



ПЕРВИЧНАЯ ОБРАБОТКА ДАННЫХ

Данные — это абстрактные представления элементов реальности, которые служат исходным материалом для формирования знаний и понимания мира. Они лежат в основе всех информационных и вычислительных процессов, позволяя нам моделировать, понимать и взаимодействовать с миром вокруг нас. Например:

Счет в ресторане содержит данные о заказанных блюдах и их стоимости.

Фотоальбом в вашем телефоне содержит данные в виде палевных изображений и видеороликов со вчерашней вечеринки.

Резюме, отправленное работодателю, содержит данные об образовании и текущем опыте работы.

Мы окружили себя данными и уже на автомате анализируем, обрабатываем и используем их. Теперь же нашей задачей будет научить машины делать то же самое. И первое, что нам предстоит сделать на этом пути, — это выяснить, а каким образом информация от нас *передается* компьютеру. Как наши мысли, воспоминания и звуки становятся текстом, картинками и аудиофайлами?

Фундаментальной концепцией в области компьютерных наук является представление информации в числовом виде. Машины, будь то суперкомпьютеры или просто смартфоны, общаются на языке цифр, оперируя нулями и единицами. Но почему именно 0 и 1? Это связано с тем, что практически все устройства состоят из миллионов маленьких электрических элементов, где ключевую роль играют два состояния: ток есть и тока нет. Для удобства эти состояния и были обозначены как 0 и 1 🦉

Этот простой на первый взгляд двоичный код лежит в основе всех современных технологий. И несмотря на то, что данный подход не является универсальным, именно этот язык, состоящий всего лишь из двух букв, получил наиболее широкое распространение.

Так, с появлением вычислительной техники мы то и дело что занимаемся созданием цифровой среды, переводя объекты реальности в бинарный код. В процессе такой трансформации, у нас уже возникло немало технических проблем и концептуальных вопросов. Например, одним из ключевых стал вопрос взаимнооднозначного соответствия между объектами природы и их цифровыми представлениями – “Неужели все что нас окружает можно преобразовать в цифры?”.



Мы попробуем найти ответы на этот и ряд других вопросов, но сделаем это в рамках машинного обучения. И так как концепция ML начинается с данных мы поговорим вначале именно о них:

Представьте себе, что вы решили испечь торт. У вас есть множество ингредиентов: мука, сахар, яйца, масло, ваниль. Все они разные по своей природе и требуют особого обращения - яйца нужно взбить, муку — просеять,

а масло — растопить. Точно так же, как ингредиенты для торта, данные бывают разными, и их нужно правильно подготовить и использовать, чтобы получить отличный результат.

Давайте узнаем, а какие у нас есть ингредиенты для приготовления моделей. И начнем мы с того, что попробуем типизировать данные разделив их по категориям:

Числовые данные (Numerical Data) — это данные, которые можно посчитать или измерить. Они могут быть представлены в виде чисел и с ними можно осуществлять математические операции. Эти данные делятся на два типа: дискретные и непрерывные.

Дискретные данные — это данные, которые можно посчитать, но нельзя измерить. Вы можете посчитать сколько книг в вашей библиотеке 100, 200, 300, сколько машин стоит на парковке 1, 2, 20, сколько матчей в сезоне выиграла ваша команда. Но вы не сможете про них сказать, например, что ваша команда выиграла 2.5 матча, или на полке стоит 1.5 книги. Говоря на языке математики, дискретные данные отождествляются с целыми числами.

Непрерывные данные — это данные, которые можно измерить, но нельзя посчитать. Они часто используются для физических величин, которые могут быть выражены с любой степенью точности. Например, при измерении температуры в комнате с помощью обычного термометра, вы можете увидеть 22°C, но, если вы будете использовать более точный прибор, значение составит 21.567°C. Время прохождения дистанции спортсменом может быть 12.34 секунды или 12.3456 секунды. Непрерывные данные, как вы наверно уже догадались, отождествляются с вещественными (действительными) числами.

Категориальные данные (Categorical Data) — это данные, которые не могут быть выражены числами напрямую. Вместо этого они разделяются на категории или группы. Эти данные делятся на номинальные и порядковые.

Номинальные данные — это категории, которые не имеют естественного порядка. Они просто обозначают различные группы или типы. Например, цвета автомобилей: красный, синий, зелёный. Домашние

животные: кошка, собака, хомяк, попугай. Марки мобильных телефонов: Apple, Samsung, Huawei.

Порядковые данные — это категории, которые имеют естественный порядок. Например, Оценка качества обслуживания: плохо, удовлетворительно, хорошо, отлично. Размеры одежды: маленький, средний, большой, очень большой. Степень остроты соуса: "мягкий", "средний", "острый", "очень острый".

Иногда категориальные данные можно представить в виде числовых значений для удобства обработки в моделях машинного обучения. Однако важно помнить, что такие числа не имеют математического значения, и операции над ними порой не имеют никакой логики. Например:

Любимый сезон года?	Ваш уровень удовлетворенности обслуживанием?
Зима: 1	Недоволен: 2
Весна: 2	Нейтрален: 3
Лето: 3	Доволен: 4
Осень: 4	Очень доволен: 5

Однако мы не можем сказать, что зима + весна = лето или доволен > недоволен.

Отлично. Теперь для лучшего понимания дальнейшего изложения введем конкретный пример. Представьте, что вы владелец уютного итальянского ресторана на берегу побережья - "La Dolce Vita". Ваш ресторан всегда пользовался большой популярностью благодаря вкусным блюдам, приятной атмосфере и отличному обслуживанию.

Но вот уже несколько месяцев подряд вы замечаете тревожную тенденцию: количество посетителей стало уменьшаться. Некоторые постоянные клиенты приходят все реже, а новые гости не проявляют особого интереса. Вас стала беспокоить эта ситуация, и вы захотели найти ее причину.

Для начала вы решаете собрать данные о посещаемости вашего ресторана и предпочтениях клиентов. Вы уже знаете, что собирать

информацию можно двумя способами как вручную – проводя небольшие опросы среди гостей, так и автоматически – интегрировав онлайн систему анкетирования и оценок. Для лучшего эффекта вы используете оба метода.

Понимая, что данных будет слишком много вы определяете какие из них, будут иметь критическое значение для анализа, а какие нет - то есть вы выделяете фичи. Например, записывая сколько посетителю лет и какого он пола и игнорируя, к примеру тот факт - какой у него цвет волос.

После чего вы *выбираете структуру*, которая, во-первых, поможет вам упорядочить все собранные данные, а во-вторых, более качественно провести анализ. В нашем случае лучше всего подойдет обыкновенная *таблица*.

В результате чего, после нескольких недель упорного труда ваши собранные данные выглядят следующим образом:

Таблица -1 Результаты клиентского опроса ресторана La Dolce Vita

№	Дата визита	Пол	Возраст	Любимое блюдо	Оценка	Комментарии
1	04.07.24	Муж	35	Спагетти	4	Паста была пресной
2	05.07.24	Жен	28	Маргарита	3	Пицца была хорошая, но обслуживание медленное.
3	06.07.24	Муж	42	Лазанья	5	Как всегда, превосходно! Пять с плюсом
4	07.07.24	Жен	30	Цезарь	2	Салат был не свежий, официант забыл про наш столик
5	08.07.24	Муж	45	Тирамису	5	Лучший десерт в городе! Скоро приду к вам снова
6	09.07.24	Жен	25	Брускетта	4	Понравилась закуска, не хватило живой музыки
7	10.07.24	Муж	33	Эспрессо	5	Кофе был идеальный.
8	11.07.24	Муж	14	Джелато	2	Джелато отстой, горите в аду
9	12.07.24	Муж	36	Спагетти	3	Переварили макароны

Таблица 1 – это наш итоговый результат. Далее мы шаг за шагом воспроизведем все этапы ее создания и обработки. Но перед этим небольшое напутствие:

Работа с данными — это самый нудный и рутинный процесс в машинном обучении, который по иронии судьбы является главным фактором, влияющим на точность и качество получаемых результатов. Пропустить его

мы увы не можем, однако можем избавить себя от излишней монотонности, разбив процесс обучения на несколько частей.

В этой книге, как и в фильмах с параллельными сюжетными линиями, мы будем периодически переходить к другим аспектам машинного обучения, возвращаясь к теме данных по мере ее необходимости. Вам важно не терять нить повествования и не путать очередность процессов. Для этого вспоминайте путеводную звезду, которая не позволит вам сбиться с пути:

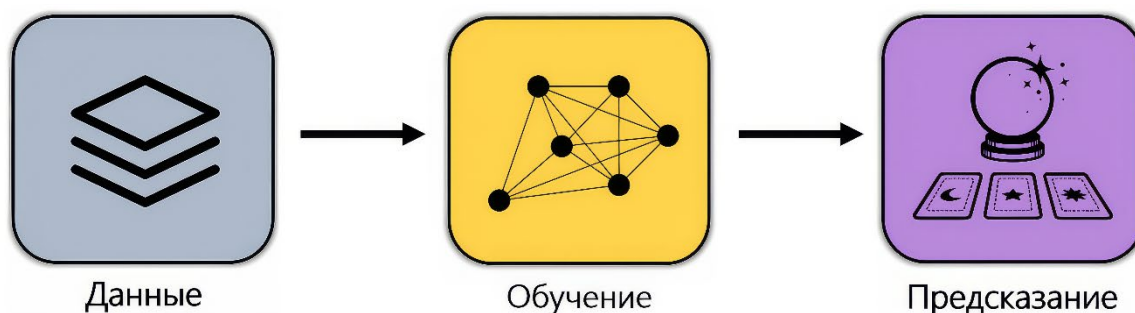


Рисунок – 3 Схематичное изображение концепции машинного обучения

Итак, возвращаемся немного назад, ресторан La Dolce Vita. После того как все сведения были собраны и упорядочены в выбранную структуру, следующим важным шагом становится их интеграция.

Интеграция (Data Integration) — это процесс объединения данных из различных источников в единый, целостный набор, пригодный для анализа.

Данные из разных источников могут существенно различаться по структуре формату и содержанию. Например, одни данные могут быть представлены в виде таблиц, тогда как информация в другом источнике может храниться в виде картинок. Приведение всего этого множества к единому знаменателю минимизирует риск возникновения ошибок и обеспечивает корректность выполнения последующего анализа.

Так как мы с вами собирали сведения двумя способами – ручным и автоматическим, представим, что у нас по итогу получилось 2 формата данных CSV и JSON. Давайте их воссоздадим у себя на компьютере:

CSV (Comma-Separated Values) — это простой текстовый формат, в котором данные представлены в виде строк и столбцов, разделённых

запятые. Часто в таком формате пренебрегают пробелами, во избежание проблем при обработке.

Создадим в папке DATA, текстовый файл *visits.txt* и перекинем в него нижележащие данные. Далее поменяем расширение на *visits.csv*

```
Дата визита,Пол,Возраст,Любимое блюдо
04.07.24,Муж,35,Спагетти
05.07.24,Жен,28,Маргарита
06.07.24,Муж,42,Лазанья
07.07.24,Жен,30,Цезарь
08.07.24,Муж,45,Тирамису
09.07.24,Жен,25,Брускетта
10.07.24,Муж,33,Эспрессо
11.07.24,Муж,14,Джелато
12.07.24,Муж,36,Спагетти
```

JSON (JavaScript Object Notation) - лёгкий и гибкий формат для представления структурированных данных, широко используемый в веб-разработке. Поступаем аналогичным образом создаем в папке текстовый файл, перекидываем данные и изменяем формат – в результате имеем *comment.json*

```
[
  {"Дата визита": "04.07.24", "Оценка": 4, "Комментарии": "Паста была пресной"},
  {"Дата визита": "05.07.24", "Оценка": 3, "Комментарии": "Пицца была хорошая, но обслуживание медленное."},
  {"Дата визита": "06.07.24", "Оценка": 5, "Комментарии": "Как всегда, превосходно! Пять с плюсом"},
  {"Дата визита": "07.07.24", "Оценка": 2, "Комментарии": "Салат был не свежий, официант забыл про наш столик"},
  {"Дата визита": "08.07.24", "Оценка": 5, "Комментарии": "Лучший десерт в городе! Скоро приду к вам снова"},
  {"Дата визита": "09.07.24", "Оценка": 4, "Комментарии": "Понравились закуски, не хватило живой музыки"},
  {"Дата визита": "10.07.24", "Оценка": 5, "Комментарии": "Кофе был идеальный."},
  {"Дата визита": "11.07.24", "Оценка": 2, "Комментарии": "Джелато отстой, горите в аду"},
  {"Дата визита": "12.07.24", "Оценка": 3, "Комментарии": "Переварили макароны"}
]
```

Далее создаем модуль с расширением .py я назвал его *Dolce_Vita.py*

Теперь чтобы продвинуться дальше нам нужно кое-что изучить. Когда вы работаете с текстовыми данными из различных источников, файлы могут быть сохранены в разных кодировках.

Кодировка – это процесс установления взаимосвязи между элементами одного множества и элементами другого, посредством определённых правил.

Когда речь заходит о кодировке, мне в голову врезается картинка пляшущих человечков из серии рассказов о Шерлоке Холмсе:



По мне это один из лучших нативных способов сказать о том, что такое кодирование информации.

Если же мы говорим о кодировке в разрезе информатики, то это процесс управления электричеством, основанный на принципах двоичного представления. Давайте более подробно рассмотрим его:

Бит (Binary digit) — это минимальная единица информации в компьютере, которая может принимать одно из двух значений: 0 или 1.

Используя только эти две цифры, мы уже можем закодировать информацию, например 1 – это ДА, 0 – это НЕТ, или 1 – ИСТИНА, 0 – ЛОЖЬ.

Но нам этого мало, мы хотим закодировать еще больше информации. Тогда мы начинаем увеличивать последовательность 0 и 1. Например если мы увеличим длину последовательности до двух знаков, то мы сможем закодировать уже 4 символа, например это могут быть буквы русского алфавита: **00** – П, **01** – А, **10** – У, **11** – М.

МАМА = [11][01][11][01]

УМ = 1011

ПАПА = [00][01][00][01]

ПУМА = 00101101

Отметим, что если не указать компьютеру какое количество бит мы выбираем для кодировки, то он мог бы разбить нашу последовательность на группы разной длины, например, по четыре или восемь бит:

$$\text{МАМА} = [1101][1101] \quad \text{ПАПА} = [00010001],$$

То есть, для правильной интерпретации данных нам важно заранее определить правило, по которому будет происходить разбиение.

Первая широко используемая кодировка символов состояла из 8 бит. Восемь бит – а сколько это?

1 бит: 2 возможных состояния (0, 1)

2 бита: 4 возможных состояния (00, 01, 10, 11)

3 бита: 8 возможных состояний (000, 001, 010, 011, 100, 101, 110, 111)

4 бита: 16 возможных состояний (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111)

Не трудно заметить, что каждый раз, добавляя новый бит, мы удваиваем количество возможных комбинаций: 2 – 4 – 8 – 16 - ... То есть общая формула расчета всех комбинаций нулей и единиц принимает следующий вид 2^N . Теперь легко посчитать количество возможных состояний восьми бит:

$$8 \text{ бит} = 2^8 = 256$$

256 различных комбинаций: 00000000, 00000001, 00000010, 00000011 и так далее до 11111111. Эта единица информации стала настолько широко используемой, что получила своё собственное название — байт.

Байт (Byte) — происходит от слова “кусать”, “часть”, — это последовательность из 8 битов.

Проверка на понимание: какое минимальное количество бит информации потребуется чтобы закодировать русский алфавит? А байт?

Ок, получается нам важно понимать, какую кодировку выбрал пользователь, потому что неверное её распознавание компьютером может

вызвать ошибки декодирования и привести к неправильному отображению текста. Это особенно актуально для данных, содержащих символы, отличные от стандартного английского алфавита, например, кириллические символы, или китайские иероглифы.

Если вы активный пользователь ПК, то наверняка уже встречались с подобными ошибками декодирования:

Ошибка: Invalid JSON response. Response text: JFIF
\$. ' ",# (7),01444 '9=82<.342 C 2!
!22 " "
} !1A Qa "q 2 #B R \$3br
&'()*456789:CDEFGHIYZcdefghi

Становится понятно, что если каждый пользователь будет придумывать свою собственную кодировку, то ни к чему хорошему это не приведет. Поэтому логичным выходом отсюда является разработка стандартизированных, шаблонных кодировок. Рассмотрим наиболее распространенные из них — это UTF-8, Windows-1251, ASCII.

ASCII (American Standard Code for Information Interchange) — это старейшая и самая простая кодировка, созданная в 1960-х годах. Она использует 7 бит для представления символов, что позволяет ей закодировать 128 уникальных символов. Хотя сама таблица ASCII определена как 7-битная, многие системы на уровне хранения и передачи данных используют 8 бит для каждого символа, заполняя восьмой бит нулём или используя его для других задач.

ASCII включает:

Английские буквы (A-Z, a-z)

Цифры (0-9)

Специальные символы (например, !, @, #, \$, %)

Управляющие символы (например, пробел, табуляция, перенос строки)

Поскольку ASCII использует только 7 бит, он не может представлять символы других алфавитов, таких как кириллица, китайские иероглифы и другие неанглийские символы.

Windows-1251 — это кодировка, разработанная для поддержки кириллических алфавитов, включая русский, украинский, болгарский и другие аналогичные языки. Использует 8 бит.

Windows-1251 включает:

Все символы ASCII (с кодами 0-127)

Кириллические буквы (с кодами 128-255)

Специальные символы и дополнительные знаки, используемые в кириллических алфавитах

UTF-8 (Unicode Transformation Format - 8-bit) - это самая распространённая и универсальная кодировка. Она является частью стандарта Unicode, который стремится включить символы всех письменных языков мира. UTF-8 использует от 1 до 4 байтов для кодирования каждого символа. По умолчанию в Python используется именно она.

UTF-8 может представлять символы из всех языков, включая:

Все символы ASCII (в 1 байт)

Кириллические, греческие, арабские буквы, китайские иероглифы и многие другие символы (в 2-4 байта)

Специальные символы, эмодзи, древние письменности и т.д.


Хорошо, теперь возвращаемся обратно к процессу интеграции данных. У нас есть 2 файла разных форматов CSV и JSON, давайте для начала их декодируем. Открываем *Dolce_Vita.py* и начинаем писать код.

Первоначально импортируем библиотеки, которые нам понадобятся:

```
import pandas as pd # библиотека для работы с табличными
данными
import json # библиотека для работы с JSON-форматом
import chardet # библиотека для определения кодировки
файлов
```

Если у вас не скачены необходимые библиотеки создайте виртуальное окружение или прям в командной строке выполните команду:

pip install pandas chardet

Далее подгружаем подготовленные файлы. Для тех, кто сталкивается с этим первый раз ищите дополнительную информацию тут 

Вначале определим кодировку CSV файла:

```
# Определение кодировки CSV-файла
with open('visits.csv', 'rb') as file:
    result = chardet.detect(file.read(10000))
print(result)
```

with open('visits.csv', 'rb') as file:

Мы открываем файл visits.csv в режиме чтения в бинарном виде ('rb'). Бинарный режим используется, потому что кодировка определяется по байтам, а не по символам. Конструкция *with* автоматически закроет файл после завершения работы с ним.

file.read(10000):

Читаем первые 10,000 байт файла. Это делается, чтобы проанализировать часть файла и на основе этих данных попытаться определить его кодировку.

chardet.detect()

Мы передаем байты файла в функцию detect() из библиотеки chardet, которая возвращает словарь с информацией о предполагаемой кодировке.

Запустив код, вы сможете увидеть аналогичный ответ:

```
{'encoding': 'windows-1251', 'confidence': 0.875241684948003, 'language': 'Russian'}
```

encoding: показывает, какая кодировка была определена. Видим, что наша кодировка *windows-1251*

confidence: вероятность правильного определения кодировки.

language: если возможно, указывает предполагаемый язык текста.

Теперь зная кодировку, мы можем загрузить данные из CSV файла. В этом нам поможет библиотека `pandas`. В ней есть такая структура данных, которая называется **DataFrame**. Она представляет собой двумерную таблицу, подобную таблице в Excel, с индексами для строк и столбцов.

Давайте вначале на нее посмотрим, а потом дадим еще несколько комментариев.

```
# Загрузка данных из CSV-файла с указанием кодировки
visits_dv = pd.read_csv('visits.csv',
encoding=result['encoding'])

# Просмотр загруженных данных
print(visits_dv.head()) #или print(visits_dv)
```

Создаем переменную `visits_dv` далее используем метод из библиотеки `pandas` для прочтения CSV-файла. В скобках указываем имя файла и его кодировку.

Так как часто файлы содержат большое количество строчек, у нас нет необходимости проверять каждую из них, нам достаточно прочесть лишь их корректное отображение. Для этого в библиотеке `pandas` есть функция `head()` которая по умолчанию выводит первые 5 строк `DataFrame`.

В нашем случае мы в принципе можем ей пренебречь так как знаем, что наш файлик небольшой. Запускаем код и видим:

	Дата визита	Пол	Возраст	Любимое блюдо
0	04.07.24	Муж	35	Спагетти
1	05.07.24	Жен	28	Маргарита
2	06.07.24	Муж	42	Лазанья
3	07.07.24	Жен	30	Цезарь
4	08.07.24	Муж	45	Тирамису

Рисунок 4 – Исходные данные о посетителях из CSV-файла в структуре DataFrame

Замечаем, что в структуре `DataFrame` по умолчанию происходит нумерация строк от 0 до `n-1` и в качестве шапки берется первая строчка CSV файла.

Аналогичным образом поступаем и для файла JSON. Вот итоговый код:

```
import pandas as pd # библиотека для работы с табличными данными
import json # библиотека для работы с JSON-форматом
import chardet # библиотека для определения кодировки файлов

# Определение кодировки CSV-файла
with open('visits.csv', 'rb') as file:
    result = chardet.detect(file.read(10000))
    print(f"CSV file encoding: {result['encoding']}")

# Загрузка данных из CSV-файла с указанием кодировки
visits_dv = pd.read_csv('visits.csv', encoding=result['encoding'])

# Определение кодировки JSON-файла
with open('comment.json', 'rb') as file:
    result = chardet.detect(file.read(10000))
    print(f"JSON file encoding: {result['encoding']}")

# Загрузка данных из JSON-файла с указанием кодировки
comments_dv = pd.read_json('comment.json',
encoding=result['encoding'])

# Просмотр загруженных данных
print(visits_dv.head())
print()
print(comments_dv.head())
```

CSV file encoding: windows-1251

JSON file encoding: windows-1251

	Дата визита	Пол	Возраст	Любимое блюдо
0	04.07.24	Муж	35	Спагетти
1	05.07.24	Жен	28	Маргарита
2	06.07.24	Муж	42	Лазанья
3	07.07.24	Жен	30	Цезарь
4	08.07.24	Муж	45	Тирамису

	Дата визита	Оценка	Комментарии
0	04.07.24	4	Паста была пресной
1	05.07.24	3	Пицца была хорошая, но обслуживание медленное.
2	06.07.24	5	Как всегда, превосходно! Пять с плюсом
3	07.07.24	2	Салат был не свежий, официант забыл про наш ст...
4	08.07.24	5	Лучший десерт в городе! Скоро приду к вам снова

Отлично, теперь давайте объединим данные. Наиболее распространенный способ объединения данных — это объединение по

общему ключу. Ключ представляет собой поле или набор полей, которые однозначно идентифицируют записи в каждом из наборов данных.

В нашем случае это общим ключом выступает *Дата визита*. Давайте уберем абсолютно все `print()` и добавим в наш код в самом конце вот такое продолжение:

```
merged_dv = pd.merge(visits_dv, comments_dv, on='Дата визита', how='inner')  
  
print(merged_dv)
```

	Дата визита	Пол	...	Оценка	Комментарии
0	04.07.24	Муж	...	4	Паста была пресной
1	05.07.24	Жен	...	3	Пицца была хорошая, но обслуживание медленное.
2	06.07.24	Муж	...	5	Как всегда, превосходно! Пять с плюсом
3	07.07.24	Жен	...	2	Салат был не свежий, официант забыл про наш ст...
4	08.07.24	Муж	...	5	Лучший десерт в городе! Скоро приду к вам снова
5	09.07.24	Жен	...	4	Понравилась закуска, не хватило живой музыки
6	10.07.24	Муж	...	5	Кофе был идеальный.
7	11.07.24	Муж	...	2	Джелато отстой, горите в аду
8	12.07.24	Муж	...	3	Переварили макароны

Видим, что данные объединились, однако отображаются не все. Python по умолчанию показывает только определенное количество строк и столбцов, чтобы избежать переполнения экрана. Если вы хотите отобразить всю таблицу, можно использовать метод преобразования `to_string()`. Для этого в `print` допишите `print(merged_dv.to_string())`

	Дата визита	Пол	Возраст	Любимое блюдо	Оценка	Комментарии
0	04.07.24	Муж	35	Спагетти	4	Паста была пресной
1	05.07.24	Жен	28	Маргарита	3	Пицца была хорошая, но обслуживание медленное.
2	06.07.24	Муж	42	Лазанья	5	Как всегда, превосходно! Пять с плюсом
3	07.07.24	Жен	30	Цезарь	2	Салат был не свежий, официант забыл про наш столик
4	08.07.24	Муж	45	Тирамису	5	Лучший десерт в городе! Скоро приду к вам снова
5	09.07.24	Жен	25	Брускетта	4	Понравилась закуска, не хватило живой музыки
6	10.07.24	Муж	33	Эспрессо	5	Кофе был идеальный.
7	11.07.24	Муж	14	Джелато	2	Джелато отстой, горите в аду
8	12.07.24	Муж	36	Спагетти	3	Переварили макароны

Для объединения файлов мы использовали метод `merge()`. Этот метод позволяет объединить только два `DataFrame` за раз. Параметр `on` указывает на поле, по которому происходит объединение. Параметр `how` указывает на тип объединения.

Для первого погружения думаю этой информации более чем достаточно. Если вам этого было мало не переживайте мы еще долго будем возиться с данными, и вы еще успеете от них устать.

ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

1. Объясните разницу между дискретными и непрерывными типами данных на примерах из реальной жизни?
2. Что такое интеграция данных и связан ли этот процесс с кодировкой? Если да, то как? Если нет, то почему?
3. Назовите 3 самые распространенные кодировки, в чем их принципиальное отличие?

ПРАКТИЧЕСКИЕ ЗАДАЧИ

Дан список отзывов клиентов в виде строк. Напишите функцию `analyze_reviews(reviews)`, которая подсчитает количество положительных и отрицательных отзывов. Функция должна вернуть два значения: количество положительных отзывов и количество отрицательных отзывов.

Положительные ключевые слова: "хорошо", "отлично", "прекрасно", "превосходно", "лучший"

Отрицательные ключевые слова: "плохо", "ужасно", "отвратительно", "никогда"

```
# Пример использования
reviews = [
    "Отличная еда, все было прекрасно!",
    "Обслуживание было ужасным, больше не приду.",
    "Хорошо приготовлено, но могло быть лучше.",
    "Отвратительно! Никогда больше сюда не пойду.",
    "Все было превосходно, лучший ресторан!"
]

positive, negative = analyze_reviews(reviews)
print(positive) # Ожидаемый результат: 3
print(negative) # Ожидаемый результат: 2
```


Ответ:

```
def analyze_reviews(reviews):
    positive_keywords = ["хорошо", "отлично", "прекрасно", "превосходно",
                        "лучший"]
    negative_keywords = ["плохо", "ужасно", "отвратительно", "никогда"]

    positive_count = 0
    negative_count = 0

    for review in reviews:
        review_lower = review.lower()
        if any(word in review_lower for word in positive_keywords):
            positive_count += 1
        if any(word in review_lower for word in negative_keywords):
            negative_count += 1

    return positive_count, negative_count

# Пример использования
reviews = [
    "Отличная еда, все было прекрасно!",
    "Обслуживание было ужасным, больше не приду.",
    "Хорошо приготовлено, но могло быть лучше.",
    "Отвратительно! Никогда больше сюда не пойду.",
    "Все было превосходно, лучший ресторан!"
]

positive, negative = analyze_reviews(reviews)
print(positive) # Ожидаемый результат: 3
print(negative) # Ожидаемый результат: 2
```

ЗАДАЧИ НА ЛОГИКУ

1. Попробуйте отгадать, какое слово тут закодировано:

12 16 15 17 13 21 19 6 20 18

2. Как можно определить, что оценка, оставленная посетителем, является некорректной или была выбрана случайным образом, без должного обоснования?

ДОПОЛНИТЕЛЬНЫЙ МАТЕРИАЛ

1. Архитектура ЭВМ <https://www.youtube.com/watch?v=dVZrHGNGvb0>