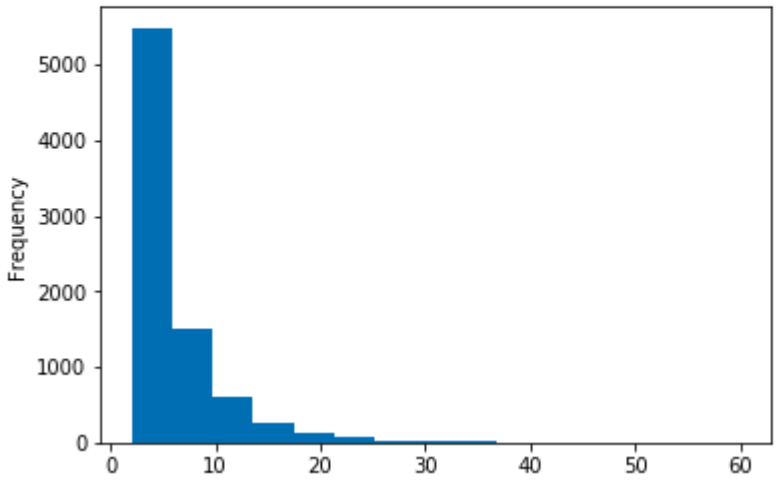
Шаг 14

pandas

Самый простой способ визуализировать данные – вызвать метод `plot` у датафрэйма (или гистограмма значений в колонке `orders` :

```
users_data.orders.plot('hist', bins=15)
```

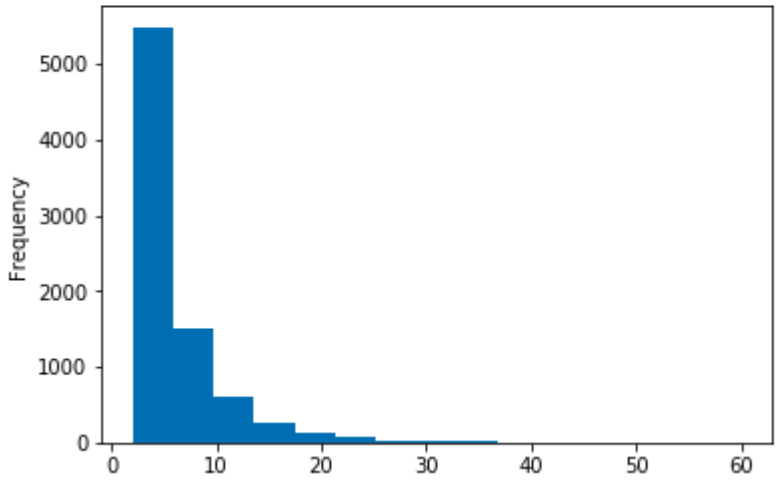
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d35869e8>
```



Другой вариант записи:

```
users_data.orders.plot.hist(bins=15)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0f0dd1ef0>
```



Функции рисования имеют весьма большое количество параметров, используйте их при не число диапазонов (корзин или бакетов), на которые мы разделяем значения.

Документация (https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)

Шаг 15

seaborn

Продвинутая библиотека, позволяющая делать очень красивые графики. Согласно конвенц образом:

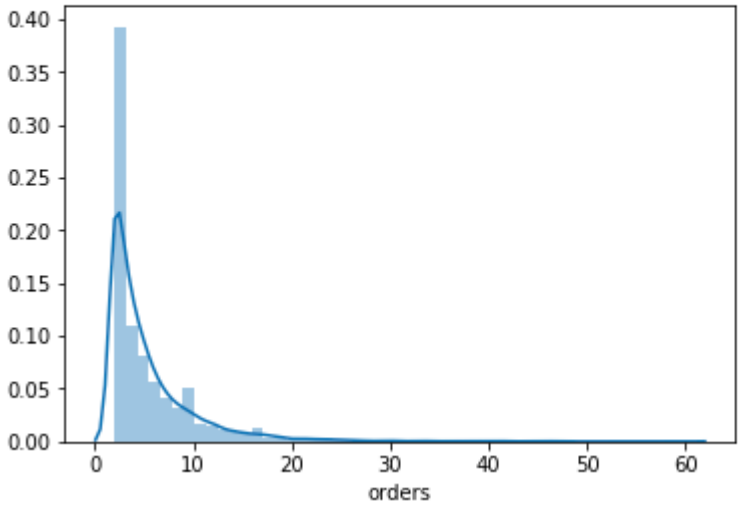
```
import seaborn as sns
```

Ниже представлены примеры создания графиков с её помощью.

Гистограмма

```
sns.distplot(users_data.orders)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d9315d68>
```

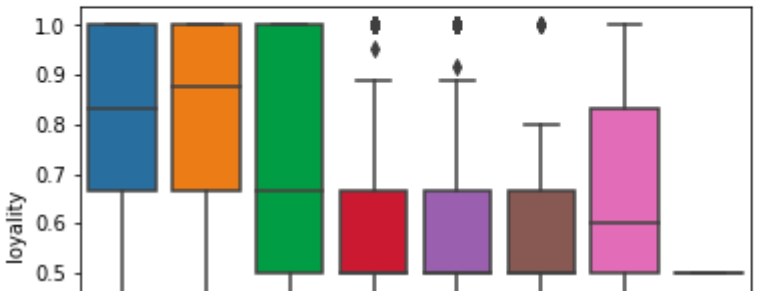


Документация (https://seaborn.pydata.org/generated/seaborn.distplot.html)

Боксплот

```
sns.boxplot(data=users_data, x='lovely_brand', y='loyality')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d8ead048>
```

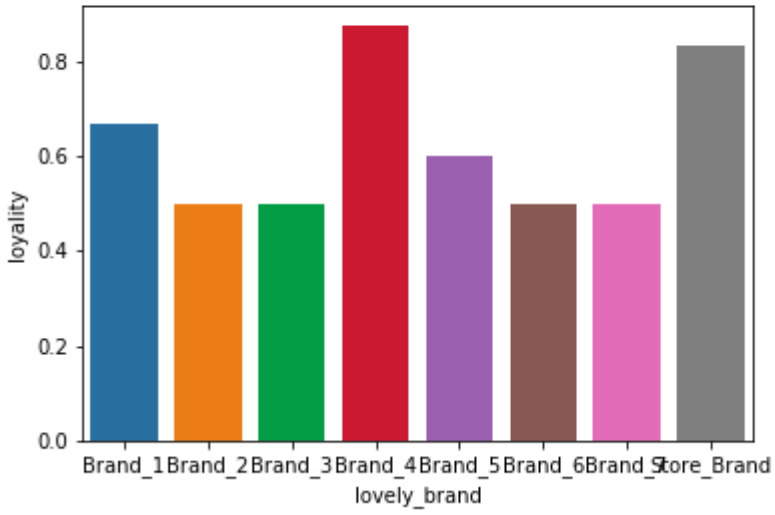


Документация (https://seaborn.pydata.org/generated/seaborn.boxplot.html)

Барплот

```
sns.barplot(x='lovely_brand', y='loyality', data=loyality_stat)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d8a28f28>
```



На графиках перекрываются подписи. Скоро мы разберёмся, как это можно исправить.

Документация (https://seaborn.pydata.org/generated/seaborn.barplot.html)

Шаг 16

matplotlib

Базовая библиотека для рисования графиков в Python. На ней построены более продвинуть библиотеки типа `seaborn`. Через `matplotlib` можно нарисовать что угодно, но часто на эт кода, и её в основном используют для тонкой настройки графиков и их сохранения.

Традиционно `matplotlib` импортируется следующим образом:

```
import matplotlib.pyplot as plt
```

Настройка графиков

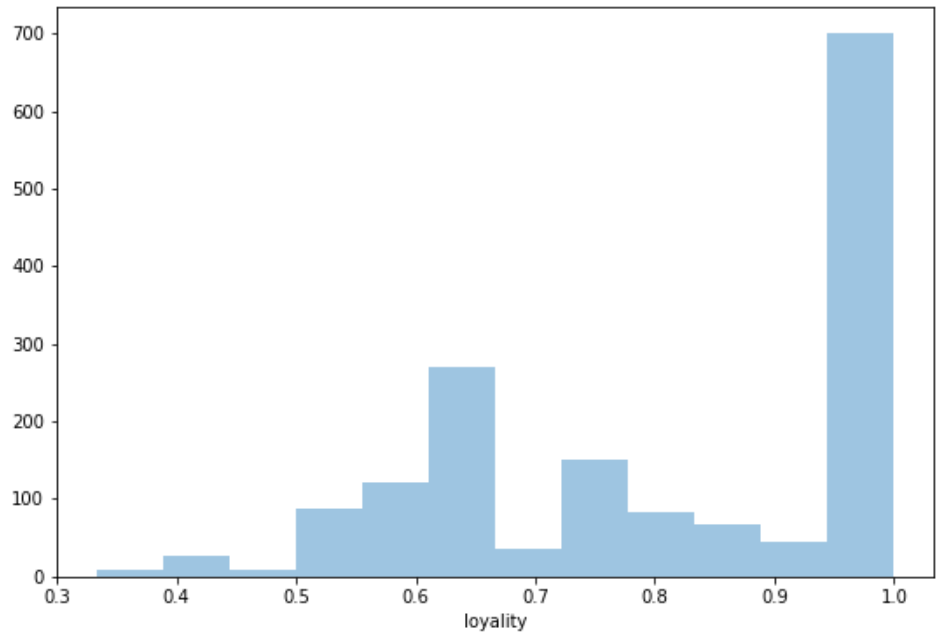
Важный момент – большинство настроек должны быть написаны к каждому графику отдел
написанные в ячейке с одним графиком, не будут применены к другому.

Изменить размер

В `figure` в `figsize` подаётся кортеж (как список, только в круглых скобках) с масштабом
`высота)`

```
plt.figure(figsize=(9, 6))
sns.distplot(users_data.query('lovely_brand == "Store_Brand"').loyality, kde=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe0d3514240>
```



Больше информации (https://stackoverflow.com/questions/332289/how-do-you-change-the-size-

Сохранение картинки

Сохранить график можно с помощью `savefig` , где аргумент – это путь к сохраняемой карт
формат):

```
sns.distplot(users_data.query('lovely_brand == "Store_Brand"').loyality, kde=False)
plt.savefig('1.jpg')
```

Документация (https://matplotlib.org/api/_as_gen/matplotlib.pyplot.savefig.html)