Краткое описание процесса анализа данных

Языки программирования, используемые в инструкции – **Python 3** и **SQL**. Основная библиотека, которая требуется практически на всех этапах анализа – **pandas**. Основной инструмент – **Jupyter Notebook**.

0. Получение и уточнение задачи

В начале работы важно выяснить:

- кто заказчик задачи (от этого зависит формат представления результатов работы);
- как можно разбить задачу на части (другими словами «декомпозировать»; для любого решения необходим план последовательных действий);
- что нужно посчитать (какие именно показатели/характеристики/метрики нужны на каждом этапе);
- какие данные необходимы для этих расчетов.

Полезные инструменты на этом этапе:

- профессиональный и жизненный опыт;
- анализ существующих решений в компании и за её пределами.

1. Сбор данных

Данные для анализа можно получить несколькими способами:

- Майнинг или парсинг веб ресурсов;

Web mining (web — «сеть»; mining — «добыча, разработка полезных ископаемых») и парсинг (parse — «проводить разбор, анализ»), т.е. извлечение текста, таблиц, изображений, структуры сайтов и данных о пользователях из существующих сайтов. Кроме того, к некоторым сайтам и веб-сервисам можно обращаться через специальный интерфейс передачи данных - API (Application Programming Interface - «программный интерфейс приложения»).

- Извлечение из существующих баз данных (БД).

БД – хранилище структурированной информации, набор таблиц с данными и заданные взаимосвязи между этими таблицами. Для операций с базами используются системы управления базами данных (СУБД). БД могут предоставляться внешними источниками, например, государственные или муниципальные БД адресов/юридических лиц, а могут собираться внутри компании, например, записи об активности пользователей приложения или сайта. Часто работу с БД автоматизируют –однотипную последовательность действий задают с помощью скриптов, так чтобы данные сохранялись и агрегировались по расписанию (каждый день, раз в неделю/месяц/год).

На этом этапе используются следующие инструменты:

- библиотека **requests**, для отправления http-запросов и получения ответов от сервера;
- библиотека **re**, для работы со строками текста, поиска и обработки шаблонов регулярных выражений;
- библиотека **BeautifulSoup**, для работы с html-разметкой страниц;
- модуль **json**, для работы с данными в формате JSON;
- библиотека **sqlalchemy**, для работы с БД с помощью Python;
- библиотека **psycopg2**, для работы с СУБД PostgreSQL.

2. Предварительная обработка данных

Обычно, в собранных данных содержится некоторое количество ошибок, самые распространенные: пропуски, дубликаты, неверные типы и невозможные значения. Независимо от причины возникновения, например, технические ошибки при сборе или пользовательские ошибки при вводе данных, необходимо обработать их перед началом анализа. В некоторых случаях полезно сообщить о проблемах с данными тем специалистам, которые отвечают за их сбор.

- Поиск дубликатов;

Дубликаты могут быть полные (записи полностью идентичны) или скрытые (например, данные в некоторых столбцах отличаются только регистром или количеством пробелов). Иногда встречаются записи, отличающиеся формой слова, но обозначающие одно и то же. Для обработки таких дубликатов нужно использовать стемминг (нахождение основы слова), лемматизацию (приведение слова к словарной форме), регулярные выражения, а в некоторых случая приходится вручную исправлять данные. Кроме того, дублироваться могут не целые строки, а значения в столбце, которые должны быть уникальны. Важно понять природу дубликатов, и если они ошибочны, то их нужно удалить.

- Поиск пропусков;

В зависимости от того, где содержится пропуск, это может привести к удалению всей записи, к замене пропуска на среднее/медианное/крайнее значение признака, или к игнорированию пропуска.

- Приведение к нужным типам;

На этом этапе нужно проверить имеющиеся данные на соответствие их предполагаемым типам: целочисленные значения должны принадлежать одному из целочисленных типов, например, `int` или `bigint`, числа с десятичной частью – `float`, дата и время – `datetime` с нужным форматом.

- Определение невозможных значений.

Иногда данные не пропущены и не дублируют друг друга, но при более внимательном рассмотрении оказывается, что они противоречат здравому смыслу. Например, количество купленных товаров или отправленных сообщений записаны с отрицательным знаком или время окончания интернет-сессии предшествует времени её начала. Это может быть как ошибкой, так и особенностью в сборе данных. Если есть возможность узнать, как именно подготавливались данные, это нужно обязательно сделать.

На этом этапе используются следующие инструменты:

- библиотека **`NLTK`**, для лемматизации;
- библиотека **`pymystem3`**, для стемматизации на русском языке.

3. Исследовательский анализ данных

После нахождения ошибок необходимо изучить имеющиеся данные. Один из способов – посмотреть, какие бывают значения в каждом из столбцов.

- Список уникальных значений;

Для категориальных данных, и для количественных из небольшого диапазона возможных значений, полезно вывести список уникальных значений в столбце с указанием количества записей для каждого. Это можно сделать методом `value_counts()`, примененным к столбцу датафрейма.

- Гистограмма распределения и диаграмма размаха;

В случае количественных переменных, которые распределены в большом диапазоне значений, более наглядным будет построение гистограмм распределения. В этом случае весь диапазон изменения величины делится на определенное число «корзин» - частей этого диапазона, и считается, сколько записей оказалось в той или иной части диапазона. Метод `hist()` примененный к столбцу датафрейма строит гистограмму распределения. Также, для визуализации распределения по одному признаку может использовать метод `boxplot()` для построения диаграммы размаха.

- Диаграмма рассеяния;

Если мы хотим исследовать, как распределены записи не по одному признаку (как было в гистограмме), а по двум, можем построить диаграмму рассеяния. В этом случае каждая запись будет представлена точкой на координатной плоскости, где по оси x будут значения 1-го признака, а по оси y – 2-го. Диаграмма носит название `ScatterPlot`, и может быть построена одноименным методом. Метод `plotting.scatter_matrix(df)` построит диграмму распределения для каждой пары из всех признаков датафрейма `df`.

- Матрица корреляции;

Взаимосвязь двух величин называется корреляцией, а её количественной оценкой может служить коэффициент корреляции Пирсона. Коэффициент принимает значения от -1 до 1, и чем больше значение по модулю, тем сильнее взаимосвязь. Коэффициент может быть посчитан для двух столбцов датафрейма методом `corr()`. Если применить тот же метод к целому датафрейму, получим матрицу коэффициентов корреляции для каждой пары признаков. Матрицу удобно визуализировать т.н. «тепловой картой», для этого нужно использовать метод `heatmap()`. Важно помнить, что корреляция говорит только о взаимосвязи двух величин, ничего не говоря о том, что из двух признаков является причиной, а что следствием, и есть ли вообще причинно-следственная связь между этими двумя признаками.

- Числовые характеристики рассеяния;

Для набора чисел можем определить характерные значения методом `describe()`: количество чисел в наборе, среднее арифметическое, стандартное отклонение, минимальное и максимальное значение, медиана, первый и третий квартили.

- Определение аномальных значений.

Анализируя распределение количественных признаков, часто можно встретить аномальные значения – слишком большие или слишком маленькие значения, не характерные для большинства записей. В зависимости от задачи, такие значения можно оставить или удалить. При удалении обычно ориентируются на значения перцентилей 95, 97, 99: такие значения признака, что соответственно 95, 97, 99 процентов от всех значений меньше этого перцентиля. Для подсчета используют метод `percentile()`. Выбросы и аномалии могут существенно повлиять на итоговые выводы, поэтому важно предварительно их проанализировать.

На этом этапе используются следующие инструменты:

- библиотека **`matplotlib`**, для работы с графиками;
- библиотека **`seaborn`**, для работы с графиками;
- библиотека **`numpy`**, для работы с числами.

4. Анализ ключевых метрик

После подготовки данных и подробного ознакомления с ними, переходят к расчету ключевых показателей. В зависимости от области работы компании и заказчика задачи, эти показатели могут сильно различаться: от ежедневного количества активных пользователей и времени, проведенного на сайте, до среднего чека, суммарной месячной выручки и наиболее популярной категории товаров.

5. Статистический анализ данных

Одним из возможных этапов анализа данных является формулирование и проверка гипотез. Обычно это делается для оценки эффективности изменений, например, в приложении или на сайте изменили дизайн или способ взаимодействия с сервисом, и нужно ответить на вопрос, как эти изменения повлияли на пользователей. В таком случае проводят A/B-тест, делят небольшую выборку пользователей из генеральной совокупности всех посетителей сайта на 2 группы, одной группе показывают версию без изменений, другой – с изменениями. Затем анализируют поведение пользователей и значение ключевых метрик в двух группах. Выводы делают на основе статистических критериев – тестов, которые с определенной долей вероятности говорят о наличии или отсутствии различий между средними значениями двух генеральных совокупностей, между средним значением и константой, между средними значениями для зависимых выборок.

На этом этапе используются следующие инструменты:

- библиотека **`scipy`**, для работы со статистическими критериями;
- библиотека **`statsmodels`**, для работы со статистическими критериями;
- библиотека **`math`**, для работы с математическими операциями.

6. Построение прогнозных моделей

Типичной задачей анализа является построение математической модели, описывающей поведение пользователей, и что более важно, позволяющей делать предсказания об их поведении в будущем. Для реализации подобных задач используют алгоритмы машинного обучения. Входными данными в данном случае будут наборы данных – признаков поведения пользователей и список значений целевых переменных (факт оттока или возврата кредита, средний недельный чек), соответствующих этим признакам. Затем данные делятся на 3 части: обучающую, тестовую и валидационную выборки, например, в соотношении 60:20:20. Обучающая выборка используются для построения моделей, тестовая – для определения лучшей из них с помощью специальных метрик, а валидационная – для оценки работы окончательной версии модели после настройки её гиперпараметров. После обучения и настройки модель может использоваться в бизнесе для построения прогнозов.

Также, машинное обучение может применяться для построения другого рода моделей: выявления типичных групп пользователей или кластеров. Данные в этом случае стандартизуются, вычисляется матрица расстояний между отдельными записями, и наиболее близкие друг другу записи объединяются в группы. Имея подобную модель можно по-разному взаимодействовать с пользователями из разных групп.

На этом этапе используются следующие инструменты:

- библиотека **`sklearn`**, для реализации алгоритмов машинного обучения.

7. Презентация результатов

Обязательным завершающим этапом работы аналитика является представление заказчику результатов анализа. Обычно, кроме выводов и рекомендаций, готовят презентацию, которая визуализирует данные и из которой должны следовать сформулированные выводы. Содержание и способ презентации результата может быть разной в зависимости от заказчика:

- для коллег-аналитиков это может быть отчет в Jupyter Notebook с большим количеством технической информации;
- для менеджеров продукта это может быть подробный отчет в формате презентации с основными выводами и рекомендациями;
- отчет для руководителей высшего звена должен содержать ответы на конкретные вопросы.

Ещё одним результатом анализа может быть построение интерактивного и автоматически обновляемого отчета – дашборда. Особенно это полезно при необходимости построения большого количества однотипных отчетов на регулярной основе.

На этом этапе используются следующие инструменты:

- библиотеки **`dash, dash_core_components, dash_html_components`**, для построения и отображения дашбордов;
- библиотека **`plotly`**, для построения интерактивных графиков;
- сервис **Google Презентации**, для создания презентаций.