

빅데이터 분석&시각화

교과목 전체 목차

1. 데이터 분석 이해

- ▣ 데이터 분석 기초
- ▣ 데이터 분석 파이썬 라이브러리

2. 통계 분석결과 시각화

- ▣ Matplotlib 차트 그리기
- ▣ Seaborn 차트 그리기
- ▣ 기술 통계 분석과 시각화
- ▣ 상관 분석 + 히트맵 시각화

3. 텍스트 빈도 분석과 시각화

- ▣ Wordcloud, pytagcloud 시각화 생성하기
- ▣ 영문 분석(NLTK) 및 시각화
- ▣ 한글 분석(KoNLPy) 및 시각화

4. 지리정보 분석과 시각화

- ▣ Forium 사용 지도 생성하기
- ▣ 지리정보 분석 후 맵 생성하기
- ▣ [행정 구역 데이터 분석+choropleth] 행정구역별 의료기관 현황분석

5. 다양한 데이터 분석 및 시각화 방법

- 4. 데이터 수집에 도움되는 사이트
- 5. 데이터 분석 및 시각화 대쉬보드 툴 (tableau)

6. Django 시각화 결과 웹 배포

- ▣ Django
- ▣ 웹프로그래밍

1장. 데이터 분석의 이해

1. 데이터 분석 기초

1. 데이터 수집의 이해
2. 데이터 전처리 이해
3. 데이터 분석의 이해

2. 파이썬 데이터 분석 라이브러리

1. numpy
2. pandas

1. 데이터 분석 기초

빅데이터 처리 과정



1. 데이터 분석 기초

□ 데이터 수집 이해

▣ 데이터 선정

▣ 데이터 수집 방법 선정

▣ 데이터 수집 도구 선정



(1) 수집할 데이터 선정

(예) 축구 승리에 영향을 미치는 요인이 있을까?

▣ 축구와 직접 관련된 데이터

- 축구 기록 관련: 포메이션, 볼점유율, 패스성공률, 슈팅시도, 태클 등
- 축구 선수 관련: 연봉, 최근 경기 성적, 평균 나이 등
- 그 외: 홈원정 여부, 경기시작 시간(낮·밤), 직전 경기의 승패 여부 등

▣ 축구와 직접 관련이 없는 데이터

- 경기 당일의 날씨
- 관중의 수, 응원단 유무
- 유니폼의 색깔
- 그 외 관련이 있을 것 같은 데이터

1. 데이터 분석 기초

□ 데이터 수집 이해

- 데이터 선정
- **데이터 수집 방법 선정**
- 데이터 수집 도구 선정



(2) 데이터 수집 방법 선정

□ 데이터를 어디서 구할 수 있는가?

- 어디서 제공해주는가?
- 제공한다면 무료인가?
- 다운로드하는 파일 형식은 어떠한가?
- 수집한 데이터를 사용해도 법적으로 문제가 없는가?

□ 데이터를 쉽게 구할 수 없다면?

API나 CSV 파일 등 쉽게 다운로드할 수 없다면?

- 웹 크롤링 등으로 필요한 데이터를 수집할 수 있는가?
- 다른 데이터를 활용하여 필요한 데이터를 생성할 수 있는가?
- (예) 일자별 최고 기온 데이터가 없는 대신 시간대별 기온 데이터가 있다면, 직접 일자 별 최고 기온 데이터를 생성할 수 있음

1. 데이터 분석 기초

□ 데이터 수집 이해

- ▣ 데이터 선정
- ▣ **데이터 수집 방법 선정**
- ▣ 데이터 수집 도구 선정



(2) 데이터 수집방법 선정

▣ 수집할 데이터의 양은 충분한가?

- 의미 있는 결과를 얻으려면 수집할 데이터는 한쪽에 편향되지 않아야 하며, 충분히 양이 많아야 함

(예) 평균 이용량이 가장 많은 지하철 호선은?

1호선은 7월, 2호선은 10월, 3호선은 6월, 학생들의 방학, 휴가철 등 여러가지 변수가 있기 때문에 균형 있고 충분한 데이터를 수집한 뒤 분석을 해야함

▣ 신뢰할 만한 데이터인가?

- 수집한 데이터가 신뢰할 만한 데이터인지 구별하는 것이 중요해 짐

(예) 출처가 불분명한 데이터 : 지식IN, 카페 글 등

1. 데이터 분석 기초

□ 데이터 수집 이해

- ▣ 데이터 선정
- ▣ 데이터 수집 방법 선정
- ▣ **데이터 수집 도구 선정**



(3) 데이터 수집도구 선정

▣ 일회성 데이터인가? 주기적으로 데이터를 수집해야 하는가?

- 일회성 데이터 : 수집한 데이터를 CSV, TXT 등의 파일형식으로 저장하여 활용하거나 **필요할 때마다 데이터를 수집**하여 그때그때 활용해도 됨

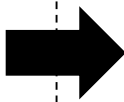
(예) 변하지 않는 데이터 : 국가, 도시 정보 등

- 주기적으로 수집이 필요한 데이터: 필요할 때마다 데이터를 처음부터 끝까지 수집하기에 어려움이 있으므로 **데이터베이스 활용, 자동화 시스템 구축**을 통해 주기적으로 **데이터를 수집·저장하는 과정**이 필요함

(예) 변하는 데이터 : 시계열 데이터

1. 데이터 분석 기초

□ 데이터 수집 이해

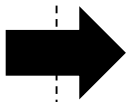
- ▣ 데이터 선정
- ▣ 데이터 수집 방법 선정
- ▣ **데이터 수집 도구 선정** 

(3) 데이터 수집도구 선정

- ▣ 수집한 데이터의 형식에 따른 도구 선정
 - TXT 파일 형식의 로그 데이터:
 - TXT 파일을 불러오기 위한 파이썬의 open 함수와 정형화된 데이터의 패턴을 기준으로 원하는 데이터만 추출하기 위해 파이썬 자료형, 정규 표현식 등의 문법을 활용
- 홈페이지 내 특정 데이터
 - 홈페이지 소스를 가져오기 위한 **파이썬의 Requests**와 같은 **HTTP 관련 모듈**과 **홈페이지 소스**에서 **특정 데이터를 추출**하기 위한 **BS4, Selenium**과 같은 모듈 활용

1. 데이터 분석 기초

□ 데이터 수집 이해

- 데이터 선정
- 데이터 수집 방법 선정
- **데이터 수집 도구 선정** 

(3) 데이터 수집도구 선정

- 수집한 데이터의 형식에 따른 도구 선정

- 항상 똑같은 형식, 개수의 데이터:

- **관계형 데이터베이스**를 통해 데이터를 수집하고 저장하면 추후에 수집한 데이터를 다시 불러와서 사용하기에 편리함

- 언제든지 새로운 항목이 추가될 수 있는 데이터:

- **JSON과 같은 파일 포맷의 데이터를 저장**하여 언제든지 새로운 항목을 추가, 수정, 삭제할 수 있음

```
1  {  
2    "이름": "홍길동",  
3    "나이": 25,  
4    "성별": "여",  
5    "주소": "서울특별시 양천구 목동",  
6    "특기": ["농구", "도술"],  
7    "가족관계": {"#": 2, "아버지": "홍판서", "어머니": "춘섬"},  
8    "회사": "경기 수원시 팔달구 우만동"  
9  }
```

1. 데이터 분석 기초

□ 데이터 수집 이해

- ▣ 데이터 선정
- ▣ 데이터 수집 방법 선정
- ▣ **데이터 수집 도구 선정**

(3) 데이터 수집도구 선정

- ▣ 무조건 코딩으로 데이터를 수집해야 하는가?

모든 데이터를 파이썬의 모듈을 활용해 수집할 필요는 없음

쉽게 수집 가능한 데이터라도 코딩이 필요한 경우

- ⊖ 하루에 한번씩, 한 시간에 한 번씩 클릭 해야할 때
- ⊖ 정확한 시간에 수집해야 할 때
- ⊗ 하루 이틀이 아닌 한달, 6개월, 1년처럼 장기적으로 수집이 필요할 때

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ **데이터 전처리의 개념**
- ▣ 데이터 전처리의 필요성
- ▣ 데이터 전처리의 종류

1) 데이터 전처리란?

- ▣ 데이터 정제, 데이터 전처리, 데이터 클렌징, 데이터 가공, 데이터 핸들링 등으로 다양하게 표현함
- ▣ 수집한 데이터를 분석에 적합하게 만들기 위해(정제, 클렌징, 가공 등) 별도 과정을 거친다는 것을 의미함
- ▣ 수집한 데이터를 **분석하려는 목적에 따라 해당 데이터를 분석에 가장 알맞도록 정제하는 과정**을 말함

데이터 전처리 역시 데이터 분석 과정의 일부임

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ **데이터 전처리의 필요성**
- ▣ 데이터 전처리의 종류

2) 데이터 전처리의 필요성

- ▣ 데이터 분석에 좋다는 최신기술을 사용하더라도 충분히 전처리 되지 않은 데이터를 입력한다면 좋은 결과를 얻을 수 없을 뿐만 아니라 왜곡된 분석 결과를 얻을 수도 있음
- ▣ 수집된 데이터가 정형화되고 신뢰할 만한 데이터만 있을 수는 없기때문에 데이터 전처리과정은 필수로 이루어져야 함
- ▣ 정형화된 데이터도 경우에 따라 데이터의 전처리가 필요할 수 있음

(예) 홍길동, 길동 홍 등 같은 사람이 다르게 인식될 수도 있음

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ **데이터 전처리의 필요성**
- ▣ 데이터 전처리의 종류

2) 데이터 전처리의 필요성

- ▣ 수집한 데이터 내에 이상 값들이 있다면 전처리 과정을 통해 데이터를 정제해야 함
- ▣ 이상 값이란?
 - 결측 값: 특정 데이터가 없는 경우
 - 입력 오류: 입력하면서 오류가 발생한 경우
 - 데이터 처리 오류: 데이터를 수집하는 과정에서 잘못 가져온 경우
 - 그 외 정상적이지 않은 데이터들: 측정, 실험 등의 오류 값 등
- ▣ 이상 값들을 처리하지 않으면?
 - 편향되거나 오차 분산이 커지는 등 데이터 분석 결과가 크게 달라질 수 있음

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ 데이터 전처리의 필요성
- ▣ **데이터 전처리의 종류**

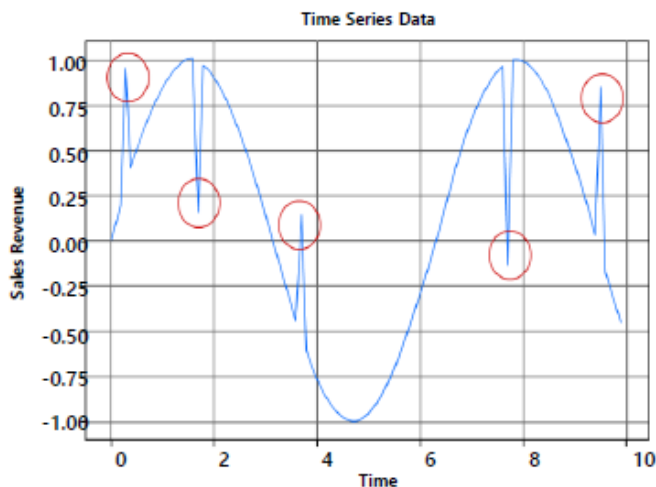
3) 데이터 전처리의 종류

- ▣ 전처리를 하는 방법은 수집된 데이터의 특징과 분석 목적에 따라 여러 가지로 나뉘며, 파이썬의 문법, 모듈 등을 활용할 수 있음
 - 이상 데이터 제거
 - 데이터 통합
 - 데이터 변환(정규화, 요약, 이산화 등)
 - 데이터 축소, 특징 추출

(1) 이상 데이터의 예시: 시계열 데이터에서의 이상 데이터

| 번호 | 이름 | 나이 |
|----|-----|------|
| 1 | 홍길동 | 24 |
| 2 | 이길동 | 63 |
| 3 | 김길동 | 9142 |
| 4 | 최길동 | -5.2 |
| 5 | 박길동 | - |

[이상 데이터 예시]



1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ 데이터 전처리의 필요성
- ▣ **데이터 전처리의 종류**

3) 데이터 전처리의 종류

(2) 데이터 통합의 예시

- 관련 있는 데이터들을 통합하여 일관성 있는 데이터 형태로 변환하는 과정

| 번호 | 이름 | 나이 |
|----|-----|----|
| 1 | 홍길동 | 22 |
| 2 | 이길동 | 19 |
| 3 | 김길동 | 34 |
| 4 | 최길동 | 56 |
| 5 | 박길동 | 42 |

| 이름 | 성별 | 혈액형 |
|-----|----|-----|
| 김길동 | 남 | A |
| 홍길동 | 여 | A |
| 최길동 | 여 | O |
| 박길동 | 여 | AB |
| 이길동 | 남 | B |

- 두 테이블의 데이터를 하나로 통합하여 활용할 수 있음

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ 데이터 전처리의 필요성
- ▣ **데이터 전처리의 종류**

3) 데이터 전처리의 종류

(3) 데이터 변환의 예시

- 여러 데이터간의 비교를 위해 정규화하는 과정 등 기존 데이터를 변환하여 활용

| 이름 | 타율 | 홈런 | | 이름 | 타율 | 홈런 |
|-----|-------|----|---|-----|-------|-------|
| 홍길동 | 0.245 | 34 | | 홍길동 | 0.245 | 0.828 |
| 이길동 | 0.321 | 20 | | 이길동 | 0.321 | 0.714 |
| 김길동 | 0.333 | 11 | → | 김길동 | 0.333 | 0.171 |
| 최길동 | 0.297 | 5 | | 최길동 | 0.297 | 0.0 |
| 박길동 | 0.258 | 40 | | 박길동 | 0.258 | 1.0 |

1. 데이터 분석 기초

□ 데이터 전처리 이해

- ▣ 데이터 전처리의 개념
- ▣ 데이터 전처리의 필요성
- ▣ **데이터 전처리의 종류**

3) 데이터 전처리의 종류

(4) 데이터 축소, 특징 추출의 예시

- 분석 목적에 필요 없는 데이터를 줄이거나 데이터에서 특징을 추출하고, 데이터의 차원을 축소하는 등

(예) 개와 고양이를 구분하기 위한 데이터

| 분류 | 꼬리 길이 | 눈동자 모양 | 다리 개수 | 유전자 형태 |
|-----|-------|--------|------------------------|--------|
| 개 | | | 분석 목적에 맞지 않는 데이터 | |
| 고양이 | | | | |
| 고양이 | | | | |
| 개 | | | | |
| 개 | | | | |

1. 데이터 분석 기초

□ 데이터 분석의 이해

- ▣ **데이터 분석 방법**
- ▣ 데이터 종류에 따른 분석 방법
- ▣ 파이썬에서 데이터 수집 및 분석방법

1) 데이터 분석 방법

- ▣ 데이터를 분석하는 방법: 데이터의 종류, 데이터 분석의 목적, 분야, 분석 도구 등에 따라 다양 함
- ▣ 수학, 통계학, 기계학습, 데이터 시각화 등 다양한 분석방법이 있고, 파이썬, 엑셀, R, Matlab 등 다양한 도구를 활용해 분석할 수 있음

데이터 분석의 목적은 수많은 데이터 중에서 의사결정 등에 도움이 되는 정보를 발견하고 이를 활용하여 가치를 창출하는데 있음

1. 데이터 분석 기초

□ 데이터 분석의 이해

▣ 데이터 분석 방법

- ▣ 데이터 종류에 따른 분석 방법
- ▣ 파이썬에서 데이터 수집 및 분석방법

1) 데이터 분석 방법

(1) 통계분석

- 통계를 기반으로 분석하는 방법
 - 회귀분석, 상관분석, 군집분석, 주성분분석 등

(2) 기계학습분석

- 컴퓨터에 스스로 학습하고 문제를 해결하는 능력을 줌
 - 회귀, 분류, 예측, 군집 등

(3) 시각화

- 분석 결과를 좀 더 자세하게 파악하기 위해 시각화하기도 하지만 여러 데이터 요소간의 관계나 단순 나열된 데이터에서 알지 못한 인사이트를 시각화를 통해 발견할 수 있음

1. 데이터 분석 기초

□ 데이터 분석의 이해

- ▣ 데이터 분석 방법
- ▣ **데이터 종류에 따른 분석 방법**
- ▣ 파이썬에서 데이터 수집 및 분석방법

2) 데이터 종류에 따른 분석방법

(1) 정량적 데이터 분석

- 데이터가 수치화된 형태일 때 분석하는 방법
- 객관적으로 데이터를 분석, 평가할 수 있음

주로 정형 데이터로 통계분석 등을 적용할 수 있음

(2) 정성적 데이터 분석

- 숫자가 아닌 질적으로 평가되는 데이터를 분석하는 방법
- 서술 형태로 표현되는 범주형 데이터를 분석함

주로 비정형 데이터로 텍스트 내 빈도 분석, SNS 데이터 분석 등을 할 수 있음

1. 데이터 분석 기초

□ 데이터 분석의 이해

- ▣ 데이터 분석 방법
- ▣ 데이터 종류에 따른 분석 방법
- ▣ **파이썬에서 데이터 수집 및 분석방법**

3) 파이썬에서 데이터 수집 및 분석 방법

(1) 파이썬의 기본문법 활용

- 파이썬의 기본 자료형, 기본 모듈을 활용하여 데이터를 수집, 전처리, 분석할 수 있음

(2) 파이썬의 외부 모듈 활용

- 잘 알려진 외부 라이브러리를 설치해 데이터를 수집 및 분석할 수 있음
 - 웹크롤링 : BS4, Selenium, Requests
 - 기계학습 : Scipy, Tensorflow, Keras, Pytorch
 - 통계/빅데이터분석 : Numpy, Pandas
 - 시각화 : Matplotlib, Seaborn
 - 데쉬보드 솔루션 : tableau, Power BI

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

□ Pandas 모듈 활용

1) Numpy 모듈의 개념

□ Numpy의 정의

- 대규모, 다차원배열을 쉽게 처리할 수 있도록 도와주는 파이썬의 외부 모듈
- 기본적으로 array라는 자료형을 사용함.
- 행렬의 개념과 비슷함.

□ 설치 방법

- Anaconda 설치 시 함께 설치됨
- pip install numpy로도 설치 가능함

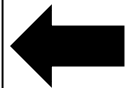
□ 호출 방법

- import numpy as np로 호출함

```
import numpy as np
```

```
a = [1,2,3,4,5]  
b = np.array(a)  
print(b)  
print(type(b))
```

```
[1 2 3 4 5]  
<class 'numpy.ndarray'>
```



2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용
- Pandas 모듈 활용

2) Numpy 모듈의 특징

- 파이썬의 리스트 자료형과 아주 유사함
- 실제로는 리스트와 비슷한 기능들을 사용할 수 있음
 - 예 : 인덱싱과 슬라이싱(a : 리스트, b: Numpy)

```
import numpy as np
```

```
a = [1,2,3,4,5]  
b = np.array(a)
```

```
print(type(a))  
print(type(b))
```

```
<class 'list'>  
<class 'numpy.ndarray'>
```

```
print(a[0])  
print(b[0])
```

```
1  
1
```

```
print(a[0:3])  
print(b[0:3])
```

```
[1, 2, 3]  
[1 2 3]
```


2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용
- Pandas 모듈 활용

2) Numpy 모듈의 특징

- 리스트 자료정보보다 다양한 방식의 인덱싱 지원
 - 리스트 : 요소를 기준으로 인덱싱
 - 행렬 : 위치를 기준으로 인덱싱

```
import numpy as np

a = [[1,1,1],[2,2,2],[3,3,3]]
b = np.array(a)

print(b[:,2])
print(b[1,:2])
```

```
[1 2 3]
[2 2]
```

```
print(b[1][1])
print(b[1,1])
```

```
2
2
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용 2) Numpy 모듈의 특징

□ Pandas 모듈 활용

□ 행렬 형태의 행렬 연산 지원

- 리스트: 연결(+), 반복(*) 연산만 지원, (-, /) 연산 불가능
- Numpy: 실제 행렬과 같이 행렬의 *, +, - 등의 연산 제공, 사칙연산 가능
- => 수학 계산에 용이함

```
import numpy as np
```

```
a = [1,2,3,4,5]  
b = np.array(a)
```

```
print(a + a)  
print(b + b)
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]  
[ 2  4  6  8 10]
```

```
print(a * 2)  
print(b * 2)
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]  
[ 2  4  6  8 10]
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

□ Pandas 모듈 활용

2) Numpy 모듈의 특징

▣ 다차원 행렬 및 행렬 연산 지원

- shape: 현재 행렬의 크기를 구할 수 있음
- 크기가 다른 두 행렬의 연산 지원

```
import numpy as np
```

```
a = [[1,1,1],[2,2,2],[3,3,3]]  
b = np.array(a)
```

```
print(a)  
print(b)
```

```
[[1, 1, 1], [2, 2, 2], [3, 3, 3]]  
[[1 1 1]  
 [2 2 2]  
 [3 3 3]]
```

```
print(b.shape)  
print(b * 3)
```

```
(3, 3)  
[[3 3 3]  
 [6 6 6]  
 [9 9 9]]
```

```
import numpy as np
```

```
a1 = [[1,1,1],[2,2,2],[3,3,3]]  
a2 = [1,2,3]  
b1 = np.array(a1)  
b2 = np.array(a2)
```

```
print(b1)  
print(b2)
```

```
[[1 1 1]  
 [2 2 2]  
 [3 3 3]]  
[1 2 3]
```

```
print(b1 + b2)
```

```
[[2 3 4]  
 [3 4 5]  
 [4 5 6]]
```

2. 파이썬 데이터 분석 라이브러리

- **Numpy 모듈 활용**

- **Pandas 모듈 활용**

2) Numpy 모듈의 특징

- 제공하는 함수 종류

- 수학과 관련된 함수: sqrt, log, max 등
- 행렬을 쉽게 정의할 수 있는 함수: zeros, ones, arange

```
# 수학과 관련 함수  
print(np.sqrt(4))  
print(np.log(10))  
print(np.max([1,2,3]))
```

```
2.0  
2.302585092994046  
3
```

2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용

- **Pandas 모듈 활용**

1) Pandas 모듈의 개념

- **Pandas 모듈**

- 행과 열로 이루어진 데이터를 쉽게 다룰 수 있도록 도와주는 파이썬의 데이터 분석 전용 외부 모듈
- 특히 대용량의 데이터를 처리하는데 편리함

- **설치방법**

- Anaconda 설치 시 함께 설치 됨
- pip install pandas로도 설치 가능 함

- **호출방법**

- 흔히 import pandas as pd로 호출하여 사용
- import pandas로만 호출해도 되지만 편의를 위해 pd로 줄여 사용

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

□ Pandas 모듈 활용

2) Pandas 모듈의 특징-

□ Pandas의 자료형 : **Series** 자료형

- 인덱스와 값을 가지고 있음
- 별도로 인덱스, 값을 출력할 수 있음, 정의할 때 인덱스를 따로 정해줄 수 있음

```
import pandas as pd
```

```
a = pd.Series([1,2,3,4],index = ['a','b','c','d'])  
print(a)  
print(a.index)
```

```
a    1
```

```
b    2
```

```
c    3
```

```
d    4
```

```
dtype: int64
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ Pandas의 자료형 : Series 자료형

- 파이썬의 딕셔너리 자료형
- Numpy의 Array 자료형도 Series 자료형으로 만들 수 있음
- 딕셔너리의 키가 Series의 인덱스가 됨

```
import pandas as pd
```

```
a = pd.Series({'a':1,'b':2,'c':3})  
print(a)  
print(a.index)
```

```
a    1  
b    2  
c    3  
dtype: int64  
Index(['a', 'b', 'c'], dtype='object')
```

```
import pandas as pd  
import numpy as np
```

```
a = np.array([1,2,3,4])  
b = pd.Series(a)  
print(b)
```

```
0    1  
1    2  
2    3  
3    4  
dtype: int32
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ Pandas의 자료형 : DataFrame 자료형

- 행과 열로 이루어진 자료형
- Series와 마찬가지로 파이썬의 딕셔너리 자료형 또는 Numpy의 array로도 정의 할 수 있음
- Series: 인덱스, 값으로만 구성, 1차원 배열 형태의 자료구조
- •DataFrame: 행과 열로 구성, 2차원 테이블 형태의 자료 구조

```
import pandas as pd
```

```
a = pd.Series({'a':[1,1,1],'b':[2,2,2],'c':[3,3,3]})  
b = pd.DataFrame({'a':[1,1,1],'b':[3,3,3],'c':[3,3,3]})
```

```
print(type(a))  
print(type(b))
```

```
print(a)
```

```
a    [1, 1, 1]  
b    [2, 2, 2]  
c    [3, 3, 3]  
dtype: object
```

```
print(b)
```

```
   a  b  c  
0  1  3  3  
1  1  3  3  
2  1  3  3
```


2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용 2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ Pandas의 자료형 : DataFrame 자료형

- 행: index, 열: columns
- 특정 칼럼의 데이터를 손쉽게 변경할 수 있음

```
import pandas as pd
```

```
a = pd.DataFrame({'a':(1,2), 'b':1, 'c':3})  
print(a)  
print(a.index)  
print(a.columns)
```

```
   a  b  c  
0  1  1  3  
1  2  1  3  
RangeIndex(start=0, stop=2, step=1)  
Index(['a', 'b', 'c'], dtype='object')
```

```
a['b'] = [3,4]  
print(a)
```

```
   a  b  c  
0  1  3  3  
1  2  4  3
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용 2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ DataFrame의 다양한 함수

- index와 columns를 변경할 수 있음
 - iloc: 행 인덱스로 값을 가져올 수 있음
 - loc: 행 이름으로 값을 가져올 수 있음

```
import pandas as pd

a = pd.DataFrame({'a':(1,2),'b':1,'c':3})

print(a)
a.index = ['x','y']
a.columns = ['i','j','k']
print()
print(a)
```

```
   a  b  c
0  1  1  3
1  2  1  3
```

```
   i  j  k
x  1  1  3
y  2  1  3
```

```
print(a['i'])
```

```
x    1
y    2
Name: i, dtype: int64
```

```
print(a.iloc[0])
```

```
i    1
j    1
k    3
Name: x, dtype: int64
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ DataFrame의 다양한 함수

- describe(): DataFrame에서 계산 가능한 값들에 대한 결과를 간략하게 보여 줌

```
import pandas as pd
```

```
a = pd.DataFrame({'a':(1,2,3),'b':[4,5,6],'c':[7,8,9]})  
print(a)
```

```
print(a.describe())
```

```
a b c  
0 1 4 7  
1 2 5 8  
2 3 6 9  
a b c  
count 3.0 3.0 3.0  
mean 2.0 5.0 8.0  
std 1.0 1.0 1.0  
min 1.0 4.0 7.0  
25% 1.5 4.5 7.5  
50% 2.0 5.0 8.0  
75% 2.5 5.5 8.5  
max 3.0 6.0 9.0
```

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용 2) Pandas 모듈의 특징

□ Pandas 모듈 활용

□ DataFrame의 다양한 함수

- sum(): 합계를 보여 줌
 - axis 옵션으로 행, 열 기준을 변경할 수 있음

□ 그 외 다양한 함수들

- min, max: 최소, 최대값
- argmin, argmax: 최소, 최대 값의 인덱스를 반환
- mean : 평균
- median: 중간 값
- •std, var: 표준편차, 분산
- •unique: 특정 행 또는 열에서 중복 값을 제외한 유니크 값을 반환

```
print(a.sum())
```

```
a    6  
b   15  
c   24  
dtype: int64
```

```
print(a.sum(axis=1))
```

```
0    12  
1    15  
2    18  
dtype: int64
```

2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용

1) 영화장르별빈도수분석

- 데이터불러오기

- Pandas 모듈 활용

- Pandas 모듈의 read_csv 함수를 활용해 DataFrame으로 불러올 수 있음

```
import pandas as pd
```

```
data = pd.read_csv('ml-latest-small/movies.csv')  
data
```

| movieId | | title |
|---------|---|-------------------------|
| 0 | 1 | Toy Story (1995) |
| 1 | 2 | Jumanji (1995) |
| 2 | 3 | Grumpier Old Men (1995) |

2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용
- Pandas 모듈 활용
- 모듈 활용 데이터 분석 실습

1) 영화장르별빈도수분석

- 장르분리하기
 - 반복문을 활용하여 genres 칼럼의 장르 값들을 모두 분리하여 리스트에 저장

```
genre = []  
for i in data['genres']:  
    genre.extend(i.split('|'))  
print(len(genre))  
print(genre)
```

22084

['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy', 'Comedy', 'Action', 'Crime', 'Thriller', 'Comedy', 'Romance', 'Comedy', 'Horror', 'Adventure', 'Animation', 'Children', 'Comedy', 'Action', 'Comedy', 'Crime', 'Drama', 'Thriller', 'Thriller', 'Drama', 'Sci-Fi', 'Drama', 'Romance', 'Drama', 'Crime', 'Drama', 'Drama', 'Mystery', 'Sci-Fi', 'Thriller',

2. 파이썬 데이터 분석 라이브러리

- Numpy 모듈 활용
- Pandas 모듈 활용
- 모듈 활용 데이터 분석 실습

1) 영화장르별빈도수분석

- 중복된 장르 제거하기
 - Pandas의 unique 함수를 활용하여 중복을 제거한 장르를 저장

```
unique_genre = pd.unique(genre)
print(len(unique_genre))
print(unique_genre)
```

20

['Adventure' 'Animation' 'Children' 'Comedy' 'Fantasy' 'R
'Action' 'Crime' 'Thriller' 'Horror' 'Mystery' 'Sci-Fi' 'War'
'Documentary' 'IMAX' 'Western' 'Film-Noir' '(no genres l

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

1) 영화장르별빈도수분석

□ Pandas 모듈 활용

□ 빈도수를 분석하기 위한 DataFrame 생성하기

- Numpy 모듈의 zeros 함수를 활용하여 장르별 빈도수를 분석하기 위해 비어 있는 DataFrame 생성

```
import numpy as np
zero_data = np.zeros(len(unique_genre))
result = pd.DataFrame(zero_data, index=unique_genre, columns=['count'])
print(result)
```

| | count |
|--------------------|-------|
| Adventure | 0.0 |
| Animation | 0.0 |
| Children | 0.0 |
| Comedy | 0.0 |
| Fantasy | 0.0 |
| Romance | 0.0 |
| Drama | 0.0 |
| Action | 0.0 |
| Crime | 0.0 |
| Thriller | 0.0 |
| Horror | 0.0 |
| Mystery | 0.0 |
| Sci-Fi | 0.0 |
| War | 0.0 |
| Musical | 0.0 |
| Documentary | 0.0 |
| IMAX | 0.0 |
| Western | 0.0 |
| Film-Noir | 0.0 |
| (no genres listed) | 0.0 |

2. 파이썬 데이터 분석 라이브러리

□ Numpy 모듈 활용

1) 영화장르별빈도수분석

□ Pandas 모듈 활용

▣ 장르 빈도수 분석하기

- 반복문을 활용하여 기존 전체 장르가 저장된 리스트를 하나씩 체크하며, 해당데이터의 값을+1 해줌

```
for i in genre:  
    result.loc[i] +=1  
print(result)
```

| | count |
|--------------------|--------|
| Adventure | 1263.0 |
| Animation | 611.0 |
| Children | 664.0 |
| Comedy | 3756.0 |
| Fantasy | 779.0 |
| Romance | 1596.0 |
| Drama | 4361.0 |
| Action | 1828.0 |
| Crime | 1199.0 |
| Thriller | 1894.0 |
| Horror | 978.0 |
| Mystery | 573.0 |
| Sci-Fi | 980.0 |
| War | 382.0 |
| Musical | 334.0 |
| Documentary | 440.0 |
| IMAX | 158.0 |
| Western | 167.0 |
| Film-Noir | 87.0 |
| (no genres listed) | 34.0 |