분석용 데이터

획득 전략

획득 데이터 가용성 분석

학습 목표

학습 목표

- 데이터 가용성 확인을 위한 데이터 전처리 과정을 설명할 수 있다.
- Numpy, Pandas 모듈을 활용하여 데이터 가용성을 분석할 수 있다.

학습 내용

- 데이터 가용성을 위한 전처리의 이해
- Numpy, Pandas 모듈을 활용하여 데이터 가용성 분석

데이터 전처리의 개념

1) 데이터 전처리의 의미

데이터 전처리란,

원자료(Raw Data)를 데이터 분석 목적과 방법에 맞는 형태로
처리하기 위하여 불필요한 정보를 분리 제거하고
가공하기 위한 예비적인 조작

- 빅데이터(Big Data) 분석, 데이터 마이닝(Data Mining)을 위해서는 각 알고리즘의 요구사항에 따라 잘 준비된 데이터가 필수적
- 따라서 수집된 데이터들을 목적에 맞게 효과적으로 가공 필요
- 이를 위해 데이터의 측정 오류를 줄이고 잡음(Noise), 왜곡, 편차를 최소화
- 정밀도, 정확도, 이상값(Outlier), 결측값(Missing Value), 모순, 불일치, 중복 등의 문제를 해결하기 위한 방법으로 사용

데이터 전처리의 개념

2) 데이터 전처리 주요 기법

데이터 정제(Cleansing)

결측값(Missing Value; 빠진 데이터)들을 채워 넣고,
이상치를 식별 또는 제거하고, 잡음 섞인 데이터를
평활화(Smoothing)하여 데이터의 불일치성을 교정하는 기술

데이터 변환(Transformation)

데이터 유형 변환 등 데이터 분석이 쉬운 형태로 변환하는 기술로
정규화(Normalization), 집합화(Aggregation),
요약(Summarization), 계층 생성 등의 방법을 활용

데이터 필터링(Filtering)

오류 발견, 보정, 삭제 및 중복성 확인 등의 과정을 통해
데이터의 품질을 향상하는 기술

데이터 통합(Integration)

데이터 분석이 용이하도록 유사 데이터 및 연계가
필요한 데이터들을 통합하는 기술

데이터 축소(Reduction)

분석 시간을 단축할 수 있도록 데이터 분석에
활용되지 않는 항목 등을 제거하는 기술

Numpy와 Pandas 모듈

1) Numpy 모듈의 개념

Numpy 모듈

- C언어로 구현된 파이썬 라이브러리로써, 고차원으로 구성되어 있는 행렬의 수치계산을 위해 개발
- Numerical Python의 줄임말로 다양한 수치 연산을 위한 여러 가지 기능들을 가지고 있고, 특히 통계 관련 기능들을 많이 포함
- 벡터 및 행렬 연산에 있어서 매우 편리한 기능을 제공
- 파이썬으로 데이터분석을 할 때 거의 필수적으로 사용되는 라이브러리인 Pandas와 Matplotlib의 기반

❖ 이미지 임베딩(Image Embedding) : 이미지를 기계가 이해할 수 있는 숫자 형식으로 변환
❖ 워드 임베딩(Word Embedding) : 텍스트를 기계가 이해할 수 있는 숫자 형식으로 변환

Numpy와 Pandas 모듈

2) Numpy 자료 구조

(1) Array

1차원 Array

➔ 데이터를 1개의 열 형태로 저장하고 관리 - 벡터

2차원 Array

➔ 데이터를 행과 열 형태로 저장하고 관리 - 행렬

3차원 Array

➔ 데이터를 행과 높이 형태로 저장하고 관리 - 배열

공통점

- Array에 저장되는 데이터의 유형은 모두 동일해야 함

Array : 컴퓨터 표의 형태로 저장된 값들의 모음

Numpy와 Pandas 모듈

3) Pandas 모듈의 개념

Pandas 모듈

- 데이터 조작 및 분석을 위한 Python 프로그래밍 언어용으로 작성된 소프트웨어 라이브러리
- 한 개인에 대해 여러 기간동안 관찰을 한다는 데이터 세트에 대한 계량 경제학 용어인 "패널 데이터"라는 용어에서 파생
- 또한 "Python 데이터 분석"이라는 문구 자체에서 유래
- 각 컬럼의 데이터 유형은 동일해도 되고 다른 유형의 데이터도 사용 가능

정보

데이터가 표 형태로 들어가 있을 경우 Pandas를 많이 사용

4) Series 유형과 Data Frame 유형

Series 유형	Data Frame 유형
컬럼이 1개인 표 형태	컬럼이 2개 이상인 표 형태

Numpy 사용하기

```python
import numpy as np

#numpy를 사용하기 위해 array 형태로 만들기
data1 = [1,2,3,4,5]
array1 = np.array(data1)
print(array1)

# array 의 크기와 형태 확인
array1.shape
```

```python
#numpy를 사용하기 위해 array 형태로 만들기
# 데이터 유형이 다를경우 - 정수와 실수
data2 = [1 , 2 , 3 , 4 , 5.5]
array2 = np.array(data2)
print(array2)

# array 의 크기와 형태 확인
array2.shape
```

```python
#numpy를 사용하기 위해 array 형태로 만들기
# 데이터 유형이 다를 경우 - 정수와 문자
data3 = [1 , 2 , 3 , 4 , '서진수']
array3 = np.array(data3)
print(array3)

# array 의 크기와 형태 확인
array3.shape
```

주의사항

데이터가 반드시 array 안에 들어있어야 Numpy 모듈 활용 가능

shape 명령어

데이터의 행과 열의 크기를 알려주는 명령어

Numpy 사용하기

```
array2 = np.array(data2)
print(array2)

# array 의 크기와 형태 확인
array2.shape
```

```
[1.  2.  3.  4.  5.5]
```

```
(5,)
```

```
#numpy를 사용하기 위해 array 형태로 만들기
# 데이터 유형이 다를 경우 - 정수와 문자
data3 = [1 , 2 , 3 , 4 , '서진수']
array3 = np.array(data3)
print(array3)

# array 의 크기와 형태 확인
array3.shape
```

```
['1' '2' '3' '4' '서진수']
```

```
(5,)
```

```
#array() 함수안에 리스트 바로 사용하기
array4 = np.array( [1,2,3,4,5] )
print(array4)
```

중요

array 안의 데이터는 반드시 **모두 동일한 형태**로 구성해야 함

Numpy 사용하기

- 데이터 타입 확인

```
# 데이터 타입 확인하기
print('array1:',array1.dtype)
print('array2:',array2.dtype)
print('array3:',array3.dtype)
print('array4:',array4.dtype)
```

```
# 2차원의 array 생성하기
array5 = np.array( [ [1,2,3],[3,4,5],[5,6,7] ])
print(array5)

array5.shape
```

```
# reshape( ) 함수를 사용하여 차원을 변경하기
array6 = np.arange(12).reshape(4,3)
print(array6)
```

```
# reshape( ) 함수를 사용하여 차원을 변경하기
array7 = array6.reshape(2,6)
print(array7)
```

```
# reshape( ) 함수를 사용하여 차원을 변경하기
array8 = array6.reshape(-1,1)
print(array8)
```

```
# 참고 : reshape( ) 함수는 원본 데이터를 복사하여 복사본의 형태를 바꾸고
# resize( ) 함수는 원본 데이터를 변경합니다
```

reshape

원하는 array의 **행렬의 차원을 변환**할 수 있는 명령어

Numpy 사용하기

- vstack( ) : 세로로 병합, hstack( ) : 가로로 병합

```
#vstack()
print('-vstack:','\n',np.vstack( (array9 , array10)) )
print()

#hstack()
print('-hstack:','\n',np.hstack( (array9 , array10)) )
```

```
# 2개 이상의 리스트를 array 로 합치기-column_stack()
import numpy as np
name = np.array(['홍길동','일지매','전우치'])
loc = np.array(['서울','부산','대전'])
print(name)
print(loc)
print()

all_data = np.column_stack( (name,loc))
print(all_data)
```

column_stack()

원하는 컬럼만 병합하여 새로운 array로 생성

```
# array 데이터 조회하기
# 2차원 array 조회하기
array2 = np.array([ [1,2,3],[4,5,6],[7,8,9] ])
print(array2)
print('\n')
print(array2[2,2])
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

주의사항

파이썬은 번호를 0부터 시작

Pandas 사용하기

```
data2 = pd.read_excel("c:\\py_temp\\프로야구선수성적_타자.xlsx")
data2.head()
```

	순위	선수명	팀명	타율	경기수	타수	득점	안타	2루타	3루타	홈런	타점
0	1	김선빈	KIA	0.370	137	476	84	176	34	1	5	64
1	2	박건우	두산	0.366	131	483	91	177	40	2	20	78
2	3	박민우	NC	0.363	106	388	84	141	25	4	3	47
3	4	나성범	NC	0.347	125	498	103	173	42	2	24	99
4	5	박용택	LG	0.344	138	509	83	175	23	2	14	90

```
data3 = pd.read_csv("c:\\py_temp\\프로야구선수성적_타자_ANSI.csv")
data3.head()
```

```
data3 = pd.read_csv("c:\\py_temp\\프로야구선수성적_타자_ANSI.csv" , encoding="cp949")
data3.head()
```

```
data3.head(3)
```

```
data3.tail(3)
```

```
data3.shape
```

주의사항

데이터를 불러올 때 인코딩이 맞지 않으면 에러 발생