[Первичный анализ данных с Pandas](https://habrahabr.ru/company/ods/blog/322626/)

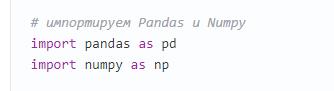
Pandas — это библиотека Python, предоставляющая широкие возможности для анализа данных. Данные, с которыми работают датасаентисты, часто хранятся в форме табличек — например, в форматах .csv, .tsv или .xlsx. С помощью библиотеки Pandas такие табличные данные очень удобно загружать, обрабатывать и анализировать с помощью SQL-подобных запросов. А в связке с библиотеками Matplotlib и Seaborn Pandas предоставляет широкие возможности визуального анализа табличных данных.

Основными структурами данных в Pandas являются классы Series и DataFrame. Первый из них представляет собой одномерный индексированный массив данных некоторого фиксированного типа. Второй – это двухмерная структура данных, представляющая собой таблицу, каждый столбец которой содержит данные одного типа. Можно представлять её как словарь объектов типа Series. Структура DataFrame отлично подходит для представления реальных данных: строки соответствуют признаковым описаниям отдельных объектов, а столбцы соответствуют признакам.

Библиотека pandas активно используется в современном data science для работы с данными, которые могут быть представлены в виде таблиц (а это очень, очень большая часть данных)

pandas есть в пакете Anaconda, но если вдруг у Вас её по каким-то причинам нет, то можно установить, раскомментировав следующую команду







Будем показывать основные методы в деле, анализируя [набор данных](https://bigml.com/user/francisco/gallery/dataset/5163ad540c0b5e5b22000383) по оттоку клиентов телеком-оператора. Прочитаем данные (метод read\_csv) и посмотрим на первые 5 строк с помощью метода head:



| **Название** | **Описание** | **Тип** |
| --- | --- | --- |
| **State** | Буквенный код штата | номинальный |
| **Account length** | Как долго клиент обслуживается компанией | количественный |
| **Area code** | Префикс номера телефона | количественный |
| **International plan** | Международный роуминг (подключен/не подключен) | бинарный |
| **Voice mail plan** | Голосовая почта (подключена/не подключена) | бинарный |
| **Number vmail messages** | Количество голосовых сообщений | количественный |
| **Total day minutes** | Общая длительность разговоров днем | количественный |
| **Total day calls** | Общее количество звонков днем | количественный |
| **Total day charge** | Общая сумма оплаты за услуги днем | количественный |
| **Total eve minutes** | Общая длительность разговоров вечером | количественный |
| **Total eve calls** | Общее количество звонков вечером | количественный |
| **Total eve charge** | Общая сумма оплаты за услуги вечером | количественный |
| **Total night minutes** | Общая длительность разговоров ночью | количественный |
| **Total night calls** | Общее количество звонков ночью | количественный |
| **Total night charge** | Общая сумма оплаты за услуги ночью | количественный |
| **Total intl minutes** | Общая длительность международных разговоров | количественный |
| **Total intl calls** | Общее количество международных разговоров | количественный |
| **Total intl charge** | Общая сумма оплаты за международные разговоры | количественный |
| **Customer service calls** | Число обращений в сервисный центр | количественный |

Целевая переменная: **Churn** – Признак оттока, бинарный признак (1 – потеря клиента, то есть отток). Потом мы будем строить модели, прогнозирующие этот признак по остальным, поэтому мы и назвали его целевым.

Посмотрим на размер данных, названия признаков и их типы.



Выведем названия столбцов:



Чтобы посмотреть общую информацию по датафрейму и всем признакам, воспользуемся методом **info**:



**Изменить тип колонки** можно с помощью метода astype. Применим этот метод к признаку Churn и переведём его в int64:



Метод **describe** показывает основные статистические характеристики данных по каждому числовому признаку (типы int64 и float64): число непропущенных значений, среднее, стандартное отклонение, диапазон, медиану, 0.25 и 0.75 квартили.



Чтобы посмотреть статистику по нечисловым признакам, нужно явно указать интересующие нас типы в параметре include.



Для категориальных (тип object) и булевых (тип bool) признаков можно воспользоваться методом **value\_counts**. Посмотрим на распределение данных по нашей целевой переменной — Churn:



Посмотрим на распределение пользователей по переменной Area code. Укажем значение параметра normalize=True, чтобы посмотреть не абсолютные частоты, а относительные.



Для подсчета пропущенных значений в фрейме данных используем метод [df.isnull()](https://www.delftstack.com/ru/howto/python-pandas/how-to-count-the-nan-values-in-a-column-in-pandas-dataframe/#%25D0%25BC%25D0%25B5%25D1%2582%25D0%25BE%25D0%25B4-df.isnull.sum-%25D0%25B4%25D0%25BB%25D1%258F-%25D0%25BF%25D0%25BE%25D0%25B4%25D1%2581%25D1%2587%25D0%25B5%25D1%2582%25D0%25B0-nan-%25D0%25B2%25D1%2585%25D0%25BE%25D0%25B6%25D0%25B4%25D0%25B5%25D0%25BD%25D0%25B8%25D0%25B9), метод isna() для подсчета NaN в одной или нескольких колонках, например:





Сортировка

DataFrame можно отсортировать по значению какого-нибудь из признаков. В нашем случае, например, по Total day charge (ascending=False для сортировки по убыванию):



Сортировать можно и по группе столбцов:



Индексация и извлечение данных

DataFrame можно индексировать по-разному. В связи с этим рассмотрим различные способы индексации и извлечения нужных нам данных из датафрейма на примере простых вопросов.

Для извлечения отдельного столбца можно использовать конструкцию вида DataFrame['Name']. Воспользуемся этим для ответа на вопрос: **какова доля людей нелояльных пользователей в нашем датафрейме?**



14,5% — довольно плохой показатель для компании, с таким процентом оттока можно и разориться.

Очень удобной является **логическая индексация** DataFrame по одному столбцу. Выглядит она следующим образом: df[P(df['Name'])], где P — это некоторое логическое условие, проверяемое для каждого элемента столбца Name. Итогом такой индексации является DataFrame, состоящий только из строк, удовлетворяющих условию P по столбцу Name.

Воспользуемся этим для ответа на вопрос: **каковы средние значения числовых признаков среди нелояльных пользователей?**



Скомбинировав предыдущие два вида индексации, ответим на вопрос: **сколько в среднем в течение дня разговаривают по телефону нелояльные пользователи**?



**Какова максимальная длина международных звонков среди лояльных пользователей (Churn == 0), не пользующихся услугой международного роуминга ('International plan' == 'No')?**



Датафреймы можно индексировать как по названию столбца или строки, так и по порядковому номеру. Для индексации **по названию** используется метод **loc**, **по номеру** — **iloc**.

В первом случае мы говорим *«передай нам значения для id строк от 0 до 5 и для столбцов от State до Area code»*, а во втором — *«передай нам значения первых пяти строк в первых трёх столбцах»*.

*Хозяйке на заметку:* когда мы передаём slice object в **iloc**, датафрейм слайсится как обычно. Однако в случае с **loc** учитываются и начало, и конец слайса





Если нам нужна первая или последняя строчка датафрейма, пользуемся конструкцией df[:1] или df[-1:]:



Применение функций к ячейкам, столбцам и строкам

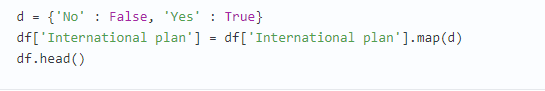
**Применение функции к каждому столбцу: apply**



Метод apply можно использовать и для того, чтобы применить функцию к каждой строке. Для этого нужно указать axis=1.

**Применение функции к каждой ячейке столбца: map**

Например, метод map можно использовать для **замены значений в колонке**, передав ему в качестве аргумента словарь вида {old\_value: new\_value}:



Аналогичную операцию можно провернуть с помощью метода replace:



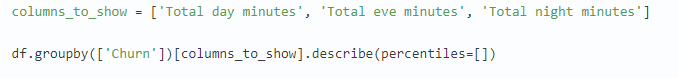
Группировка данных

В общем случае группировка данных в Pandas выглядит следующим образом:



1. К датафрейму применяется метод **groupby**, который разделяет данные по grouping\_columns – признаку или набору признаков.
2. Выбираем нужные нам столбцы (columns\_to\_show).
3. К полученным группам применяется функция или несколько функций.

**Группирование данных в зависимости от значения признака Churn и вывод статистик по трём столбцам в каждой группе.**



Сделаем то же самое, но немного по-другому, передав в agg список функций:

#### 