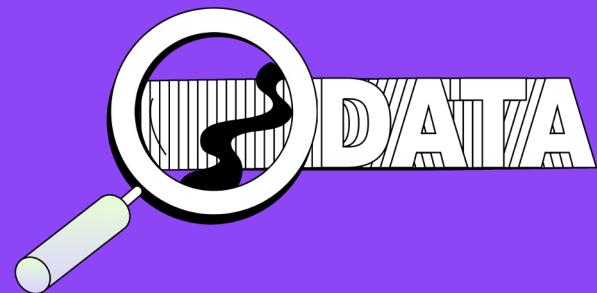


Анализ данных с библиотекой Pandas

Урок 2





Что будет на уроке сегодня

- 📌 Анализ табличные данные
- 📌 Статистики таблиц
- 📌 Фильтрация данных
- 📌 Сортировка данных





Библиотека pandas



Зачем pandas?

	Google Sheets	Microsoft Excel	Pandas
Ограничения по данным	4 989 894 ячеек	1 048 576 строк и 16 384 столбцов	Нет ограничений
Стоимость	Бесплатно	8.25\$ в месяц	Бесплатно



Начало работы с Pandas

Чтобы начать работу с функциями и классами из библиотеки:

- 📌 В командной строке нужно написать pip install pandas (если работаете не через анаконду)
- 📌 В командной строке нужно написать conda install pandas (если работаете через анаконду)
- 📌 Импортировать себе в ноутбук библиотеку через import pandas as pd

```
import pandas as pd
```



Объект Series



Синтаксис создания pd.Series

Объект – это набор данных (переменных) и методов (функций), которые с этими данными взаимодействуют.

Pandas Series представляет из себя объект, похожий на одномерный массив, но отличительной чертой является наличие индексов. Индекс находится слева, а сам элемент справа.

one	4
two	7
three	6
four	3
five	9
dtype: int64	



Синтаксис создания pd.Series

```
pandas.Series(input_data, index, data_type)
```

- 📌 input_data: ввод в виде списка, константы, массива NumPy, Dict и т. д.
- 📌 index: значения индексов.
- 📌 data_type (опционально): тип данных.



Пример создания pd.Series

```
a = pd.Series([4, 7, 6, 3, 9],  
             index=['one', 'two', 'three', 'four', 'five'])  
a
```

```
one      4  
two      7  
three     6  
four      3  
five      9  
dtype: int64
```



Объект DataFrame



Синтаксис создания pd.Series

Объект **DataFrame** является табличной структурой данных. В любой таблице всегда присутствуют строки и столбцы. При этом в столбцах можно хранить данные разных типов данных.

Столбцами в объекте DataFrame выступают объекты Series, строки которых являются их элементами.

	Age	Country	Gender
0	46	Spain	Female
1	37	Spain	Female
2	44	Germany	Male
3	42	Germany	Male
4	42	France	Male



Синтаксис создания pd.DataFrame

```
pandas.DataFrame(input_data, index)
```

- 📌 input_data: ввод в виде Dict, 2D массива NumPy, Series и т. д.
- 📌 index: значения индексов.



Пример создания pd.DataFrame

```
df = pd.DataFrame({  
    'Age': [46, 37, 44, 42, 42],  
    'Country': ['Spain', 'Spain', 'Germany', 'Germany', 'France'],  
    'Gender': ['Female', 'Female', 'Male', 'Male', 'Male']  
})  
df
```

	Age	Country	Gender
0	46	Spain	Female
1	37	Spain	Female
2	44	Germany	Male
3	42	Germany	Male
4	42	France	Male



Считывание данных



Считывание данных

В целом, pandas поддерживает все самые популярные форматы хранения данных: csv, excel, sql, html и многое другое, но чаще всего приходится работать именно с csv файлами (comma separated values)

```
df = pd.read_csv('./Churn_Modelling.csv')
df
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	938
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	790



Считывание данных

Аргумент `sep` указывает разделитесь столбцов, поставим, к примеру `sep=';'`, в этом случае pandas будет искать символ `;`, чтобы разбить столбцы, но ничего не найдет, поэтому все значения сольются воедино.

```
pd.read_csv('./Churn_Modelling.csv', sep=';')
```

	RowNumber,CustomerId,Surname,CreditScore,Geography,Gender,Age,Tenure,Balance,NumOfProducts,HasCrCard,IsActiveMember,EstimatedSalary,Exited
0	1,15634602,Hargrave,619,France,Female,42,2,0,1...
1	2,15647311,Hill,608,Spain,Female,41,1,83807.86...
2	3,15619304,Onio,502,France,Female,42,8,159660....
3	4,15701354,Boni,699,France,Female,39,1,0,2,0,0...
4	5,15737888,Mitchell,850,Spain,Female,43,2,1255...



Первичный анализ данных



Первичный анализ данных

Метод info() - сводная таблица по типам данных, по количеству непропущенных объектов и по потреблению памяти в таблице

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
RowNumber      10000 non-null int64
CustomerId     10000 non-null int64
Surname        10000 non-null object
CreditScore    10000 non-null int64
Geography      10000 non-null object
Gender          10000 non-null object
Age             10000 non-null int64
Tenure          10000 non-null int64
Balance         10000 non-null float64
NumOfProducts   10000 non-null int64
HasCrCard       10000 non-null int64
IsActiveMember  10000 non-null int64
EstimatedSalary 10000 non-null float64
Exited          10000 non-null int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



Первичный анализ данных

Основные статистики можно получить через метод `describe()`

```
df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000



Первичный анализ данных

`describe()` по умолчанию считает статистики только для вещественных признаков, а строковые игнорирует, чтобы `describe` посчитал на них статистики, нужно явно это указать

```
df.describe(include=['object'])
```

	Surname	Geography	Gender
count	10000	10000	10000
unique	2932	3	2
top	Smith	France	Male
freq	32	5014	5457



Фильтрация



Фильтрация

Фильтрация в pandas основывается на булевых масках.

Булевая маска – бинарные данные, которые используются для выбора определенных объектов из структуры данных.

```
df['Gender'] == 'Male'  
0      False  
1      False  
2      False  
3      False  
4      False  
...  
9995     True  
9996     True  
9997    False  
9998     True  
9999    False  
Name: Gender, Length: 10000, dtype: bool
```



Фильтрация

Булевую маску можно передать в датафрейм и на выходе мы получим только те объекты, которые следуют данному условию

```
male = df[df['Gender'] == 'Male']  
male
```

RowIndex	CustomerId	Surname	CreditScore	Geography	Gender	
5	6	15574012	Chu	645	Spain	Male
6	7	15592531	Bartlett	822	France	Male
8	9	15792365	He	501	France	Male
9	10	15592389	H?	684	France	Male
10	11	15767821	Bearce	528	France	Male



Логическое И

При операторе & нужно, чтобы выполнялось два условия одновременно, чтобы у объекта был женский пол (df['Gender'] == 'Female') и чтобы количество продуктов, которыми пользуется клиент было больше или равно 3 (df['NumOfProducts'] >= 3)

```
df[(df['Gender'] == 'Female') & (df['NumOfProducts'] >= 3)]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts
2	3	15619304	Onio	502	France	Female	42	8	159660.80
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74
30	31	15589475	Azikiwe	591	Spain	Female	39	3	0.00
88	89	15622897	Sharpe	646	France	Female	46	4	0.00
90	91	15757535	Heap	647	Spain	Female	44	5	0.00



Логическое ИЛИ

При операторе | нужно, чтобы выполнялось хотя бы одно условие. Либо у человека есть карта банка (df['HasCrCard']), либо количество продуктов, которыми пользуется клиент было больше или равно 3 (df['NumOfProducts'] >= 3)

```
df[(df['HasCrCard']) | (df['NumOfProducts'] >= 3)]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	True
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	True
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	True
5	6	15574012	Chu	645	Spain	Male	44	8	113755.78	2	True
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	True



Логическое НЕ

При операторе ~ булевая маска обращается: True меняется на False и наоборот. Возвращаем всех клиентов, которые проживают не в Испании (~(df['Geography'] == 'Spain'))

```
df[~(df['Geography'] == 'Spain')]
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	True
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	True
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	False
6	7	15592531	Bartlett	822	France	Male	50	7	0.00	2	True
7	8	15656148	Obinna	376	Germany	Female	29	4	115046.74	4	True



Индексация



.loc

Данный метод позволяет взять объект по конкретному ключу строки или столбца

Возвращаем в примере 1, 4, 5 строку по ключу и столбцы с названиями 'Gender' и 'Age'

```
df_small.head()
```

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44
11	Spain	Male	24
14	Spain	Female	35

```
df_small.loc[[1, 4, 5], ['Gender', 'Age']]
```

	Gender	Age
1	Female	41
4	Female	43
5	Male	44



.iloc

Данный метод позволяет взять объект по порядковому ключу строки или столбца.

В нашем примере есть ключи по индексам 1, 4, 5, но через iloc можем их достать по порядку: 0,1,2

```
df_small.head()
```

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44
11	Spain	Male	24
14	Spain	Female	35

```
df_small.iloc[[0, 1, 2]]
```

	Geography	Gender	Age
1	Spain	Female	41
4	Spain	Female	43
5	Spain	Male	44



Сортировка



Сортировка

Метод `sort_values()` сортирует таблицу по признаку. Сортировка будет выполнена от меньшего к большему. В данном примере узнаем самого младшего клиента

```
df.sort_values('Age')
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
3512	3513	15657779	Boylan	806	Spain	Male 18
1678	1679	15569178	Kharlamov	570	France	Female 18
3517	3518	15757821	Burgess	771	Spain	Male 18
9520	9521	15673180	Onyekaozulu	727	Germany	Female 18
2021	2022	15795519	Vasiliev	716	Germany	Female 18
...
3387	3388	15798024	Lori	537	Germany	Male 84
3033	3034	15578006	Yao	787	France	Female 85
2458	2459	15813303	Rearick	513	Spain	Male 88
6759	6760	15660878	T'ien	705	France	Male 92
6443	6444	15764927	Rogova	753	France	Male 92



Сортировка

ascending=False - применить сортировку наоборот от большего к меньшему

В данном примере узнаем самого младшего старшего клиента

```
df.sort_values('Age', ascending=False)
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age
6443	6444	15764927	Rogova	753	France	Male	92
6759	6760	15660878	Tien	705	France	Male	92
2458	2459	15813303	Rearick	513	Spain	Male	88
3033	3034	15578006	Yao	787	France	Female	85
3387	3388	15798024	Lori	537	Germany	Male	84
...
9782	9783	15728829	Weigel	509	France	Male	18
2141	2142	15758372	Wallace	674	France	Male	18
9501	9502	15634146	Hou	835	Germany	Male	18
9520	9521	15673180	Onyekaozulu	727	Germany	Female	18
1619	1620	15770309	McDonald	656	France	Male	18



Сортировка

Можно сортироваться по двум и более признакам. Сначала сортировка пройдет в первом признаком, а если встречаются одинаковые значения, то сортировка коснется и второго признака

Узнаем самых молодых клиентов с самыми низкими кредитными оценками

```
df.sort_values(['Age', 'CreditScore'])
```

RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	
9782	9783	15728829	Weigel	509	France	Male	18
1678	1679	15569178	Kharlamov	570	France	Female	18
9029	9030	15722701	Bruno	594	Germany	Male	18
7334	7335	15759133	Vaguine	616	France	Male	18
9526	9527	15665521	Chiazagomekpele	642	Germany	Male	18
...	
3387	3388	15798024	Lori	537	Germany	Male	84
3033	3034	15578006	Yao	787	France	Female	85
2458	2459	15813303	Rearick	513	Spain	Male	88
6759	6760	15660878	Tien	705	France	Male	92
6443	6444	15764927	Rogova	753	France	Male	92



Итоги урока

- 📌 Узнали, как анализировать табличные данные
- 📌 Научились считать статистики датафрейма
- 📌 Более детально изучили фильтрацию данных
- 📌 Разобрались с сортировками





Что будет на следующем уроке

- 📌 Создание, изменение и удаление признаков
- 📌 Группировки данных
- 📌 Объединение таблиц
- 📌 Встроенные визуализации





Полезные ссылки

Типы данных <https://youtu.be/c4Cg3TUIH0E>

Введение в pandas <https://youtu.be/gJKN8zyG5c0>

Документация pandas <https://pandas.pydata.org/docs/>

Аналитикам: большая шпаргалка по Pandas
<https://smysl.io/blog/pandas/>





Спасибо за урок!

