report

7-1-pandas report

1. Pandas 공식문서리뷰

1. pandas의 핵심기능들

다양한 형식의 데이터를 불러와서 dataframe을 구성하고 select하거나 재구성 하는등 다양한 기능들을 예시와 함께 살펴볼 수 있었습니다.

2. R, SQL 등 다른 도구들과의 차이점

R, SQL에서의 기능들이 pandas에서는 어떻게 구현되는지 비교해보고 편리함과 자율성을 확인할 수 있었습니다.

3. 시각화

chart, plot 등 다양한 시각화 도구의 기능을 확인할 수 있었습니다.

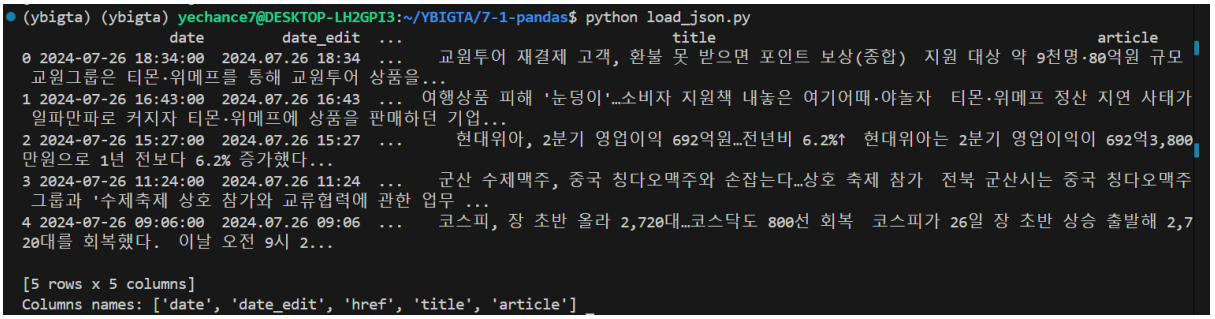
2. 한경 기사들 pandas로 불러오기

```
import pandas as pd
import json

file_path = '/home/yechance7/YBIGTA/5-2-Web/results.json'

# Load the JSON file into a pandas DataFrame
df = pd.read_json(file_path)

print(df.head())
print("Columns names:", df.columns.tolist())
```



3. 데이터 저장 포맷

IO는 Input/output의 줄임말로써 pandas는 다양한 형태의 파일 포맷의 입출력 기능을 지원합니다.

1. Pickle

- **특징**: 파이썬 객체를 직렬화(serialization)하여 파일에 저장하고, 이를 다시 역직렬화(deserialization)하여 원래의 객체로 복원할 수 있습니다.
- **필요/맥락**: 파이썬 객체를 파일로 저장하고 다시 불러와야 할 때 사용됩니다. 데이터 분석 작업 중간에 상태를 저장하거나 모델을 저장할 때 유용합니다.
- **사용 예**:
  - 데이터 분석 중간 결과 저장
  - 머신러닝 모델 저장

직렬화(Serialization)와 역직렬화(Deserialization)

- **직렬화**: 데이터 구조나 객체 상태를 저장하거나 전송하기 위해 이진 또는 텍스트 형식으로 변환하는 과정.
- **역직렬화**: 직렬화된 데이터를 원래의 데이터 구조나 객체로 복원하는 과정.
- **필요성**:
  - 데이터의 영속성 유지 (예: 파일 저장, 데이터베이스 저장)
  - 데이터 전송 (예: 네트워크 통신, 원격 프로시저 호출)
- **사용 예**:
  - 객체 저장 (Pickle)
  - 데이터 전송 (JSON, XML)

2. CSV, TSV

- **CSV (Comma-Separated Values)**: 데이터의 각 필드를 쉼표로 구분하는 텍스트 파일 포맷.
- **TSV (Tab-Separated Values)**: 데이터의 각 필드를 탭으로 구분하는 텍스트 파일 포맷.
- **특징**: 간단하고 널리 사용되는 포맷으로, 다양한 애플리케이션과 호환됩니다. 사람이 읽고 쓰기 쉬움.
- **필요/맥락**: 데이터 교환, 간단한 데이터 저장, 로깅 등에 사용됩니다.
- **사용 예**:
  - 데이터 수집 및 교환
  - 로깅 및 간단한 데이터 저장

3. JSON

- **JSON (JavaScript Object Notation)**: 키-값 쌍으로 데이터를 저장하는 경량 데이터 교환 포맷.
- **특징**: 사람이 읽기 쉽고, 대부분의 프로그래밍 언어에서 쉽게 파싱할 수 있습니다. 계층적 데이터 구조를 표현할 수 있습니다.
- **필요/맥락**: 웹 애플리케이션에서 서버와 클라이언트 간 데이터 교환에 주로 사용됩니다.
- **사용 예**:
  - 웹 API 응답/요청

    - 설정 파일
    - 데이터 저장 및 전송

4. HTML

- **특징**: 웹 페이지를 구성하는 마크업 언어. 데이터는 표 형식으로 HTML 테이블 내에 저장될 수 있습니다.
- **필요/맥락**: 웹 페이지에 데이터를 표시하거나, 웹에서 데이터를 스크래핑할 때 사용됩니다.
- **사용 예**:
    - 웹 페이지 데이터 표출
    - 웹 스크래핑
- **형태**: 마크업 언어, 웹 페이지 표 형태
- **예시**:

```
<!DOCTYPE html>
<html>
<head>
    <title>Example Table</title>
</head>
<body>
    <table border="1">
        <tr>
            <th>Name</th>
            <th>Age</th>
            <th>City</th>
        </tr>
    </table>
</body>
</html>
```

5. XML

- **XML (eXtensible Markup Language)**: 데이터를 태그로 감싸서 구조화된 형식으로 저장하는 마크업 언어.
- **특징**: 데이터를 계층적으로 표현할 수 있으며, 사람과 기계가 모두 읽기 쉽습니다.

- **필요/맥락**: 데이터 전송 및 저장, 특히 시스템 간의 데이터 교환에서 사용됩니다.
- **사용 예**:
  - 데이터 교환 (예: SOAP 프로토콜)
  - 설정 파일
- **형태**: 마크업 언어, 태그로 구분된 데이터
- **예시**:

```
<people>
    <person>
        <name>Alice</name>
        <age>30</age>
        <city>New York</city>
    </person>
    <person>
        <name>Bob</name>
        <age>25</age>
        <city>Los Angeles</city>
    </person>
    <person>
        <name>Carol</name>
        <age>28</age>
        <city>Chicago</city>
    </person>
</people>
```

6. Parquet

- **특징**: 열 지향 저장 포맷으로, 대량의 데이터를 효율적으로 저장하고 빠르게 읽어들일 수 있습니다. Hadoop 에코시스템에서 자주 사용됩니다.
- **필요/맥락**: 대규모 데이터 분석 및 처리에 최적화되어 있으며, Spark와 같은 분산 처리 시스템에서 주로 사용됩니다.
- **사용 예**:
  - 대규모 데이터 저장 및 분석
  - Spark 및 Hadoop 환경에서의 데이터 처리

7. YAML

- **YAML (YAML Ain't Markup Language)**: 사람이 쉽게 읽고 쓸 수 있는 데이터 직렬화 포맷.
- **특징**: 간결하고, 중첩된 데이터 구조를 쉽게 표현할 수 있습니다.
- **필요/맥락**: 설정 파일 및 데이터 직렬화에 주로 사용됩니다.
- **사용 예**:
  - 설정 파일 (예: Docker, Kubernetes)
  - 데이터 직렬화
- **형태**: 텍스트 파일, 계층적 구조
- **예시**:

```
- name: Alice
  age: 30
  city: New York
- name: Bob
  age: 25
  city: Los Angeles
- name: Carol
  age: 28
  city: Chicago
```

8. TOML

- **TOML (Tom's Obvious, Minimal Language)**: 간단하고 명확하게 읽기 위한 설정 파일 포맷.
- **특징**: 사람이 읽기 쉽게 설계되었으며, 데이터를 계층적으로 표현할 수 있습니다.
- **필요/맥락**: 설정 파일에 주로 사용됩니다.
- **사용 예**:
  - 설정 파일 (예: Python의 pyproject.toml)
- **형태**: 텍스트 파일, 간결한 설정 파일 포맷
- **예시**

```
[[people]]
name = "Alice"
age = 30
city = "New York"

[[people]]
name = "Bob"
age = 25
city = "Los Angeles"

[[people]]
name = "Carol"
age = 28
city = "Chicago"
```