Основы искусственного интеллекта

Техники очистки данных. Понимание различных типов данных (структурированные, неструктурированные).

Инструменты для анализа данных

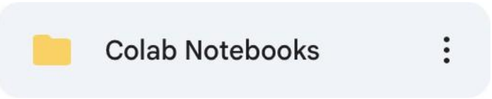
- Google Collab: https://colab.research.google.com/

- Google Sheets: https://docs.google.com/spreadsheets

Работа с Google Collab

Google Drive

- Ваши блокноты (файлы, где вы пишете код) автоматический сохраняется в Google Drive в папке “Colab Notebooks”



- Файлы “train.csv” нужно подгружать вручную в каждой сессии

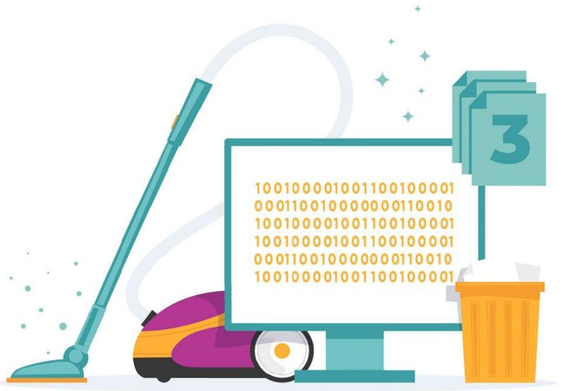
Набор данных для использования

- Набор данных - Титаник

- https://www.kaggle.com/c/titanic/data
- Необходимо загрузить набор в Google Collab

Чистка набора данных Титаник с помощью Pandas

- **Цель**: понять, как очистить и предварительно обработать набор данных «Титаник» с помощью pandas.
- **Важность**. Чистка данных — важнейший этап любого анализа данных или в подготовке данных для машинного обучения. Набор данных «Титаник» представляет собой реальный пример обработки данных.



Библиотеки для работы с данными

- **Pandas** — это программная библиотека, написанная для языка программирования Python для манипулирования и анализа данных.

- В частности, он предлагает структуры данных и операции для управления числовыми таблицами и временными рядами.

Чистка и подготовка данных

- Обработка пропущенных значений 
- Кодирование категориальных переменных 
- Стандартизация форматов данных
- Нормализация числовых значений

Обработка недостающих данных

- **Шаг 1.** Определить недостающие значения.

```
df.isnull().sum()
```

- **Шаг 2.** Определить стратегию обработки пропущенных значений:
  - Удалить строки/столбцы.
  - Заполните недостающие значения средним значением, медианой, модой или заполнителем.

Среднее значение, Медиана, и Мода

- **Среднее значение** — это среднее арифметическое, которое вычисляется путем сложения набора чисел с последующим делением полученной суммы на их количество. Например, средним значением для чисел 2, 3, 3, 5, 7 и 10 будет 5, которое является результатом деления их суммы, равной 30, на их количество, равное 6.

```
df['Age'].mean()
```

Среднее значение, Медиана, и Мода

- **Медиана** — это число, которое является серединой множества чисел, то есть половина чисел имеют значения большие, чем медиана, а половина чисел имеют значения меньшие, чем медиана. Например, медианой для чисел 2, 3, 3, 5, 7 и 10 будет 4.

- **Мода** — это число, наиболее часто встречающееся в данном наборе чисел. Например, модой для чисел 2, 3, 3, 5, 7 и 10 будет 3.

```
df['Age'].median()
```

```
df['Embarked'].mode()
```

Преобразование типов данных

- Преобразование в числовое значение (Ярлычное кодирование)
- Горячее кодирование/Быстрое кодирование (One-hot encoding)

Ярлычное кодирование

ПРОДУКТ	КАТЕГОРИЯ	ЭНЕРГЕТИЧЕСКАЯ ЦЕННОСТЬ
Яблоко	1	52
Курица	2	165
Брокколи	3	40

Быстрое кодирование

ЯБЛОКО	КУРИЦА	БРОККОЛИ	ЭНЕРГЕТИЧЕСКАЯ ЦЕННОСТЬ
1	0	0	52
0	1	0	165
0	0	1	40

Создание новых признаков
Feature Engineering

- Создайте новые значимые функции (например, ‘Величина семьи’, ‘Одинокий пассажир’).

```
                # братьев и сестер / супругов на борту
                # родителей/детей на борту'] + 1
```

```
# Создание признака ‘Величина семьи’

df['Величина семьи'] = df['SibSp'] + df['Parch'] + 1
```

```
# Создание признака ‘Одинокий пассажир'

df['Одинокий пассажир'] = (df['Величина семьи'] ==
1).astype(int)
```

Одинокий пассажир
False
False
True
False
True

Одинокий пассажир
0
0
1
0
1

Эквивалентность True/False и 1/0

Тип boolean (bool)

- В программировании **`True`** эквивалентно **`1`**, а **`False`** эквивалентно **`0`**.
- Это важно при обработке данных, например, при кодировании категориальных переменных в машинном обучении.

Логическое значение	Числовое представление
True	1
False	0

Эквивалентность True/False и 1/0

```
print(int(True))# Выведет 1

print(int(False)) # Выведет 0


print(bool(1))    # True

print(bool(0))    # False
```

Масштабирование и нормализация

- Нормализуйте или масштабируйте числовые характеристики, такие как Возраст `Age` и ‘Стоимость проезда пассажира’ `Fare`, для моделей ML, чувствительных к величине функций.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df[['Age', 'Fare']] = scaler.fit_transform(df[['Age',
Fare']])
```

Окончательно очищенный набор данных

- Проверьте очищенный набор данных.
- Сохраните очищенный набор данных для дальнейшего использования.

```
print(df.head())

df.to_csv('cleaned_titanic.csv', index=False)
```

Окончательно очищенный набор данных

- Сохраненный файл хранится во временной файловой системе Colab в каталоге /content/.
- Вы можете проверить его местоположение, выполнив:

```
import os

os.listdir('/content/')
```

Чистка и подготовка данных

- Обработка пропущенных значений 
- Кодирование категориальных переменных 
- Нормализация числовых значений 
- Стандартизация форматов данных

Стандартизация формата данных

Преобразования даты

```
import pandas as pd

df = pd.read_csv('data.csv')

df['Date'] = pd.to_datetime(df['Date'])

print(df.to_string())
```

Консоль:

https://www.w3schools.com/python/pandas/trypandas.asp?filename=demo_pandas_cleaning_to_datetime

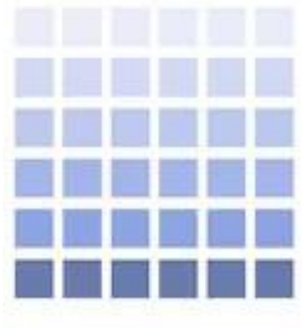
	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5

Введение

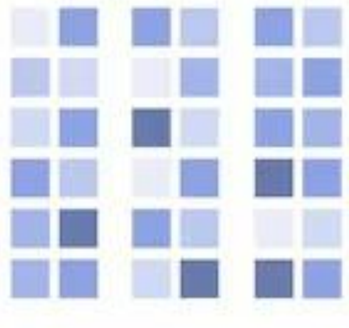
- Данные являются основой цифровой эпохи.
- Компании, организации и исследователи ежедневно сталкиваются с различными типами данных.
- Понимание их классификации и особенностей помогает эффективно обрабатывать, анализировать и использовать данные в бизнесе, науке и технологиях.

Основные типы данных

- Структурированные данные
- Неструктурированные данные
- Полуструктурированные данные



**Структурирова
нные данные**



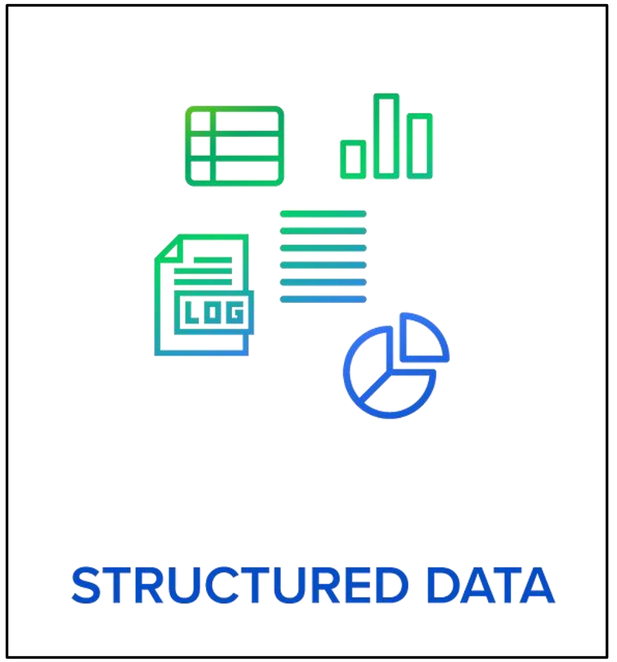
**Полуструктурир
ованные
данные**

**Неструктурир
ованные
данные**

Структурированные типы данных

Определение и Характеристики

- **Структурированные данные** – это данные, организованные по строгим правилам, обычно в формате таблиц, баз данных или других форматов с фиксированной структурой.
- **Характеристики:**
  - Организованы в строки и столбцы
  - Обладают четко определенными типами (числа, строки, даты и др.)
  - Высокая степень управляемости и предсказуемости



Примеры структурированных типов данных

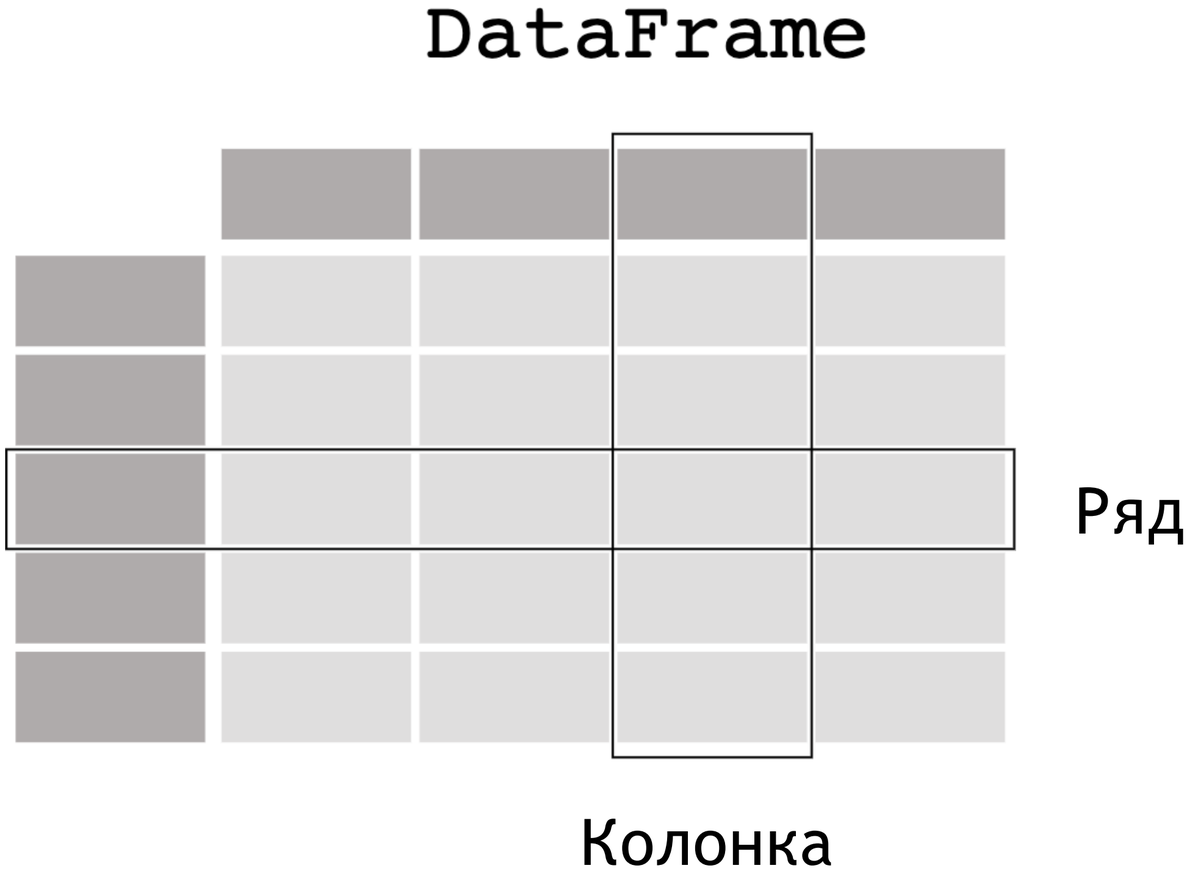
- Таблицы базы данных (например, CRM-системы, ERP-системы).
- Электронные таблицы (Excel, Google Sheets).
- Списки товаров и клиентов.
- Финансовые отчеты и транзакционные данные.

Работа с Pandas

- В **Pandas** структурированные данные обычно представлены в виде **DataFrame**

```
import pandas as pd



# Создание DataFrame

data = {'Имя': ['Анна', 'Борис', 'Виктор'], 'Возраст': [25, 30,
35], 'Город': ['Москва', 'СПб', 'Казань']}

df = pd.DataFrame(data)

print(df)
```

Представление таблицы данных pandas

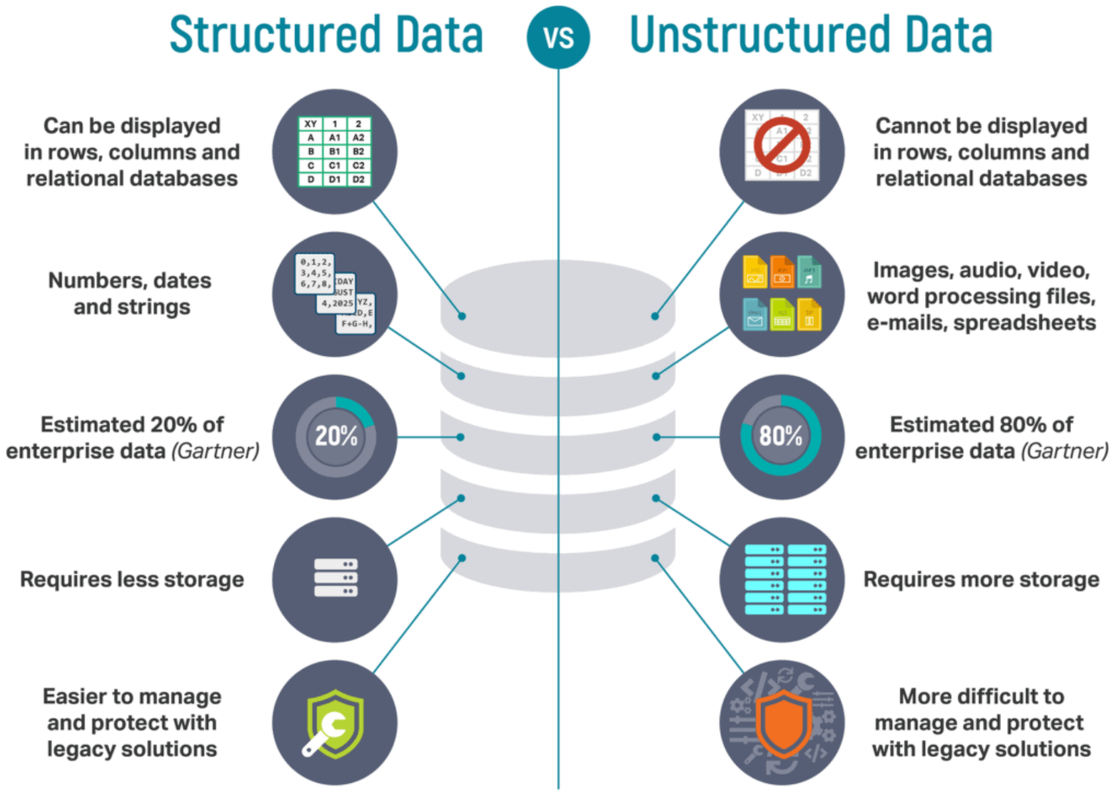


Представление таблицы данных pandas

Индекс	Имя	Возраст	Город
1	Анна	25	Москва
2	Борис	30	СПб
3	Виктор	35	Казань

Преимущества структурированных данных

- Высокая скорость обработки запросов.
- Простота поиска, фильтрации и анализа.
- Поддержка строгих правил валидации.



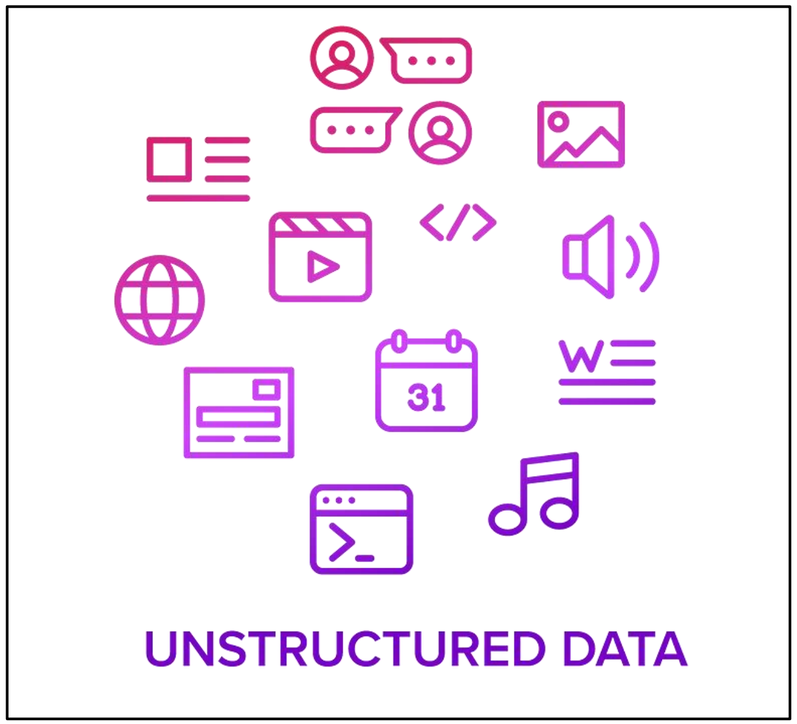
Недостатки структурированных данных

- Ограниченная гибкость – сложно обрабатывать сложные и изменяющиеся данные.
- Трудности с обработкой больших объемов информации, содержащей разнообразные форматы.

Неструктурированные типы данных

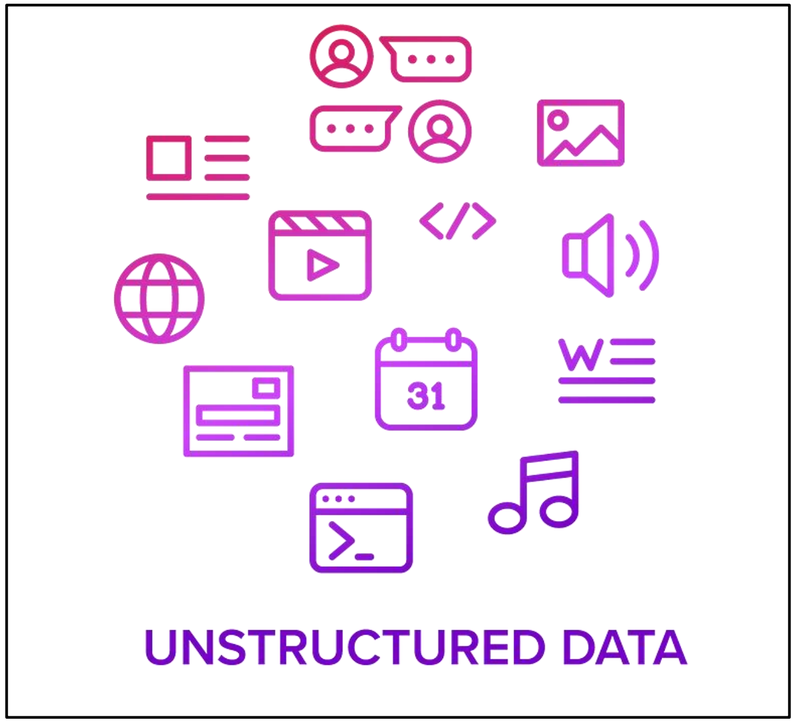
Определение и Характеристики

- **Неструктурированные данные** – это данные, не имеющие фиксированной структуры. Они могут быть представлены в различных форматах, таких как текст, изображения, видео, аудио и другие мультимедийные файлы.
- **Характеристики:**
  - Хранятся в облачных хранилищах, файловых системах и NoSQL базах данных.
  - Не подчиняются фиксированным схемам.
  - Сложны в обработке и анализе.
  - Могут содержать разнообразные форматы данных (текст, изображения, аудио, видео).



Примеры неструктурированных типов данных

- Электронная почта.
- Документы (PDF, Word, TXT).
- Социальные сети (посты, комментарии, изображения, видео).
- Лог-файлы серверов.
- Медиафайлы (аудио, видео, фотографии).



Работа с Pandas

- Хотя **Pandas** предназначен в основном для структурированных данных, он может использоваться для работы с неструктурированными данными, например, при анализе текстовых данных:

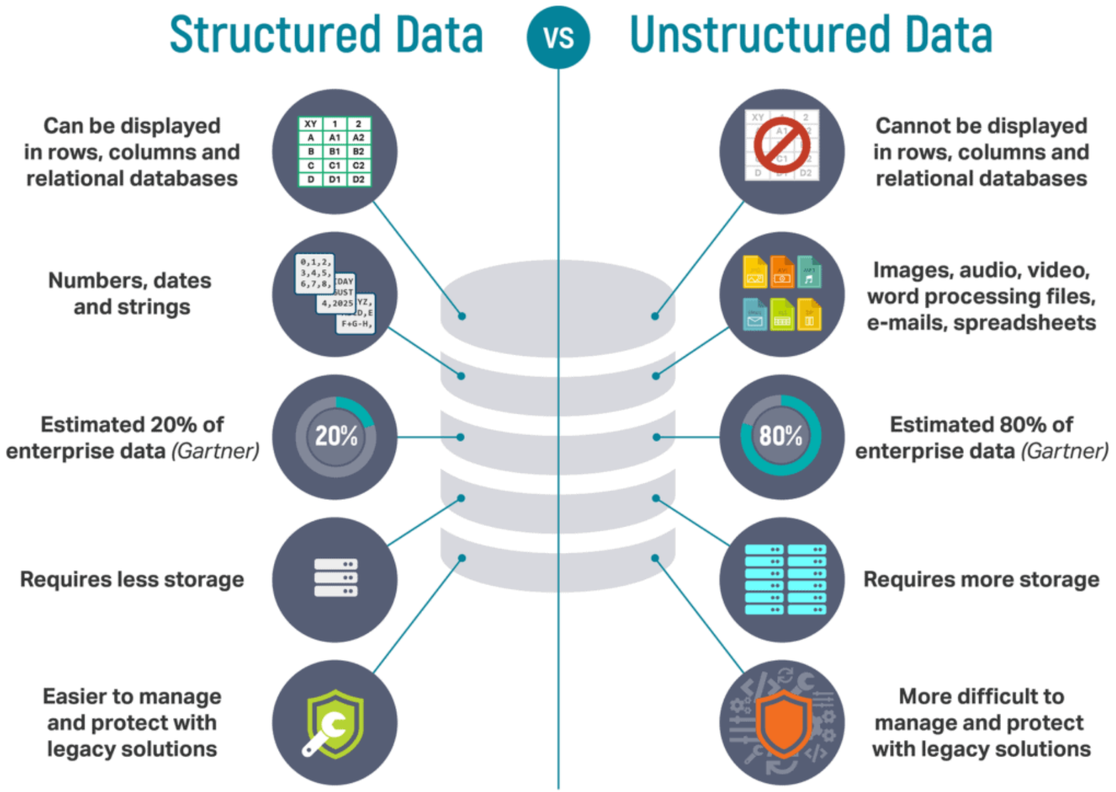
```
import pandas as pd

# Чтение текстового файла

df = pd.read_csv('example.txt', delimiter='\t')

print(df.head())
```

Преимущества неструктурированных типов данных

- Гибкость – можно работать с разными форматами данных.
- Хранение более сложной и разнообразной информации.
- Поддержка искусственного интеллекта и машинного обучения для анализа.

Недостатки неструктурированных типов данных

- Сложность обработки и анализа.
- Большие объемы данных требуют значительных вычислительных ресурсов.
- Отсутствие стандартных инструментов обработки.



Полуструктурированные типы данных

Определение и Характеристики

- **Полуструктурированные данные** занимают промежуточное положение между структурированными и неструктурированными данными. Они содержат элементы структуры, но не подчиняются строгой схеме.
- **Характеристики**
  - Используют теги, метаданные и маркеры.
  - Хранятся в NoSQL базах данных.
  - Могут легко преобразовываться в структурированные данные.

Примеры

- JSON и XML файлы.
- Логи веб-серверов.
- Данные из API.

Работа с Pandas

- Pandas поддерживает работу с JSON и XML:

```
import pandas as pd



# Чтение JSON
df = pd.read_json('data.json')
print(df.head())
```

Преимущества и недостатки полуструктурированныхданных

- **Преимущества**
  - Гибкость и возможность хранения сложных данных.
  - Хорошо подходят для работы с большими данными и веб-сервис
- **Недостатки**
  - Сложнее в обработке, чем структурированные данные.
  - Требуют дополнительных инструментов анализа и обработки.

Сравнительная таблица

Тип данных	Структурированные	Полуструктурированные	Неструктурированные
Формат хранения	Таблицы, базы данных	JSON, XML, логи	Текст, изображения, аудио, видео
Гибкость	Низкая	Средняя	Высокая
Скорость обработки	Высокая	Средняя	Низкая
Поддержка анализа	Простая	Умеренно сложная	Трудоемкая
Применение	Бизнес-аналитика, CRM	API, IoT, веб-данные	Социальные сети, мультимедиа

Спасибо за внимание!