『 3과목 :』
데이터 마트와 데이터 전처리

- Data Mart & Data Preprocessing
- Data Structures
- Data Gathering(Collect, Acquisition), Data Ingestion
- Data Invest & Exploratory Data Analysis, Data Visualization
- Data Cleansing (정제)
- **Data Integration (통합)**
- Data Reduction (축소)
- Data Transformation (변환)
- Feature Engineering & Data Encoding
- Cross Validation & Data Splitting
- Data Quality Assessment and Model Performance Evaluation
- 『3과목』 Self 점검

개요

학습목표

- 이 워크샵에서는 데이터 통합(integration)에 대해 알 수 있습니다.

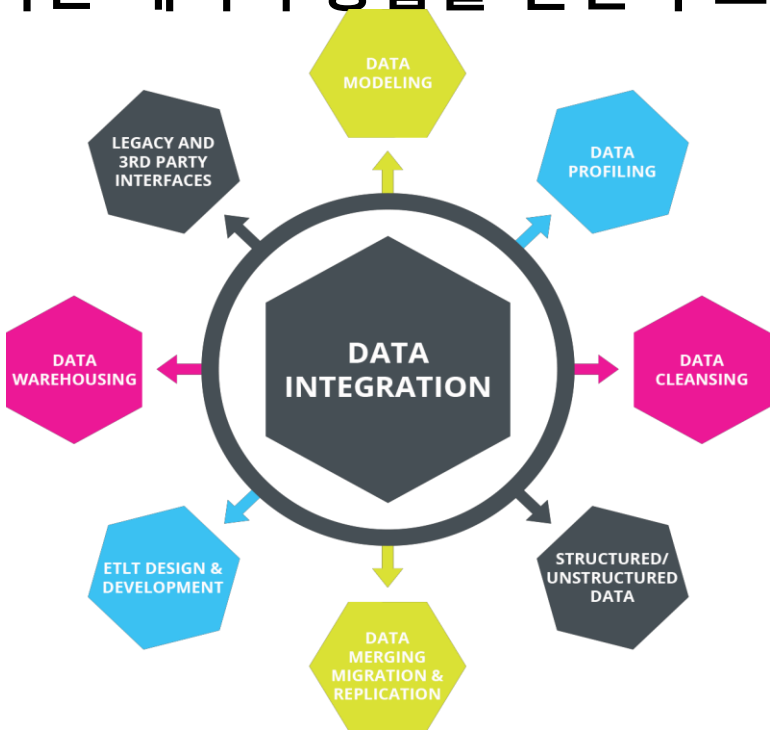
눈높이 체크

- 데이터 통합(integration)에 대해 들어보셨나요?

1. Data Integration?

데이터 통합Data Integration

- 서로 다른 출처의 여러 데이터를 결합
- 서로 다른 데이터 세트가 호환이 가능하도록 통합
- 같은 객체, 같은 단위나 좌표로 데이터를 통합
- 링크드 데이터의 핵심 목표 중 하나는 데이터 통합을 완전히 또는 거의 완전히 자동화하는 것
  - 링크드 데이터는 "웹 기술(URI, HTTP, RDF 등)"을 사용해 기계가 이해하고 탐색할 수 있도록 상호 연결된 데이터



출처 : http://www.matricis.com/en/integration-solutions/data-integration/

1. Data Integration?

데이터 통합Data Integration

- 데이터 통합data integration은 여러 데이터 저장소로부터 온 데이터의 합병
  - 데이터 웨어하우스data warehouse나 데이터마이닝data mining 같은 데이터 분석 작업은 다수의 원천 데이터로부터 하나의 통일된 데이터 저장소로 결합시키는 통합 작업 필요
  - 데이터 원천은 데이터베이스, 데이터 큐브data cube, 플랫 파일flat file 등 다양한 형태로 존재
  - 여러 데이터 원천들로부터 데이터를 통합할 때, 동일한 의미의 개체들이 서로 다르게 표현되어 있을 경우, 어떻게 일치 시킬 수 있을까? → 개체 식별 문제Entity Identification Problem

출처 : 개체 식별 문제Entity Identification Problem 이현호, Python과 SQL을 활용한 실전 데이터 전처리, 카오스북, 2018.

1. Data Integration?

데이터 값 충돌 탐지 및 해결

- 서로 다른 데이터 원천의 데이터들을 통합 할 때 동일한 개체에 대해서도 속성 값이 다를 수 있음 → 표현representation, 척도scaling, 부호화encoding 등의 차이
  - 거리를 나타내는 속성으로 어느 DB에서는 미터meter 단위로 다른 DB에서는 마일mile 단위로 저장
  - 학생 성적 데이터로 어느 DB에서는 과목별 점수가 저장되고, 다른 DB에서는 총점과 평균만 저장
- 동일한 개체의 동일한 값이 데이터 원천에 따라 다르게 표현되어 있는 경우, 데이터 통합 시에 기준을 정하여 데이터 값을 변환하여 통합시키는 것이 필요
- 통합 과정에서 DB의 속성을 일치시킬 때 데이터 구조에도 주의를 기울여야 함
- 원천 시스템의 기능적 종속성과 제약 사항들이 목표 통합 시스템의 것과 일치해야 함
  - 어느 시스템에서는 할인이 총 주문 금액에 적용되는 반면 다른 시스템에서는 주문을 구성하는 개별 항목에 적용
- 데이터의 의미적 이질성과 데이터 구조는 데이터 통합 시에 여러 문제를 발생시킴
- 여러 유형의 문제들을 신중하게 해결하여 통합 데이터의 중복과 불일치 문제를 최소화

1. Data Integration?

CSV

- CSV(쉼표로 구분된 값) 파일 형식은 데이터를 저장하고 공유하는 매우 간단한 방법.
  - CSV 파일은 데이터 테이블을 일반 텍스트로 보유.
  - 표(또는 스프레드시트)의 각 셀은 숫자나 문자열일 뿐. Excel 파일에 비해 CSV 파일의 주요 이점 중 하나는 일반 텍스트 파일을 저장, 전송 및 처리할 수 있는 프로그램이 많다는 것임. CSV 파일로 작업할 때 Excel의 일부 기능을 잃게 된다. Excel 스프레드시트의 모든 셀에는 정의된 "유형"(숫자, 텍스트, 통화, 날짜 등)이 있는 반면 CSV 파일의 셀은 원시 데이터일 뿐이다.

1. Data Integration?

CSV

- Python의 pandas library의 read_csv() 함수를 사용해서 외부 text 파일, csv 파일을 불러와서 DataFrame으로 저장. 불러오려는 text, csv 파일의 encoding 설정과 Python encoding 설정이 서로 맞지 않으면 UnicodeDecodeError 가 발생. 한글은 보통 'utf-8' 을 많이 사용하는데요, 만약 아래처럼 'utf-8' 코덱을 decode 할 수 없다고 에러 메시지가 나오는 경우가 있다.

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc1 in position 26: invalid start byte

- w_list.to_csv('Test.csv',encoding='euc-kr') #한글 저장 잘 됨
- w_list.to_csv('Test.csv',encoding='utf-8') #한글 깨짐
- w_list.to_csv('Test.csv',encoding='utf-8-sig') #utf-8로 한글 저장 잘 됨

2. Data Correction(데이터 수정)

Data Correction(데이터 수정)

- Data Correction은 잘못 기록되었거나 불완전한 데이터를 정확하고 일관된 값으로 정정하는 작업
  - 왜 필요한가?

문제 유형	예시
오타	Jonuary → January
잘못된 숫자	나이: 250세, 수량: -5개
형식 오류	날짜 형식: 13/45/2023
코드 값 불일치	성별: M, male, 남 → 통일 필요
단위 혼용	170cm, 1.7m 등
중복 값	동일 인물 여러 행 존재

2. Data Correction(데이터 수정)

Data Correction(데이터 수정)

- Data Correction은 잘못 기록되었거나 불완전한 데이터를 정확하고 일관된 값으로 정정하는 작업
  - 왜 필요한가?

문제 유형	예시
오타	Jonuary → January
잘못된 숫자	나이: 250세, 수량: -5개
형식 오류	날짜 형식: 13/45/2023
코드 값 불일치	성별: M, male, 남 → 통일 필요
단위 혼용	170cm, 1.7m 등
중복 값	동일 인물 여러 행 존재

2. Data Correction(데이터 수정)

Data Correction(데이터 수정)

- Data Correction의 주요 작업

작업	설명	예시
오타 수정	문자열 유사도, 사전 기반 자동 교정	Jonh → John
값 정규화	표준값으로 통일	M, Male, 남자 → 남
범위 확인 및 수정	값이 비정상적이면 대체	나이 300 → 평균값으로 대체
형식 변환	일관된 날짜/시간/숫자 형식	2024/05/16 → 2024-05-16
결측값 보완	NULL → 평균값/이전값/예측값 등으로 채우기	NaN → 중앙값

2. Data Correction(데이터 수정)

Data Correction vs Data Cleaning

- Data Correction은 Cleaning의 하위 작업 중 하나이지만,

구분	설명
Data Cleaning	오류 제거, 중복 제거, 결측치 처리 전체 과정
Data Correction	잘못된 값을 "수정"하는 데 초점

2. Data Correction(데이터 수정)

Column 수정

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

2. Data Correction(데이터 수정)

Column 수정

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학교'].replace({'강남고':'강동고', '강북구':'강서고'}) df
결과값1	지원번호 1번 강동고 2번 강동고 3번 강동고 4번 강동고 5번 강북고 6번 강북고 7번 강북고 8번 강북고
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

2. Data Correction(데이터 수정)

Column 수정

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학교'].replace({'강남고':'강동고'}, inplace=True) df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강동고 197 90 85 100 95 85 Python 2번 박문수 강동고 184 40 35 50 55 25 Java 3번 이순신 강동고 168 80 75 70 80 75 Javascript 4번 임꺽정 강동고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

2. Data Correction(데이터 수정)

소문자로 가공

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['SW특기'].str.lower() df['SW특기'] = df['SW특기'].str.lower() df
결과값1	지원번호 1번 python 2번 java 3번 javascript 4번 NaN 5번 NaN 6번 c 7번 python 8번 c#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강동고 197 90 85 100 95 85 python 2번 박문수 강동고 184 40 35 50 55 25 java 3번 이순신 강동고 168 80 75 70 80 75 javascript 4번 임꺽정 강동고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 c 7번 서화담 강북고 188 55 65 45 40 35 python 8번 정난정 강북고 190 100 85 90 95 95 c#
비고	

2. Data Correction(데이터 수정)

대문자로 가공

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['SW특기'] = df['SW특기'].str.upper() df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강동고 197 90 85 100 95 85 PYTHON 2번 박문수 강동고 184 40 35 50 55 25 JAVA 3번 이순신 강동고 168 80 75 70 80 75 JAVASCRIPT 4번 임꺽정 강동고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

2. Data Correction(데이터 수정)

문자열 추가

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학교'] = df['학교'] + '등학교' # 학교 데이터 + 등학교 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 6번 황진희 강북고등학교 202 80 100 95 85 80 C 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 8번 정난정 강북고등학교 190 100 85 90 95 95 C#
비고	

2. Data Correction(데이터 수정)

Column 추가

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['총합'] = df['국어'] + df['영어'] + df['수학'] + df['과학'] + df['사회'] df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465
비고	

2. Data Correction(데이터 수정)

Column 추가

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['결과'] = 'Fail' # 결과 Column 을 추가하고 전체 데이터는 Fail 로 초기화 df df.loc[df['총합'] > 400, '결과'] = 'Pass' # 총합이 400보다 큰 데이터에 대해서 결과를 Pass 로 업데이트 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Fail 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Fail 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Fail
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass
비고	

2. Data Correction(데이터 수정)

Column 삭제

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.drop(columns=['총합']) # 총합 Column 을 삭제 df.drop(columns=['국어', '영어', '수학']) # 국어, 영어, 수학 Column 을 삭제
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# Pass
결과값2	이름 학교 키 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 85 80 C 440 Pass 7번 서화담 강북고등학교 188 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 95 95 C# 465 Pass
비고	

2. Data Correction(데이터 수정)

Row 삭제

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.drop(index='4번') # 4번 학생 데이터 row 를 삭제 filt = df['수학'] < 80 # 수학 점수 80 점 미만 학생 필터링 df[filt]
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail
비고	

2. Data Correction(데이터 수정)

Row 삭제

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df[filt].index df.drop(index=df[filt].index) df
결과값1	Index(['2번', '3번', '4번', '5번', '7번'], dtype='object', name='지원번호')
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass
결과값3	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass
비고	

2. Data Correction(데이터 수정)

Row 추가

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.loc['9번'] = ['정약용', '해남고등학교', 184, 90, 90, 90, 90, 90, 'Kotlin', 450, 'Pass'] # 새로운 Row 추가 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 NaN 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass 9번 정약용 해남고등학교 184 90 90 90 90 90 Kotlin 450 Pass
비고	

2. Data Correction(데이터 수정)

Cell 수정

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.loc['4번', 'SW특기'] = 'Python' # 4번 학생의 SW특기 데이터를 Python 으로 변경 df df.loc['5번', ['학교', 'SW특기']] = ['광주고등학교', 'C'] # 5번 학생의 학교는 광주고등학교로, SW특기는 C로 변경 df
결과값1	1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 Python 325 Fail 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass 9번 정약용 해남고등학교 184 90 90 90 90 90 Kotlin 450 Pass
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 결과 지원번호 1번 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 Pass 2번 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 Fail 3번 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 Fail 4번 임꺽정 강동고등학교 187 40 60 70 75 80 Python 325 Fail 5번 강백호 광주고등학교 188 15 20 10 35 10 C 90 Fail 6번 황진희 강북고등학교 202 80 100 95 85 80 C 440 Pass 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 Fail 8번 정난정 강북고등학교 190 100 85 90 95 95 C# 465 Pass 9번 정약용 해남고등학교 184 90 90 90 90 90 Kotlin 450 Pass
비고	

2. Data Correction(데이터 수정)

Colum 순서 변경

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	cols = list(df.columns) cols df = df[[cols[-1]] + cols[0:-1]] # 맨 뒤에 있는 결과 Column 을 앞으로 가져오고, 나머지 Column 들과 합쳐서 순서 변경 df
결과값1	['이름', '학교', '키', '국어', '영어', '수학', '과학', '사회', 'SW특기', '총합', '결과']
결과값2	결과 이름 학교 키 국어 영어 수학 과학 사회 SW특기 총합 지원번호 1번 Pass 홍길동 강동고등학교 197 90 85 100 95 85 PYTHON 455 2번 Fail 박문수 강동고등학교 184 40 35 50 55 25 JAVA 205 3번 Fail 이순신 강동고등학교 168 80 75 70 80 75 JAVASCRIPT 380 4번 Fail 임꺽정 강동고등학교 187 40 60 70 75 80 Python 325 5번 Fail 강백호 광주고등학교 188 15 20 10 35 10 C 90 6번 Pass 황진희 강북고등학교 202 80 100 95 85 80 C 440 7번 Fail 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 240 8번 Pass 정난정 강북고등학교 190 100 85 90 95 95 C# 465 9번 Pass 정약용 해남고등학교 184 90 90 90 90 90 Kotlin 450
비고	

2. Data Correction(데이터 수정)

Colum 순서 변경

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df = df[['결과', '이름', '학교']] df
결과값1	결과 이름 학교 지원번호 1번 Pass 홍길동 강동고등학교 2번 Fail 박문수 강동고등학교 3번 Fail 이순신 강동고등학교 4번 Fail 임꺽정 강동고등학교 5번 Fail 강백호 광주고등학교 6번 Pass 황진희 강북고등학교 7번 Fail 서화담 강북고등학교 8번 Pass 정난정 강북고등학교 9번 Pass 정약용 해남고등학교
비고	

2. Data Correction(데이터 수정)

Column 이름 변경

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.columns df.columns = ['Result', 'Name', 'School'] df
결과값1	Index(['결과', '이름', '학교'], dtype='object')
결과값2	Result Name School 지원번호 1번 Pass 홍길동 강동고등학교 2번 Fail 박문수 강동고등학교 3번 Fail 이순신 강동고등학교 4번 Fail 임꺽정 강동고등학교 5번 Fail 강백호 광주고등학교 6번 Pass 황진희 강북고등학교 7번 Fail 서화담 강북고등학교 8번 Pass 정난정 강북고등학교 9번 Pass 정약용 해남고등학교
비고	

2. Data Correction(데이터 수정)

열(column) 수정

```
import pandas as pd

df4 = pd.DataFrame([
        ['A01', 2, 1, 60, 139, 'country', 0, 3, 3],
        ['A02', 3, 2, 80, 148, 'country', 0, 6, 4],
        ['A03', 3, 4, 50, 149, 'country', 0, 6, 8],
        ['A04', 5, 5, 40, 151, 'country', 0, 9, 11],
        ['A05', 7, 5, 35, 154, 'city', 0, 13, 11],
        ['A06', 2, 5, 45, 149, 'country', 0, 8, 6],
        ['A07',8, 9, 40, 155, 'city', 1, 14, 12],
        ['A08', 9, 10, 70, 155, 'city', 3, 12, 14],
        ['A09', 6, 12, 55, 154, 'city', 0, 11, 13],
        ['A10', 9, 2, 40, 156, 'city', 1, 14, 12],
        ['A11', 6, 10, 60, 153, 'city', 0, 11, 13],
        ['A12', 2, 4, 75, 151, 'country', 0, 5, 7]
    ], columns=['ID', 'hour', 'attendance', 'weight', 'iq', 'region', 'library', 'english_score', 'math_score'])
```

2. Data Correction(데이터 수정)

열(column) 수정

```
df4['english_score'] = 0
#df4['english_score'] = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

print(df4)
```

```
    ID  hour  attendance  weight   iq   region  library  english_score  \
0   A01     2           1      60  139  country        0              0
1   A02     3           2      80  148  country        0              0
2   A03     3           4      50  149  country        0              0
3   A04     5           5      40  151  country        0              0
4   A05     7           5      35  154     city        0              0
5   A06     2           5      45  149  country        0              0
6   A07     8           9      40  155     city        1              0
7   A08     9          10      70  155     city        3              0
8   A09     6          12      55  154     city        0              0
9   A10     9           2      40  156     city        1              0
10  A11     6          10      60  153     city        0              0
11  A12     2           4      75  151  country        0              0

    math_score
0            3
1            4
2            8
3           11
4           11
5            6
6           12
7           14
8           13
9           12
10          13
11           7
```

2. Data Correction(데이터 수정)

열(column) 수정

```
df4['english_score'] = df4['english_score'] / 2

print(df4)
```

```
    ID  hour  attendance  weight   iq   region  library  english_score  \
0   A01     2           1      60  139  country        0            0.0
1   A02     3           2      80  148  country        0            0.0
2   A03     3           4      50  149  country        0            0.0
3   A04     5           5      40  151  country        0            0.0
4   A05     7           5      35  154     city        0            0.0
5   A06     2           5      45  149  country        0            0.0
6   A07     8           9      40  155     city        1            0.0
7   A08     9          10      70  155     city        3            0.0
8   A09     6          12      55  154     city        0            0.0
9   A10     9           2      40  156     city        1            0.0
10  A11     6          10      60  153     city        0            0.0
11  A12     2           4      75  151  country        0            0.0

    math_score
0            3
1            4
2            8
3           11
4           11
5            6
6           12
7           14
8           13
9           12
10          13
11           7
```

2. Data Correction(데이터 수정)

행(row) 수정

```
df4.loc[0] = ['A13', 2, 5, 76, 153, 'country', 0, 6, 8]

print(df4)
```

```
    ID  hour  attendance  weight   iq   region  library  english_score  \
0   A13     2           5      76  153  country        0            6.0
1   A02     3           2      80  148  country        0            0.0
2   A03     3           4      50  149  country        0            0.0
3   A04     5           5      40  151  country        0            0.0
4   A05     7           5      35  154     city        0            0.0
5   A06     2           5      45  149  country        0            0.0
6   A07     8           9      40  155     city        1            0.0
7   A08     9          10      70  155     city        3            0.0
8   A09     6          12      55  154     city        0            0.0
9   A10     9           2      40  156     city        1            0.0
10  A11     6          10      60  153     city        0            0.0
11  A12     2           4      75  151  country        0            0.0

    math_score
0            8
1            4
2            8
3           11
4           11
5            6
6           12
7           14
8           13
9           12
10          13
11           7
```

3. 데이터 삭제

열(column) 삭제

```
df4 = df4.drop(columns='ID')
#df4 = df4.drop(columns=['ID'])
#df4.drop(columns=['ID'], inplace=True)
print(df4)
```

---

	hour	attendance	weight	iq	region	library	english_score	math_score
0	2	5	76	153	country	0	6.0	8
1	3	2	80	148	country	0	0.0	4
2	3	4	50	149	country	0	0.0	8
3	5	5	40	151	country	0	0.0	11
4	7	5	35	154	city	0	0.0	11
5	2	5	45	149	country	0	0.0	6
6	8	9	40	155	city	1	0.0	12
7	9	10	70	155	city	3	0.0	14
8	6	12	55	154	city	0	0.0	13
9	9	2	40	156	city	1	0.0	12
10	6	10	60	153	city	0	0.0	13
11	2	4	75	151	country	0	0.0	7

4. collate(결합, 정리)

행(row) 삭제

```
import pandas as pd

df4 = pd.DataFrame([
        ['A01', 2, 1, 60, 139, 'country', 0, 3, 3],
        ['A02', 3, 2, 80, 148, 'country', 0, 6, 4],
        ['A03', 3, 4, 50, 149, 'country', 0, 6, 8],
        ['A04', 5, 5, 40, 151, 'country', 0, 9, 11],
        ['A05', 7, 5, 35, 154, 'city', 0, 13, 11],
        ['A06', 2, 5, 45, 149, 'country', 0, 8, 6],
        ['A07',8, 9, 40, 155, 'city', 1, 14, 12],
        ['A08', 9, 10, 70, 155, 'city', 3, 12, 14],
        ['A09', 6, 12, 55, 154, 'city', 0, 11, 13],
        ['A10', 9, 2, 40, 156, 'city', 1, 14, 12],
        ['A11', 6, 10, 60, 153, 'city', 0, 11, 13],
        ['A12', 2, 4, 75, 151, 'country', 0, 5, 7]
    ], columns=['ID', 'hour', 'attendance', 'weight', 'iq', 'region', 'library', 'english_score', 'math_score'])

df4 = df4.drop(columns='ID')
#df4 = df4.drop(columns=['ID'])
#df4.drop(columns=['ID'], inplace=True)
print(df4)
```

```
------
 hour  attendance  weight   iq   region  library  english_score  math_score
0     2           1     60  139  country        0              3           3
1     3           2     80  148  country        0              6           4
2     3           4     50  149  country        0              6           8
3     5           5     40  151  country        0              9          11
4     7           5     35  154     city        0             13          11
5     2           5     45  149  country        0              8           6
6     8           9     40  155     city        1             14          12
7     9          10     70  155     city        3             12          14
8     6          12     55  154     city        0             11          13
9     9           2     40  156     city        1             14          12
10    6          10     60  153     city        0             11          13
11    2           4     75  151  country        0              5           7
```

4. collate(결합, 정리)

오름차순 정렬

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.sort_values('키') # 키 기준으로 오름차순 정렬
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 3번 이순신 강남고 168 80 75 70 80 75 Javascript 2번 박문수 강남고 184 40 35 50 55 25 Java 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C# 1번 홍길동 강남고 197 90 85 100 95 85 Python 6번 황진희 강북고 202 80 100 95 85 80 C
비고	

4. collate(결합, 정리)

내림차순 정렬

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.sort_values('키', ascending=False) # 키 기준으로 내림차순 정렬
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 6번 황진희 강북고 202 80 100 95 85 80 C 1번 홍길동 강남고 197 90 85 100 95 85 Python 8번 정난정 강북고 190 100 85 90 95 95 C# 5번 강백호 강북고 188 15 20 10 35 10 NaN 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript
비고	

4. collate(결합, 정리)

수학, 영어 점수 기준으로 오름차순

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.sort_values(['수학', '영어']) # 수학, 영어 점수 기준으로 오름차순
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 5번 강백호 강북고 188 15 20 10 35 10 NaN 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 2번 박문수 강남고 184 40 35 50 55 25 Java 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 3번 이순신 강남고 168 80 75 70 80 75 Javascript 8번 정난정 강북고 190 100 85 90 95 95 C# 6번 황진희 강북고 202 80 100 95 85 80 C 1번 홍길동 강남고 197 90 85 100 95 85 Python
비고	'수학'과 '영어' 점수가 높은 순서대로 데이터가 정렬되며, DataFrame의 모든 행(row)은 '수학' 점수가 동일한 경우 '영어' 점수를 기준으로 정렬됩니다.

4. collate(결합, 정리)

수학, 영어 점수 기준으로 내림차순

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df df.sort_values(['수학', '영어'], ascending=False) # 수학, 영어 점수 기준으로 내림차순
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 6번 황진희 강북고 202 80 100 95 85 80 C 8번 정난정 강북고 190 100 85 90 95 95 C# 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 2번 박문수 강남고 184 40 35 50 55 25 Java 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 5번 강백호 강북고 188 15 20 10 35 10 NaN
비고	

4. collate(결합, 정리)

키 컬럼 하나에 대한 정렬

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['키'].sort_values() df['키'].sort_values(ascending=False)
결과값1	지원번호 3번 168 2번 184 4번 187 5번 188 7번 188 8번 190 1번 197 6번 202
결과값2	지원번호 6번 202 1번 197 8번 190 5번 188 7번 188 4번 187 2번 184 3번 168
비고	

4. collate(결합, 정리)

인덱스에 대한 정렬

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df.sort_index() df.sort_index(ascending=False)
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
결과값2	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 8번 정난정 강북고 190 100 85 90 95 95 C# 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 6번 황진희 강북고 202 80 100 95 85 80 C 5번 강백호 강북고 188 15 20 10 35 10 NaN 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 3번 이순신 강남고 168 80 75 70 80 75 Javascript 2번 박문수 강남고 184 40 35 50 55 25 Java 1번 홍길동 강남고 197 90 85 100 95 85 Python
비고	

5. 함수 적용

원 데이터

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd df = pd.read_excel('datasets/score.xlsx', index_col='지원번호') df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고 197 90 85 100 95 85 Python 2번 박문수 강남고 184 40 35 50 55 25 Java 3번 이순신 강남고 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고 187 40 60 70 75 80 NaN 5번 강백호 강북고 188 15 20 10 35 10 NaN 6번 황진희 강북고 202 80 100 95 85 80 C 7번 서화담 강북고 188 55 65 45 40 35 PYTHON 8번 정난정 강북고 190 100 85 90 95 95 C#
비고	

5. 함수 적용

원 데이터

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	df['학교'] = df['학교'] + '등학교' df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고등학교 197 90 85 100 95 85 Python 2번 박문수 강남고등학교 184 40 35 50 55 25 Java 3번 이순신 강남고등학교 168 80 75 70 80 75 Javascript 4번 임꺽정 강남고등학교 187 40 60 70 75 80 NaN 5번 강백호 강북고등학교 188 15 20 10 35 10 NaN 6번 황진희 강북고등학교 202 80 100 95 85 80 C 7번 서화담 강북고등학교 188 55 65 45 40 35 PYTHON 8번 정난정 강북고등학교 190 100 85 90 95 95 C#
비고	

5. 함수 적용

데이터에 함수 적용 (apply)

- apply()란?
  - Pandas의 apply()는 DataFrame의 각 행 또는 열에 사용자 정의 함수를 적용할 수 있도록 해주는 메서드.
  - 축(axis) 방향을 지정해서 행 또는 열 단위로 적용 가능
  - 함수(function), 람다(lambda) 등 다양한 연산을 적용할 수 있음
- 기본 문법
  - df.apply(함수, axis=0 or 1)

인자	의미
함수	각 행 또는 열에 적용할 함수
axis=0	각 열(column)을 기준으로 적용 (기본값)
axis=1	각 행(row)을 기준으로 적용

5. 함수 적용

데이터에 함수 적용 (apply)

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	import pandas as pd import numpy as np Df_12 = pd.DataFrame({ '국어': [80, 90, 70], '영어': [85, 95, 65] }) # 열별 평균 print(Df_12.apply(np.mean)) # axis=0이 기본값
결과값1	국어 80.0 영어 81.6...
비고	예제 1: 열(column)에 함수 적용 (axis=0)

5. 함수 적용

데이터에 함수 적용 (apply)

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 학생별 총점 계산 Df_12['총점'] = Df_12.apply(lambda row: row['국어'] + row['영어'], axis=1)
결과값1	국어 영어 총점 0 80 85 165 1 90 95 185 2 70 65 135
비고	예제 2: 행(row)에 함수 적용 (axis=1)

5. 함수 적용

데이터에 함수 적용 (apply)

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	def 등급_매기기(점수): return 'A' if 점수 >= 90 else 'B' df['국어등급'] = df['국어'].apply(등급_매기기)
결과값1	…
비고	예제 3: 함수 따로 정의해서 적용

5. 함수 적용

데이터에 함수 적용 (apply)

파일	소스코드
실습환경	준비 py3_10_basic
소스코드	# 키 뒤에 cm 을 붙이는 역할 def add_cm(height): return str(height) + 'cm' df['키'] = df['키'].apply(add_cm) # 키 데이터에 대해서 add_cm 함수를 호출한 결과 데이터를 반영 df
결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고등학교 197cm 90 85 100 95 85 Python 2번 박문수 강남고등학교 184cm 40 35 50 55 25 Java 3번 이순신 강남고등학교 168cm 80 75 70 80 75 Javascript 4번 임꺽정 강남고등학교 187cm 40 60 70 75 80 NaN 5번 강백호 강북고등학교 188cm 15 20 10 35 10 NaN 6번 황진희 강북고등학교 202cm 80 100 95 85 80 C 7번 서화담 강북고등학교 188cm 55 65 45 40 35 PYTHON 8번 정난정 강북고등학교 190cm 100 85 90 95 95 C#
비고	

5. 함수 적용

데이터에 함수 적용 (apply)

파일	소스코드
실습환경	준비 py3_10_basic

소스코드	def capitalize(lang):
	  if pd.notnull(lang): # NaN 이 아닌지
	  return lang.capitalize() # 첫 글자는 대문자로, 나머지는 소문자로
	  return lang
	
	df['SW특기'] = df['SW특기'].apply(capitalize)
	df

결과값1	이름 학교 키 국어 영어 수학 과학 사회 SW특기 지원번호 1번 홍길동 강남고등학교 197cm 90 85 100 95 85 Python 2번 박문수 강남고등학교 184cm 40 35 50 55 25 Java 3번 이순신 강남고등학교 168cm 80 75 70 80 75 Javascript 4번 임꺽정 강남고등학교 187cm 40 60 70 75 80 NaN 5번 강백호 강북고등학교 188cm 15 20 10 35 10 NaN 6번 황진희 강북고등학교 202cm 80 100 95 85 80 C 7번 서화담 강북고등학교 188cm 55 65 45 40 35 Python 8번 정난정 강북고등학교 190cm 100 85 90 95 95 C#
비고	

『 3과목 :』
데이터 마트와 데이터 전처리

- Data Mart & Data Preprocessing
- Data Structures
- Data Gathering(Collect, Acquisition), Data Ingestion
- Data Invest & Exploratory Data Analysis, Data Visualization
- Data Cleansing (정제)
- Data Integration (통합)
- **Data Reduction (축소)**
- Data Transformation (변환)
- Feature Engineering & Data Encoding
- Cross Validation & Data Splitting
- Data Quality Assessment and Model Performance Evaluation
- 『3과목』 Self 점검

개요

학습목표

- 이 워크샵에서는 데이터 축소(Data Reduction에 대해 알 수 있습니다.

눈높이 체크

- 데이터 축소(Data Reduction) 에 대해 들어보셨나요?

1. Data Reduction?

데이터 축소Data Reduction

- 일반적으로 데이터는 매우 크기 때문에 대용량 데이터에 대한 복잡한 데이터 분석은 실 행하기 어렵거나 불가능한 경우가 많음
- 데이터 축소는 원래 용량 기준보다 작은 양의 데이터 표현결과를 얻게 되더라도 원 데이 터의 완결성을 유지하기 위해 사용
- 데이터를 축소하면 데이터 분석 시 좀 더 효과적이고 원래 데이터와 거의 동일한 분석 결 과를 얻어낼 수 있는 장점

출처 : https://www.cohesity.com/blog/cohesity-data-reduction-lock-stock-barrel/

1. Data Reduction?

데이터 축소Data Reduction

- 원천 데이터의 축소판(압축판)을 얻기 위한 데이터 부호화 또는 데이터 변환의 적용
  - 원천 데이터가 정보의 손실 없이 압축된다면 무손실 loseless
  - 원천 데이터의 근사치만으로 축소된다면 손실 lossy
  - 일반적으로 많이 사용되며 효과적인 손실 차원 축소 방법
    - 웨이블릿 변환 wavelet transform
    - 주성분 분석 principal components analysis

출처 : 이현호, Python과 SQL을 활용한 실전 데이터 전처리, 카오스북, 2018.

2. Data Reduction 종류

웨이블릿 변환wavelet transform

- 이산 웨이블릿 변환discrete wavelet transform, DWT: 데이터 벡터 X를 다른 수치적 벡터numerically vector X'으로 변환 (X와 X' 의 길이는 동일)
  - 각 튜플을 n 차원 데이터 벡터로 간주하면, 벡터 $X = x1, x2, \cdots, xn$를 각 튜플로 고려, 웨이블릿 변환 데이터가 원천 데이터와 같은 길이(속성 수)를 가지지만 데이터 축소로 볼 수 있는 것은 변환 데이터가 압축되어 보이기 때문
  - 웨이블릿 계수 중 가장 유력한 일부만을 저장함으로써 데이터 근사치를 유지
  - 예를 들어, 사용자가 정한 어떤 임계값보다 큰 모든 웨이블릿 계수들만 값을 유지하고 나 머지 계수들을 0으로 간주하면, 결과적인 데이터 표현은 매우 희소해지며, 데이터 희소 성data sparsity은 데이터 연산의 복잡도를 크게 감소시킬 수 있음
  - 데이터의 주요 특징들은 보존하면서도 잡음을 제거하는 역할을 하기도 하므로 데이터 정제를 위해서도 효과적임

2. Data Reduction 종류

주성분 분석Principal Components Analysis, PCA

- 주성분 분석Principal Components Analysis, PCA은 n개의 속성을 가진 튜플(n 차원의 데이터 벡터)에 대하여 데이터를 표현하는데 최 적으로 사용될 수 있는 n 차원 직교벡터orthogonal vector들에 대한 k를 찾음 ($k \leq n$)→ 감소된 차원의 공간을 갖는 데이터 공간 생성 (차원 축소)

2. Data Reduction 종류

수량 축소 방법 -표본 추출sampling

- 큰 데이터 집합을 많은 수의 임의 데이터 샘플(부분집합)로 표현 가능
- 대용량 데이터 집합 D가 N개의 튜플을 포함하고 있다고 가정

- 비복원 단순 무작위표본Simple Random Sample WithOut Replacement, SRSWOR: D로부터 N개의 튜플 중 에서 임의의 s개를 취하는 방법으로서 모든 튜플들의 표본으로 추출될 확률은 같음
- 복원 단순 무작위표본Simple Random Sample With Replacement With Replacement, SRSWR: 각 튜플이 D로부터 추출될 때마다 기록된 후 다시 제자리로 복원replace 된다는 것을 제외하면 SRSWOR와 유사, 각 튜플은 추출된 다음에 다시 추출될 수 있도록 D에 되돌려짐

- 집략표본Cluster Sample: D에 있는 튜플들이 M개의 상호 배반적 군집 cluster으로 묶여 있는 가운데 s개의 군집을 단순 무작위로 추출 ($s < M$)
- 층화표본Stratified Sample: D가 층strata이라 불리는 상호 배반적 부분들로 분할되어 있다면, 각 층에서 하나씩 단순 무작위로 추출 (예, 고객의 나이 그룹 각각에 대하여 하나의 층이 생 성되어 있는 고객 데이터로부터 층화표본을 얻음)

2. Data Reduction 종류

수량 축소 방법-히스토그램histogram

- 구간화를 사용하여 데이터 분포의 근사치를 구하는 데이터 축소의 전형적 형태
- 속성 A의 데이터를 버킷bucket 혹은 빈bin이라 불리는 분리 집합disjoint subset으로 나눔
- 각 버킷이 단일한 속성 값/빈도의 쌍으로 표현되기도 하고, 주어진 속성에 대한 연속 범 위continuous range를 나타내기도 함
- 히스토그램은 희소 데이터나 밀집 데이터 모두에 효과적, 비대칭적 데이터와 균일한 데이터 모두 매우 효과적
- 단일 속성에 대한 히스토그램은 다중 속성에 대한 것으로 확장 가능
- 다차원 히스토그램에서는 속성 간의 의존성 포착 가능
- 일반적으로 5개 까지의속성을 가진 데이터의 근사치를 구하는 데 효과적이라고 알려짐

2. Data Reduction 종류

수량 축소 방법-군집화clustering

- 군집cluster이라는 그룹으로 나눔
  - 한 군집 내 객체들과는 유사하면서도 다른 군
  - 집 내 객체들과는 유사하지 않도록 군집화
  - 유사성은 공간 내에서 객체들이 어떻게 가까 운지의 관점에 따라 거리 함수에 기반하여 정 의
  - 클러스터의 품질은 지름diameter의 표현으로 나타내고, 지름은 클러스터의 두 객체 간 최대 거리로 표현
  - 클러스터 간 중심 거리centroid distance는 클러스터 중심 간 거리로서 클러스터 품질로 대체 측정
  - 클러스터의 지름은 짧을수록(클러스터 내 객 체 간의 유사성이 강할수록), 클러스터 간 중 심 거리는 길수록(클러스터 간 유사성은 약할 수록) 군집화의 품질이 높다고 볼 수 있음

2. Data Reduction 종류

주요 기법

- 데이터 축소 기법은 고차원 데이터의 복잡성을 줄여 연산 효율성을 높이고, 노이즈를 줄이며, 모델 성능을 향상시키는 데 사용. 적절한 데이터 축소 기법을 선택하여 데이터의 유용한 정보를 보존하면서 불필요한 정보를 제거하여 더 효율적인 분석을 수행할 수 있.

유형	기법	설명
속성 축소	Feature Selection	불필요한 컬럼 제거 (예: 상관관계 낮은 변수 제거)
	Feature Extraction	기존 속성들을 조합해 새로운 축소된 속성 생성 (예: PCA)
차원 축소	PCA (주성분 분석)	고차원 데이터를 저차원 공간으로 투영
	LDA, t-SNE, UMAP 등	클래스 분리 / 시각화 목적
표본 축소	샘플링 (Sampling)	전체 데이터에서 일부만 추출
중복 제거	Deduplication	동일하거나 유사한 레코드 제거
불필요 데이터 제거	Null, Noise 제거	의미 없는 값, 잡음 제거

2. Data Reduction 종류

특성 선택 (Feature Selection)

- 필터링 방법 (Filter Methods): 통계적 기준 또는 상관 관계를 사용하여 특성을 선택. 예를 들어, 분산 기준으로 특성을 선택하거나 목표 변수와의 상관 관계가 높은 특성을 선택.
- 래퍼 방법 (Wrapper Methods): 모델을 사용하여 특성의 부분 집합을 평가하는 방식으로 선택. 예를 들어, 순차적으로 특성을 추가하거나 제거하여 가장 좋은 결과를 얻는 특성을 선택.
- 임베디드 방법 (Embedded Methods): 모델 훈련 과정에서 특성 선택을 수행. 일부 머신러닝 알고리즘은 특성 선택을 위한 내부 메커니즘을 제공.

3. 특성 선택 (Feature Selection)

필터링 방법

- 필터링 방법은 통계적 기준 또는 상관 관계를 이용하여 특성을 선택하는 방법을 의미합니다. 이 방법을 구현하는 데에는 주로 통계적 지표를 사용하여 특성을 평가하고 선택합니다.
- 아래는 붓꽃 데이터셋을 사용하여 분산 기준으로 특성을 선택하는 예시 코드입니다.

```
from sklearn.feature_selection import VarianceThreshold
import pandas as pd

# Sample dataset
data = {
    'Feature1': [1, 2, 3, 4, 5],
    'Feature2': [5, 4, 3, 2, 1],
    'Feature3': [1, 1, 1, 0, 0]
}
df = pd.DataFrame(data)

df
```

3. 특성 선택 (Feature Selection)

필터링 방법

```
# Applying VarianceThreshold to select features
selector = VarianceThreshold(threshold=0.2)  # Set the variance threshold
selected_features = selector.fit_transform(df)

# Get the indices of the selected features
indices = selector.get_support(indices=True)
print("Selected feature indices:", indices)

# New dataset with selected features
df_selected = df.iloc[:, indices]
print("New dataset with selected features:")
print(df_selected)
```

>>

```
Selected feature indices: [0 1 2]
New dataset with selected features:
   Feature1  Feature2  Feature3
0         1         5         1
1         2         4         1
2         3         3         1
3         4         2         0
4         5         1         0
```

3. 특성 선택 (Feature Selection)

필터링 방법

- VarianceThreshold를 사용하여 특정 분산 값(threshold) 이상을 가지는 특성(feature)을 선택하는 예시입니다.
  - VarianceThreshold는 주어진 분산(threshold) 값 이상을 가지는 특성을 선택합니다.
  - selector.fit_transform(df)를 사용하여 주어진 데이터프레임(df)의 특성을 변환하고 선택된 특성만 반환합니다.
  - selector.get_support(indices=True)는 선택된 특성의 인덱스를 반환합니다.
  - df.iloc[:, indices]를 사용하여 선택된 특성만 포함하는 새로운 데이터프레임(df_selected)을 생성합니다.
  - 해당 코드를 실행하면, 특성의 분산이 주어진 임계값(여기서는 0.2)보다 크거나 같은 특성들의 인덱스를 출력하고, 선택된 특성들로 이루어진 새로운 데이터프레임을 보여줍니다.

3. 특성 선택 (Feature Selection)

래퍼 방법(Wrapper Methods)

- 래퍼 방법(Wrapper Methods)은 기계 학습 알고리즘을 사용하여 특성의 하위 집합을 평가하고 모델 성능에 기반하여 최선의 특성 하위 집합을 선택하는 특성 선택 방법입니다. 그 중 하나인 Recursive Feature Elimination (RFE)을 사용하는 예시 코드입니다.

```
from sklearn.datasets import make_classification
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

# 가상 데이터 생성
X, y = make_classification(n_samples=100, n_features=10, random_state=42)

# 모델 초기화
model = LogisticRegression()

# RFE 초기화
rfe = RFE(model, n_features_to_select=5)  # 선택할 특성의 개수
```

3. 특성 선택 (Feature Selection)

래퍼 방법(Wrapper Methods)

```
# RFE 적합
fit = rfe.fit(X, y)

# 선택된 특성 랭킹
print("특성 랭킹:", fit.ranking_)

# 선택된 특성 마스크
print("선택된 특성 마스크:", fit.support_)

# 선택된 특성으로 데이터 변환
X_selected = rfe.transform(X)
print("선택된 특성만 포함된 새로운 데이터셋:")
print(X_selected)
```

>>

```
특성 랭킹: [1 1 1 1 3 4 1 5 2 6]
선택된 특성 마스크: [ True  True  True  True False False  True False False False]
선택된 특성만 포함된 새로운 데이터셋:
[[-1.14052601  1.35970566  0.86199147  0.84609208  1.75479418]
 [-0.07873421 -1.32933233  0.6273745  -1.19300559  0.49799829]
 [ 0.80742726  0.73019848 -1.28568005  0.88948365  0.04808495]
 [ 0.58846525 -0.3751207  -0.57500215 -0.14951801  0.24368721]
 [ 1.63631177 -1.64060704 -1.36045573 -0.94116325  0.13074058]
 [-0.47936995 -1.08324727  1.02255619 -1.09939128  1.36687427]
 [ 2.48904785 -2.52343407 -2.05832072 -1.45612516 -1.02438764]
 [-0.23938468  1.56931739 -0.3313761   1.306542   -1.65485667]
 [ 0.67287309 -0.72427983 -0.53963044 -0.43066755  1.09877685]
 [-1.22823431  2.38869353  0.55942643  1.7240021  -0.57689187]
 [-0.82741596  0.88629356  0.66530077  0.52576893  0.48937456]
 [ 1.6253749   0.06194498 -2.02632079  0.55253544 -0.14436041]
```

3. 특성 선택 (Feature Selection)

래퍼 방법(Wrapper Methods)

- make_classification은 가상의 데이터를 생성.
- LogisticRegression은 기본 모델로 사용.
- RFE는 모델과 선택할 특성의 개수로 초기화.
- fit 메서드는 RFE를 데이터에 맞춤.
- fit.ranking_은 특성의 중요도에 따른 순위를 제공.
- fit.support_는 선택된 특성의 boolean 마스크를 제공.
- rfe.transform(X)는 선택된 특성만 포함된 새로운 데이터를 생성.

● 실제로는 더 정교한 모델을 사용하고 교차 검증을 수행하여 최적의 특성 수를 결정하는 것이 좋다. 데이터셋과 요구 사항에 따라 조정 및 튜닝이 필요.

3. 특성 선택 (Feature Selection)

임베디드 방법(Embedded Methods)

임베디드 방법(Embedded Methods)은 특성 선택을 위한 모델 자체의 내장 방법으로, 특성의 중요도나 가중치를 통해 특성을 선택. 일반적으로 특성 선택과 모델 학습이 동시에 이루어짐. 예를 들어, Lasso나 Ridge와 같은 정규화를 사용하는 선형 회귀 모델이 있습니다. 아래는 Lasso를 사용한 특성 선택의 예시 코드.

3. 특성 선택 (Feature Selection)

임베디드 방법(Embedded Methods)

```
from sklearn.datasets import make_regression
from sklearn.linear_model import Lasso

# 가상 데이터 생성
X, y = make_regression(n_samples=100, n_features=10, random_state=42)

# Lasso 모델 초기화 (L1 정규화 사용)
lasso = Lasso(alpha=0.1)  # alpha는 정규화 강도를 조절하는 매개변수

# Lasso 모델에 데이터를 적합하여 특성 선택 수행
lasso.fit(X, y)

# 선택된 특성 확인
selected_features = lasso.coef_ != 0
print("선택된 특성 마스크:", selected_features)

# 선택된 특성으로 새로운 데이터셋 생성
X_selected = X[:, selected_features]
print("새로운 데이터셋 크기:", X_selected.shape)
```

>>

```
선택된 특성 마스크: [ True  True  True  True  True  True  True  True  True  True]
새로운 데이터셋 크기: (100, 10)
```

3. 특성 선택 (Feature Selection)

임베디드 방법(Embedded Methods)

- make_regression은 가상 데이터를 생성.
- Lasso는 L1 정규화를 사용하는 선형 회귀 모델로 초기화.
- lasso.fit()을 사용하여 데이터에 모델을 적합시키고 특성을 선택.
- lasso.coef_는 선택된 특성의 계수를 제공하며, 0이 아닌 특성은 선택된 것으로 간주.
- lasso.transform()은 선택된 특성만 포함된 새로운 데이터셋을 생성.
- 실제로는 다양한 모델과 매개변수를 사용하여 실험하고 최상의 결과를 찾아야 함. 데이터셋과 문제에 따라 적절한 임베디드 방법을 선택.

4. 차원 축소

차원 축소(Dimensionality Reduction)?

- 차원축소 기법은 대량의 빅데이터를 분석하는 데 있어, 분석대상이 되는 여러 변수들의 주요 정보는 최대한 유지하면서 데이터 세트의 변수의 개수를 줄이는 일련의 탐색적 분석 기법을 말한다.

  - 차원축소 기법은 하나의 완결된 분석 기법으로 사용되기보다는 다른 분석과정을 위한 전 단계나 분석 수행 후 결과를 개선하기 위한 방법 혹은 효과적인 시각화 등의 목적으로 사용되는 경우가 많다.
  - 데이터 차원(변수 세트)이 많은 상황에서 데이터에 대한 효과적인 시각화를 통한 통찰을 얻고자 할 때, 저차원(일반적으로 2차원)으로 투영시키거나 다른 지도학습 및 자율학습 등을 수행하는 과정에서 매우 많은 변수가 있는 상황, 즉 차원의 저주(The curse of Dimensionality)를 해결하기 위해 서로 상관성이 높거나 유사한 변수들을 공통 변수로 결합하여 분석을 수행하고자 할 때 주로 사용되는 경우가 많다.

4. 차원 축소

데이터 축소(Data Reduction)?

- 차원의 저주(The curse of dimensionality)란 고차원 공간에서 데이터 분석 및 모델링을 수행할 때 발생하는 여러 문제들을 가리키는 용어. 고차원 데이터에서는 몇 가지 문제가 발생할 수 있다.

1. 데이터 희소성(Sparsity of data): 고차원 공간에서는 데이터가 채워져 있는 공간보다 희소한 경향이 있습니다. 데이터 포인트 간의 거리가 멀어져서 데이터가 더 분산되고 샘플 간의 관계를 파악하기 어려워집니다.
2. 계산 복잡성(Computational complexity): 고차원 데이터는 고차원의 많은 특성(feature)을 가지므로, 모델 학습 및 예측에 필요한 계산 복잡성이 증가합니다. 이는 메모리 사용량과 연산 속도에 영향을 미치며, 학습 시간이 증가할 수 있습니다.
3. 과적합(Overfitting): 고차원 데이터에서는 모델이 학습 데이터에 너무 맞추어져 새로운 데이터에 대한 일반화 성능이 떨어질 수 있습니다. 모델이 불필요한 특성이나 잡음까지 학습하여 일반화 능력이 감소할 수 있습니다.
4. 차원의 저하(Dimensionality reduction): 고차원 데이터에서는 시각화와 같은 목적으로 데이터를 이해하기 어렵기 때문에, 차원 축소 기법이 필요할 수 있습니다. 이러한 기법은 주성분 분석(PCA), t-SNE 등을 포함합니다.

- 차원의 저주를 극복하기 위해서는 데이터 특성을 선택하거나 추출하여 차원을 줄이거나, 더 많은 데이터를 수집하여 희소성을 감소시키는 방법 등이 있다. 또한, 적절한 특성 선택, 차원 축소 기법, 모델의 복잡성 관리 등을 통해 차원의 저주에 대처할 수 있다.

4. 차원 축소

데이터 축소(Data Reduction)?

- 특성 선택(Feature Selection) 은 넓은 의미에서 **차원 축소 (Dimensionality Reduction) 의 한 예

구분	방식	설명	대표 기법
1. 특성 선택 (Feature Selection)	있는 변수 중 "고르기"	기존 변수 중 중요한 것만 남김	상관관계 기반, 분산 기준, SelectKBest, Lasso 등
2. 특성 추출 (Feature Extraction)	새로운 변수 " 만들기"	기존 변수들을 조합하여 새로운 특성을 생성	PCA, LDA, t-SNE, UMAP, AutoEncoder 등

4. 차원 축소

주성분 분석(PCA)

- 고차원 데이터의 특성들을 선형 변환하여 서로 독립적인 새로운 변수들인 주성분으로 변환. 데이터의 분산을 최대한 보존하는 새로운 축으로 변환하여 차원을 축소.
  - 주성분은 데이터의 가장 중요한 변동성을 설명하는 방향을 나타냅니다. 주성분 분석(PCA)은 파이썬에서 scikit-learn 라이브러리를 활용하여 간단하게 수행할 수 있습니다.

```
from sklearn.decomposition import PCA
import pandas as pd
import matplotlib.pyplot as plt

# 예시 데이터 생성
data = {
    'Feature1': [1, 2, 3, 4, 5],
    'Feature2': [5, 4, 3, 2, 1],
    'Feature3': [1, 1, 1, 0, 0]
}

# 데이터프레임 생성
df = pd.DataFrame(data)

# PCA 모델 생성 및 학습
pca = PCA(n_components=2)  # 주성분 개수 설정
pca.fit(df)
```

4. 차원 축소

주성분 분석(PCA)

- 특성 추출과 차원 축소- 주성분 분석은 n개의 속성을 가진튜플(n 차원의 데이터 벡터)에 대하여 데이터를 표현하는데 최적으로 사용될 수 있는 n차원 직교벡터orthogonal vector들에 대한 k를 찾음($k \leq n$)→ 감소된 차원의 공간을 갖는 데이터 공간 생성(차원 축소)
- 주성분 분석 모형

$$Z_1 = \gamma_{11}X_1 + \gamma_{12}X_2 + \ldots\ldots\ldots + \gamma_{1p}X_p = \gamma_1^T X$$
$$Z_2 = \gamma_{21}X_1 + \gamma_{22}X_2 + \ldots\ldots\ldots + \gamma_{2p}X_p = \gamma_2^T X$$
$$\ldots\ldots \qquad \ldots\ldots\ldots \qquad \ldots\ldots$$
$$Z_q = \gamma_{q1}X_1 + \gamma_{q2}X_2 + \ldots\ldots\ldots + \gamma_{qp}X_p = \gamma_q^T X$$

$$\text{maximize}_{\gamma 11,\ldots\ldots,\gamma 1p}\left\{\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{p}\gamma_{1j}x_{ij}\right)^2\right\}$$

$$sbj \ \sum_{j=1}^{p}\gamma^{2_{1j}} = 1$$

새로운 주성분으로 해당 주성분 분산을 최대화하는 선형조합임

4. 차원 축소

주성분 분석(PCA)

- 주성분 분석 예

ID	국어	수학
1	95	95
2	90	95
3	80	75
4	60	70
5	40	35
6	80	80
7	95	90
8	30	25
9	15	10
10	60	70
평균	64.50	64.50
분산	808	925

$$\frac{95 - 64.50}{28.425}$$

$$\frac{30.50}{28.425} \approx 1.075$$

$$Z = \frac{X - \mu}{\sigma}$$

ID	국어	수학
1	1.1	1.0
2	0.9	1.0
3	0.5	0.3
4	-0.2	0.2
5	-0.9	-1.0
6	0.5	0.5
7	1.1	0.8
8	-1.2	-1.3
9	-1.7	-1.8
10	-0.2	0.2
평균	0.0	0.0
분산	1	1

4. 차원 축소

주성분 분석(PCA)

- 주성분 분석 예

ID	국어	수학
1	1.1	1.0
2	0.9	1.0
3	0.5	0.3
4	-0.2	0.2
5	-0.9	-1.0
6	0.5	0.5
7	1.1	0.8
8	-1.2	-1.3
9	-1.7	-1.8
10	-0.2	0.2
평균	0.0	0.0
분산	1	1
공분산	0.98	0.98

분산-공분산 매트릭스
20개의 숫자를 단 4개로 만들어줌

$$\sum = \begin{matrix} \\ 국어 \\ 수학 \end{matrix} \begin{bmatrix} 국어 & 수학 \\ \mathbf{1} & \mathbf{0.98} \\ \mathbf{0.98} & \mathbf{1} \end{bmatrix}$$

고유값

$$\sum = det \begin{bmatrix} \mathbf{1-\lambda} & \mathbf{0.98} \\ \mathbf{0.98} & \mathbf{1-\lambda} \end{bmatrix}$$

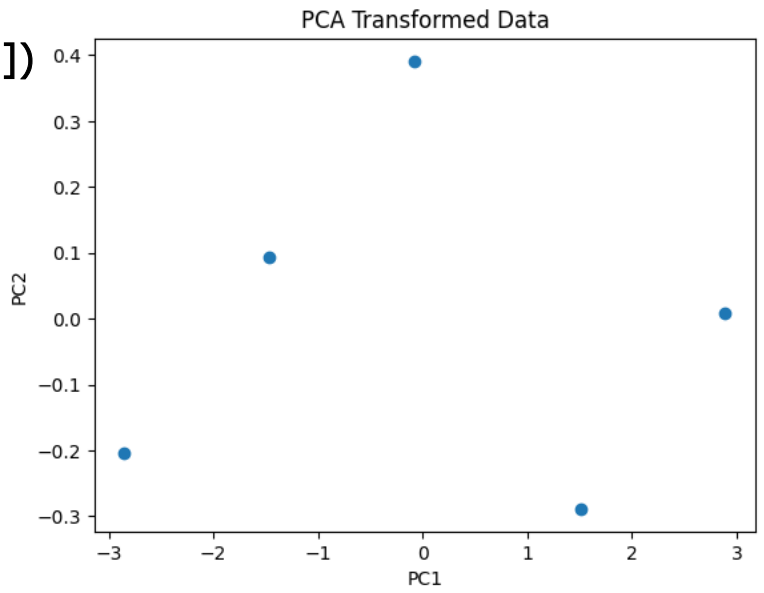
$$(1-\lambda)^2 - 0.98^2 = 0.02$$

4. 차원 축소

주성분 분석(PCA)

```
# 주성분 변환
transformed_data = pca.transform(df)

# 주성분으로 변환된 데이터프레임 생성
df_transformed = pd.DataFrame(transformed_data, columns=['PC1', 'PC2'])

# 주성분 분석 결과 확인
explained_variance_ratio = pca.explained_variance_ratio_
print("주성분 설명 분산 비율:", explained_variance_ratio)

# 시각화 (2차원으로 축소한 경우)
plt.scatter(df_transformed['PC1'], df_transformed['PC2'])
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PCA Transformed Data')
plt.show()
```

```
주성분 설명 분산 비율: [0.98646691 0.01353309]
```



4. 차원 축소

주성분 분석(PCA)

- 위 코드에서는 PCA()를 사용하여 PCA 모델을 생성하고, fit() 함수를 사용하여 데이터에 대해 주성분 분석을 수행합니다. n_components 매개변수를 통해 원하는 주성분의 개수를 지정할 수 있습니다. 그 후 transform() 함수를 사용하여 주어진 데이터를 주성분 공간으로 변환합니다.

- 또한, explained_variance_ratio_ 속성은 각 주성분이 설명하는 분산의 비율을 나타내며, 이를 통해 주성분의 중요성을 확인할 수 있습니다. 마지막으로, 시각화를 통해 주성분 공간으로 변환된 데이터를 확인할 수 있습니다.

4. 차원 축소

붓꽃 데이터 주성분 분석(PCA) 예

```
from sklearn.decomposition import PCA
import pandas as pd

# 4차원 데이터 (예: 붓꽃 데이터)
from sklearn.datasets import load_iris
X = load_iris().data

# 2차원으로 축소
pca = PCA(n_components=2)
X_reduced = pca.fit_transform(X)

print(X_reduced[:5])  # [n_samples, 2]
```

```
[[-2.68412563  0.31939725]
 [-2.71414169 -0.17700123]
 [-2.88899057 -0.14494943]
 [-2.74534286 -0.31829898]
 [-2.72871654  0.32675451]]
```

- Iris 데이터셋(150개 샘플, 4차원)을 PCA로 2차원으로 축소.
- 주성분 공간에서의 첫 5개 샘플 좌표를 출력.
- 이렇게 하면 고차원의 데이터를 시각화하거나 분석할 때 유용.
- 예를 들어, 첫 번째 샘플은 주성분 공간에서 (-2.684, 0.319) 위치에 매핑.

4. 차원 축소

t-SNE

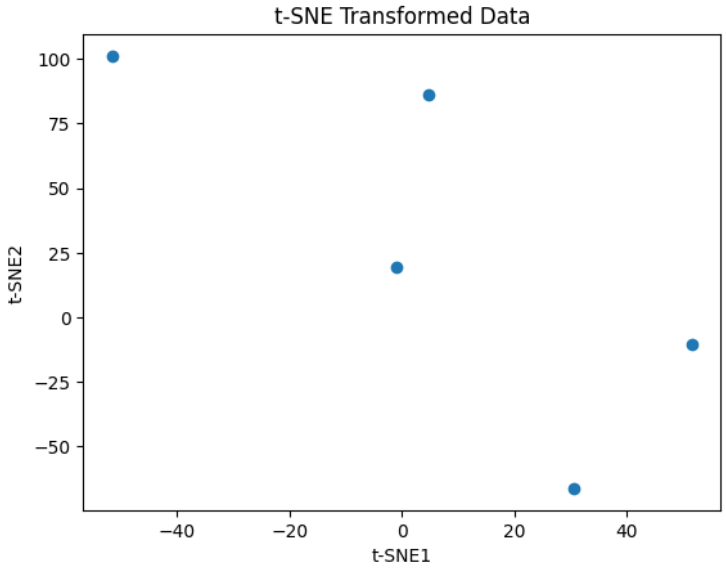
- 고차원 데이터를 저차원으로 매핑하여 시각화하는 데 자주 사용되는 비선형 차원 축소 기법. 데이터 포인트들 간의 유사성을 보존하면서 차원을 축소.
  - t-SNE(t-distributed Stochastic Neighbor Embedding)는 고차원 데이터를 저차원으로 축소하여 시각화하는 데 사용되는 비선형 차원 축소 기법입니다. scikit-learn 라이브러리를 통해 t-SNE를 쉽게 사용할 수 있습니다.

```
from sklearn.manifold import TSNE
import pandas as pd
import matplotlib.pyplot as plt

# 예시 데이터 생성
data = {
    'Feature1': [1, 2, 3, 4, 5],
    'Feature2': [5, 4, 3, 2, 1],
    'Feature3': [1, 1, 1, 0, 0]
}
```

4. 차원 축소

t-SNE

```
# 데이터프레임 생성
df = pd.DataFrame(data)

# t-SNE 모델 생성 및 학습
tsne = TSNE(n_components=2, perplexity=3, random_state=42)  # perplexity 값을 3으로 변경
tsne_result = tsne.fit_transform(df)

# t-SNE 결과 확인
df_tsne = pd.DataFrame(tsne_result, columns=['t-SNE1', 't-SNE2'])

# 시각화
plt.scatter(df_tsne['t-SNE1'], df_tsne['t-SNE2'])
plt.xlabel('t-SNE1')
plt.ylabel('t-SNE2')
plt.title('t-SNE Transformed Data')
plt.show()
```



4. 차원 축소

t-SNE

- 위 코드에서는 TSNE()를 사용하여 t-SNE 모델을 생성하고, fit_transform() 함수를 사용하여 데이터에 대해 t-SNE를 적용합니다. n_components 매개변수를 통해 축소할 차원의 수를 지정할 수 있습니다.

- 그 후, t-SNE 결과를 시각화하기 위해 각 데이터 포인트의 새로운 좌표인 't-SNE1', 't-SNE2'를 플로팅하여 저차원 공간으로 변환된 데이터를 확인할 수 있습니다. t-SNE는 데이터의 군집 또는 구조를 시각적으로 파악할 때 유용한 도구입니다.

『 3과목 :』
데이터 마트와 데이터 전처리

- Data Mart & Data Preprocessing
- Data Structures
- Data Gathering(Collect, Acquisition), Data Ingestion
- Data Invest & Exploratory Data Analysis, Data Visualization
- Data Cleansing (정제)
- Data Integration (통합)
- Data Reduction (축소)
- **Data Transformation (변환)**
- Feature Engineering & Data Encoding
- Cross Validation & Data Splitting
- Data Quality Assessment and Model Performance Evaluation
- 『3과목』 Self 점검

개요

학습목표

- 이 워크샵에서는 데이터 변환(Data Transformation)에 대해 알 수 있습니다.

눈높이 체크

- 데이터 변환(Data Transformation) 에 대해 들어보셨나요?

1. 데이터 변환(Data Transformation)

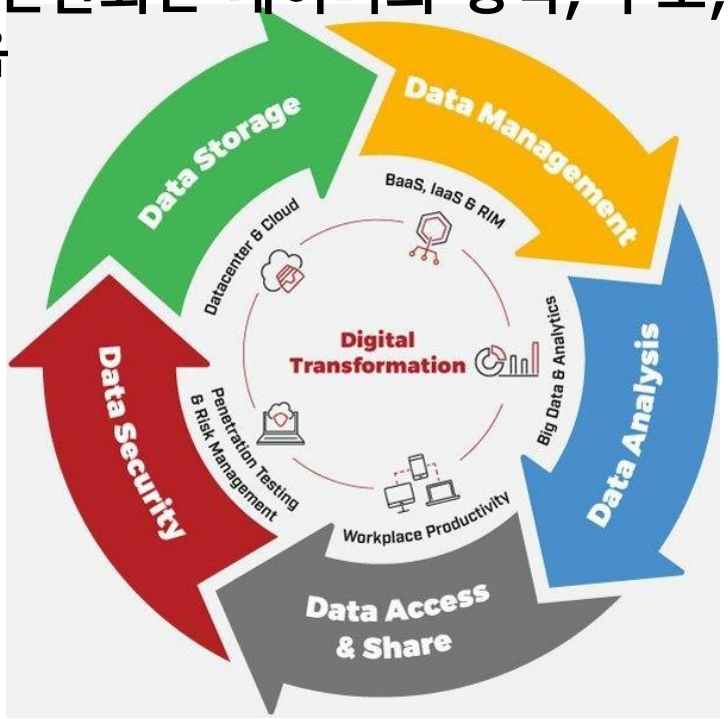
데이터 변환(Data Transformation)?

- 데이터 변환(Data Transformation)이란, 데이터를 변형하여 모델 학습에 적합한 형태로 만드는 과정.
  - 여기에는 주로 정규화(Normalization)와 표준화(Standardization)가 포함 됨.
  - 이러한 스케일링 기법들은 데이터의 스케일을 맞추고, 모델의 수렴을 빠르게 하거나 이상치에 덜 민감하게 만드는 등의 이점을 제공. 선택하는 스케일링 기법은 데이터와 모델에 따라 다르며, 주어진 문제에 가장 적합한 방법을 선택해야 함.

1. 데이터 변환(Data Transformation)

데이터 변환Data Transformation

- 데이터를 한 형식이나 구조에서 다른 형 식이나 구조로 변환
- 원본 데이터와 대상 데이터간에 필요한 데이터 변경 내용을 기반으로 데이터 변 환이 간단하거나 복잡 할 수 있음
- 데이터 변환은 일반적으로 수동 및 자동
- 단계가 혼합되어 수행
- 데이터 변환에 사용되는 도구 및 기술은 변환되는 데이터의 형식, 구조, 복잡성 및 볼륨에 따라 크게 다를 수 있음



1. 데이터 변환(Data Transformation)

주요 데이터 변환 기법

구분	기법	설명
정규화	L1, L2 Norm	벡터 크기를 1로 맞춤
스케일링	Min-Max, Standard	값의 범위를 맞추는 작업
로그 변환	log(x)	큰 값의 영향 완화, 분포 조정
루트 변환	sqrt(x)	약한 비선형 효과 조정
Box-Cox 변환	통계적 분포를 정규분포에 가깝게	
원-핫 인코딩	pd.get_dummies()	범주형 → 이진 벡터
라벨 인코딩	LabelEncoder	범주형 → 정수 값

2. 정규화

정규화(Normalization)?

- 정규화는 데이터의 값을 특정 범위로 조정하는 프로세스. 정규화는 속성값으로 -1.0 ~ 1.0과 같이 정해진 구간 내에 들도록 하는 기법

  - 일반적으로 데이터를 [0, 1] 또는 [-1, 1] 범위로 맞춤. 최소-최대 정규화(Min-Max Normalization)를 사용하여 최솟값을 0, 최댓값을 1로 변환하는 방법이 있다.
  - 최단 근접 분류와 군집화와 같은 거리 측정 등을 위해 특히 유용

2. 정규화

정규화 종류

- 최소-최대 정규화min-max normalization
  - 원본 데이터에 대하여 선형 변환 수행
  - 속성 A에 대한 최소값과 최대값을 $minA$와 $maxA$라고 가정
  - A의 값은 다음 계산식에 의해 v를 구간 에서의 값 v'으로 사상

$$v' = \frac{v - min_A}{max_A - min_A}\left(max'_A - min'_A\right) + min'_A$$

  - 최소-최대 정규화는 원본 데이터 값들 간의 관계를 보존
  - 정규화를 위한 입력이 A에 대한 원본 데이터 구간에서 벗어난 경우는 범위 초과out-of-bounds 오류 발생

2. 정규화

정규화 종류

- 구간화(Min-Max)
  - 최대값과 최소값을 사용하여 원 데이터의 최소값을 0, 최대값을 1로 만드는 방법이다. 여기에 100을 곱하여 지표관리 등 다양한 곳에 활용하기도 한다.

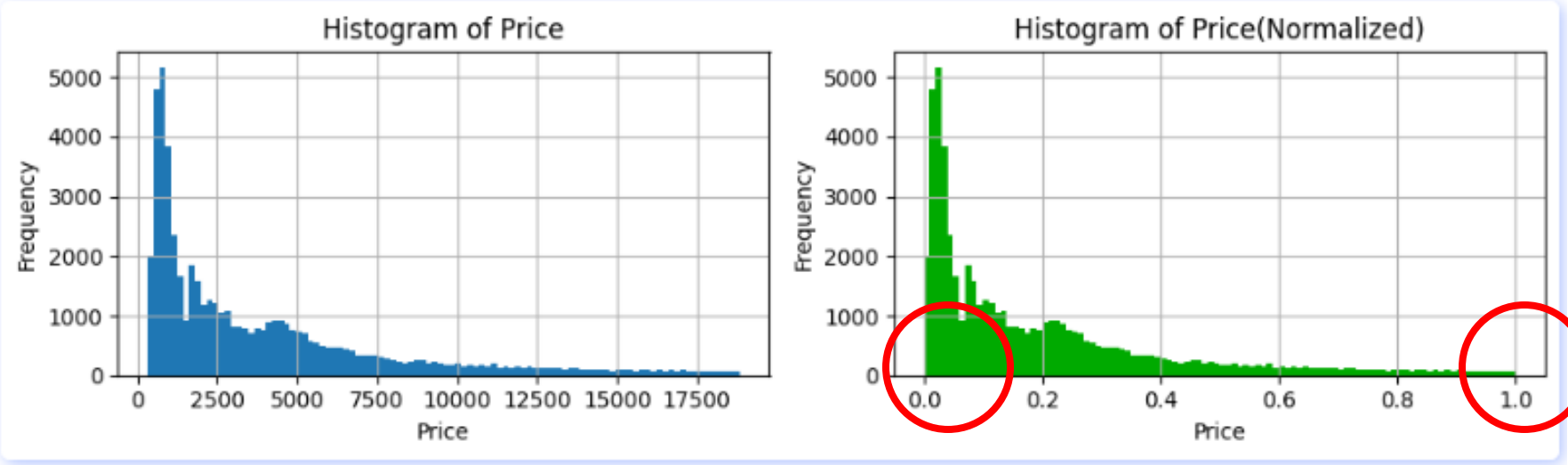
$$MinMax(x) = \frac{x - min(x)}{max(x) - min(x)}$$

  - 여기서 최소값을 빼는 것은 최소값을 0으로 만들며 범위(max - min)를 나누는 것은 범위를 1로 만들기 때문에 최종적으로 최대값이 1이 된다.

2. 정규화

정규화 종류

- 구간화(Min-Max)
  - 구간화 전과 후의 분포를 비교하면 다음과 같다.



2. 정규화

정규화 종류

- 소수 척도화decimal scaling
  - 속성 A 값들의 소수점을 이동해서 정규화
  - 이동되는 소수점의 수는 A의 최대 절대값에 의존
  - $Max\ v' < 1$을 만족하는 가장 작은 정수를 j라고 가정

  - A의 값 v는 다음 계산식에 의해 v'로 사상

$$v' = \frac{v}{10^j}$$

2. 정규화

정규화 예

- 아래 예제는 sklearn 라이브러리의 MinMaxScaler를 사용하여 최소-최대 정규화(Min-Max Normalization)를 수행하는 예시.
  - 최소-최대 정규화는 데이터를 특정 범위(일반적으로 0과 1 사이)로 변환하는 정규화 기법 중 하나. 주어진 데이터를 0과 1 사이의 값으로 변환.
  - 여기서 data는 2차원 리스트로, 각 행은 하나의 특성을 갖고 있다. MinMaxScaler를 사용하여 fit_transform() 메서드를 호출하여 데이터를 정규화.

2. 정규화

정규화 예

```
from sklearn.preprocessing import MinMaxScaler

# 데이터 생성 (예시)
data = [[10], [5], [3], [2], [8]]

# 최소-최대 정규화 적용
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)
print("정규화된 데이터:")
print(scaled_data)
>>
정규화된 데이터:
[[1.   ]
 [0.375]
 [0.125]
 [0.   ]
 [0.75 ]]
```

$$MinMax(x) = \frac{x - min(x)}{max(x) - min(x)}$$

1. $[10] \rightarrow \frac{10-2}{10-2} = 1$
2. $[5] \rightarrow \frac{5-2}{10-2} = \frac{3}{8}$
3. $[3] \rightarrow \frac{3-2}{10-2} = \frac{1}{8}$
4. $[2] \rightarrow \frac{2-2}{10-2} = 0$
5. $[8] \rightarrow \frac{8-2}{10-2} = \frac{6}{8} = \frac{3}{4}$

3. 표준화

표준화(Standardization)

- 평균과 표준편차를 사용하여 평균이 0, 표준편차를 1로 만드는 방법. 표준화는 데이터를 평균이 0이고 표준편차가 1인 분포로 변환하는 것을 의미. 각 데이터에서 평균을 빼고 표준편차로 나누어 변환.
- 흔히 z-scoring 이라고 하기도 한다.

$$Standardization(x) = \frac{x - mean(x)}{std(x)}$$

- 여기서 평균을 빼는 것은 데이터의 중심을 0으로 옮기는 것이며 표준편차를 나누는 것은 자료의 편차를 1로 만드는 것이 된다.

3. 표준화

표준화(Standardization)

- Z-score

Z is the Z-score,
X is the value,
μ is the mean of the population,
σ is the standard deviation of the population.

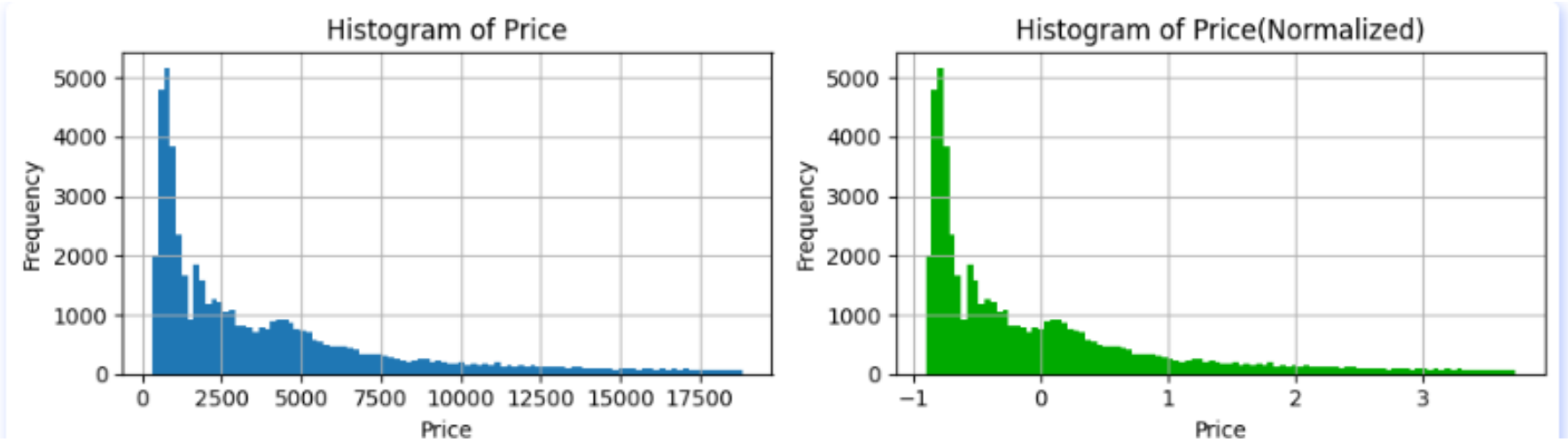
$$Z = \frac{X - \mu}{\sigma}$$

- 속성 A에 대한 값을 A의 평균과 표준편차를 기초로 정규화하는 방법
- Z-score 정규화는 속성 A의 실제 최소값과 최대값이 알려져 있지 않거나, 최소-최대 정규화에 큰 영향을 주는 이상치가 존재할 때 유용

3. 표준화

표준화(Standardization)

- 표준화(Standardization) 전과 후의 분포를 비교하면 다음과 같다.



3. 표준화

표준화(Standardization)

- 주어진 코드는 sklearn 라이브러리의 StandardScaler를 사용하여 데이터를 표준화(Standardization)하는 예시.
  - 표준화는 데이터의 평균을 0으로, 표준편차를 1로 만들어 데이터를 정규분포에 가깝게 만드는 작업. 따라서, 주어진 코드에서 data는 2차원 리스트로, 각 행은 하나의 특성을 나타냄.
  - StandardScaler를 사용하여 fit_transform() 메서드를 호출하여 데이터를 표준화.
  - 실행된 결과인 standardized_data는 표준화된 데이터를 나타내며, 각 값이 평균이 0이고 표준편차가 1인 정규분포에 가까워진 것을 확인할 수 있다.

3. 표준화

표준화(Standardization)

```
from sklearn.preprocessing import StandardScaler

# 데이터 생성 (예시)
data = [[10], [5], [3], [2], [8]]

# 표준화 적용
scaler = StandardScaler()
standardized_data = scaler.fit_transform(data)
print("표준화된 데이터:")
print(standardized_data)
>>
표준화된 데이터:
[[ 1.46341823]
 [-0.19955703]
 [-0.86474714]
 [-1.19734219]
 [ 0.79822813]]
```

$$Z = \frac{X - \mu}{\sigma}$$

평균 계산:

$$\text{평균} = \frac{10+5+3+2+8}{5} = \frac{28}{5} = 5.6$$

표준편차 계산:

$$\text{표준편차} = \sqrt{\frac{\sum (X_i - \text{평균})^2}{N}}$$

$$= \sqrt{\frac{(10-5.6)^2+(5-5.6)^2+(3-5.6)^2+(2-5.6)^2+(8-5.6)^2}{5}}$$

$$= \sqrt{\frac{20.8+0.36+10.96+14.44+4.84}{5}}$$

$$= \sqrt{\frac{51.4}{5}}$$

$$= \sqrt{10.28}$$

$$\approx 3.21$$

이제 Z 점수를 계산하여 데이터를 표준화합니다:

1. $[10] \rightarrow \frac{10-5.6}{3.21} \approx 1.37$
2. $[5] \rightarrow \frac{5-5.6}{3.21} \approx -0.19$
3. $[3] \rightarrow \frac{3-5.6}{3.21} \approx -0.81$
4. $[2] \rightarrow \frac{2-5.6}{3.21} \approx -1.12$
5. $[8] \rightarrow \frac{8-5.6}{3.21} \approx 0.75$

4. 수치형 데이터 이산화

수치형 데이터 이산화(Numeric Data Discretization)

- 이산화(Discretization) 란, 연속적인 수치형 데이터를 구간으로 나누어 이산적(불연속적)인 값으로 변환하는 것을 의미.
  - 즉, 실수 → 범주형 변수로 바꾸는 과정.
  - 예시: 나이(age)를 이산화하기

원본 값 (연속)	이산화 후 (범주)
23	청년
42	중년
65	노년

4. 수치형 데이터 이산화

주요 이산화 방법

방법	설명	예시
균등 간격 분할 (Equal-width)	전체 범위를 동일한 폭으로 나눔	[010), [1020), ...
균등 빈 분할 (Equal-frequency)	각 구간에 데이터 수가 같도록 나눔	사분위수 기준
클러스터 기반 (KMeans, DecisionTree)	데이터 특성 기반 자동 구간화	정보이득 기반
도메인 기반 수동 정의	도메인 지식을 반영하여 구간 정의	나이: 청년/중년/노년

4. 수치형 데이터 이산화

수치형 데이터 이산화 예

```
import pandas as pd

# 예시 데이터
ages = pd.Series([18, 25, 33, 45, 60, 70])

# 구간 정의
bins = [17.948, 35.333, 52.667, 70.0]
labels = ['청년', '중년', '노년']

# 이산화 수행
age_groups = pd.cut(ages, bins=bins, labels=labels)

# 결과 출력
df = pd.DataFrame({'나이': ages, '연령대': age_groups})
print(df)
```

bins: 경계값 리스트 (왼쪽 열림, 오른쪽 닫힘: (a, b])

```
   나이 연령대
0  18  청년
1  25  청년
2  33  청년
3  45  중년
4  60  노년
5  70  노년
```

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 정규화와 K-Means와 계층적 클러스터링을 모두 다뤄보는 파이썬 예제이다. 본 예제에서는 학생들의 성적 데이터를 분석하여 유의미한 인사이트를 얻어보는 과정을 다룬다. 데이터는 대학의 학생 성적 및 출석 데이터를 포함하고 있으며, 다음과 같은 컬럼으로 구성되어 있다.

컬럼명	설명
Semester	학기가 기록된 컬럼 (예: FSS2010)
Name	학생의 이름
Course	수강한 과목 (예: Database Systems I)
Mark	해당 과목에서 받은 성적
Attended	학생이 출석한 수업 횟수

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 학생 데이터 파이썬 분석
  - 데이터는 학생들의 수강 데이터로, 어떤 학기에 어떤 수업을 들었는지 출석과 성적까지 포함된 데이터이다. 한 학생이 여러 학기에 여러 과목을 들을 가능성이 크므로, 이름으로 그룹핑을 하여 이 학생의 출석과 성적이 어떤지 살펴보자.

```
# 패키지 불러오기
import pandas as pd
import numpy as np

# 데이터 읽기
student_data = pd.read_excel('datasets/ex1.xlsx')

# 데이터 개요 살펴보기
display(student_data.head())
```

	Semester	Name	Course	Mark	Attended
0	FSS2010	Alex Krausche	Database Systems I	1.3	13
1	FSS2010	Tanja Becker	Database Systems I	2.0	12
2	FSS2010	Mariano Selina	Database Systems I	1.7	5
3	FSS2010	Otto Blacher	Database Systems I	2.3	13
4	FSS2010	Frank Fester	Database Systems I	2.0	13

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 학생 데이터 파이썬 분석

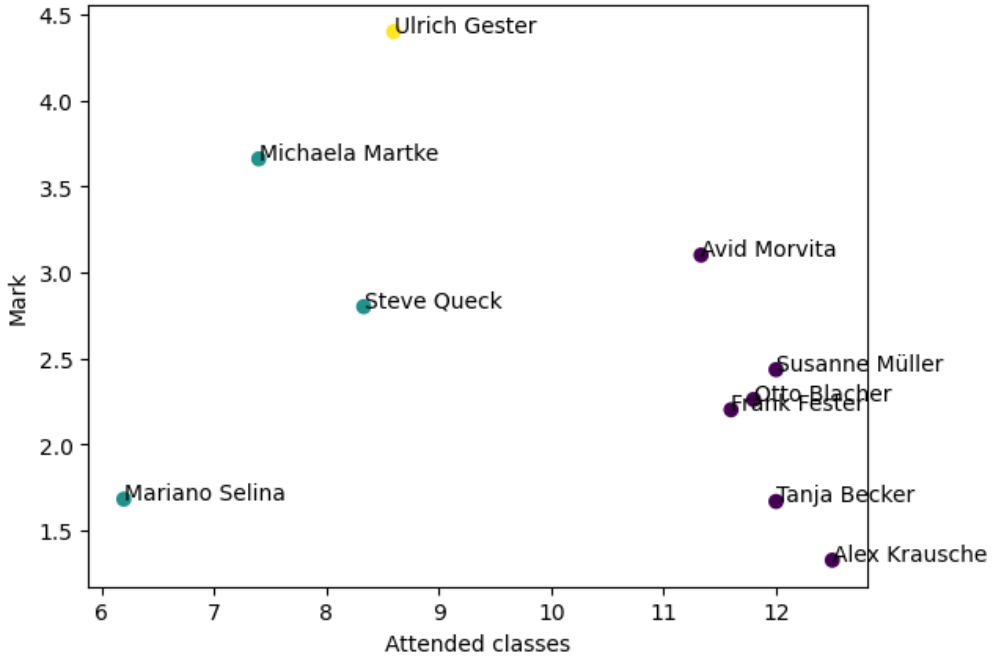
```
# 숫자형 데이터만 선택하여 그룹핑 후 평균 계산
students = student_data.groupby('Name').mean(numeric_only=True)

# 다시 데이터 살펴보기
display(students.head())
```

Name	Mark	Attended
Alex Krausche	1.325	12.500000
Avid Morvita	3.100	11.333333
Frank Fester	2.200	11.600000
Mariano Selina	1.680	6.200000
Michaela Martke	3.660	7.400000

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 학생 데이터 파이썬 분석
  - 학생들을 군집으로 나눠보자. 출석과 성적 데이터 기반이니 대략적으로 1) 출석도 좋고 성적도 좋은 학생군 2) 출석은 나쁘지만 성적은 좋은 학생군 3) 출석도 나쁘고 성적도 나쁜 학생군 으로 나눠보도록 하자.



5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

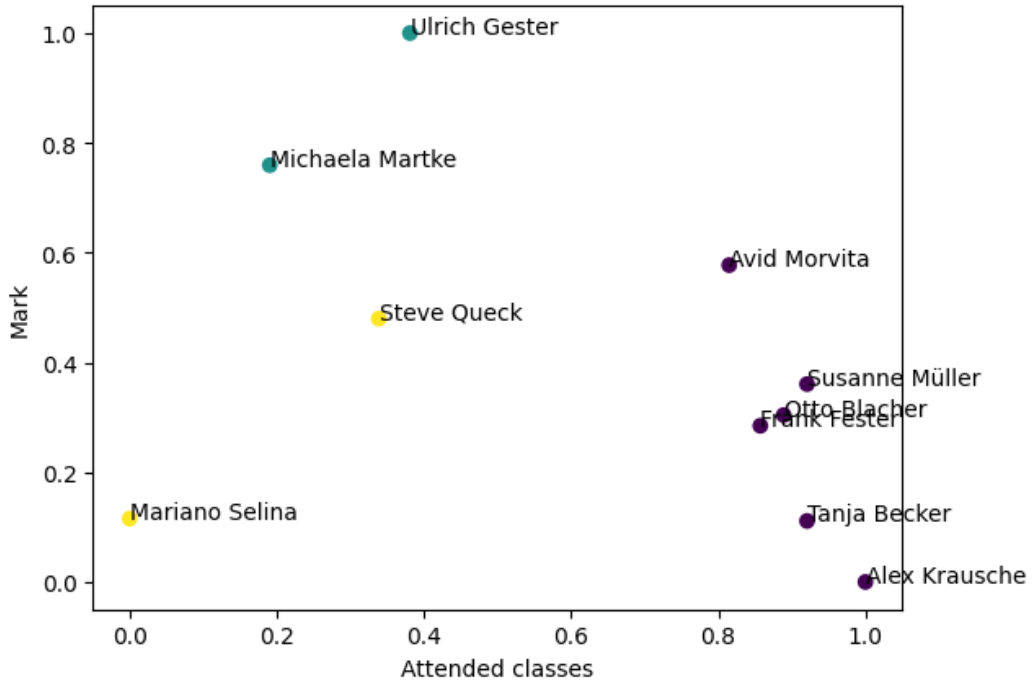
- **학생 데이터 파이썬 분석**

```
from sklearn.cluster import KMeans

import matplotlib.pyplot as plt  # 추가된 부분

# k=3 클러스터 생성
estimator = KMeans(n_clusters = 3)
cluster_ids = estimator.fit_predict(students)

# 플롯
plt.scatter(students['Attended'], students['Mark'], c=cluster_ids)
plt.xlabel("Attended classes")
plt.ylabel("Mark")

# 범례 달기
for name, mark, attended in students.itertuples():
    plt.annotate(name, (attended, mark))

plt.show()
```

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 학생 데이터 파이썬 분석
  - 학생 데이터를 클러스터링했더니 '출석'에 좌우된 것이 보인다. 성적은 0~5점 사이인데 반해 출석은 0~13까지 범위가 더 크기 때문이다. 두 개의 클래스의 단위가 다를 때는 정규화(normalization)를 해주어야 한다. 여기서는 MinMaxScaler()로 정규화를 진행하고자 한다.



5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

● 학생 데이터 파이썬 분석

```
from sklearn.preprocessing import MinMaxScaler

# normalizer 생성
min_max_scaler = MinMaxScaler()

# 정규화하기
students[['Mark', 'Attended']] = min_max_scaler.fit_transform(students[['Mark', 'Attended']])

# 클러스터링 생성
estimator = KMeans(n_clusters = 3)
cluster_ids = estimator.fit_predict(students)


plt.scatter(students['Attended'], students['Mark'], c=cluster_ids)
plt.xlabel("Attended classes")
plt.ylabel("Mark")

for name, mark, attended in students.itertuples():
    plt.annotate(name, (attended, mark))

plt.show()
```

5. Data Transformation 종합 예제

클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 같은 데이터로 계층적 클러스터링을 해보자. 어떤 linkage mode(군집 간의 거리를 재는 방식)가 적합한지 반복해서 돌려보고 시각화

```
from scipy.cluster.hierarchy import dendrogram, linkage

# 최단연결법은 single, 평균연결법은 average, 최장연결법은 complete으로 표기한다.
modes = ['single', 'average', 'complete']

plt.figure(figsize=(20,5))

# subplot() 함수를 사용하여 서브 플롯을 추가하고 반환 값을 y_axis라는 변수에 할당
# sharey 매개 변수를 사용하여 서브 플롯에 대한 모든 호출에 이 변수를 전달하여 모든 플롯이 동일한 y 축 사용
y_axis = None

# 모든 linkage mode 반복 생성
for i, mode in enumerate(modes):
    # 서브플롯 추가, y축은 공유
    y_axis = plt.subplot(1, 4, i + 1, sharey = y_axis)

    # 레이블링
    plt.title('Dendrogram - linkage mode: {}'.format(mode))
    plt.xlabel('distance')
    plt.ylabel('student')

    # 클러스터링
    clustering = linkage(students[['Mark', 'Attended']], mode)

    # 덴드로그램
    dendrogram(clustering, labels=list(students.index), orientation='right')
plt.tight_layout()
plt.show()
```

5. Data Transformation 종합 예제

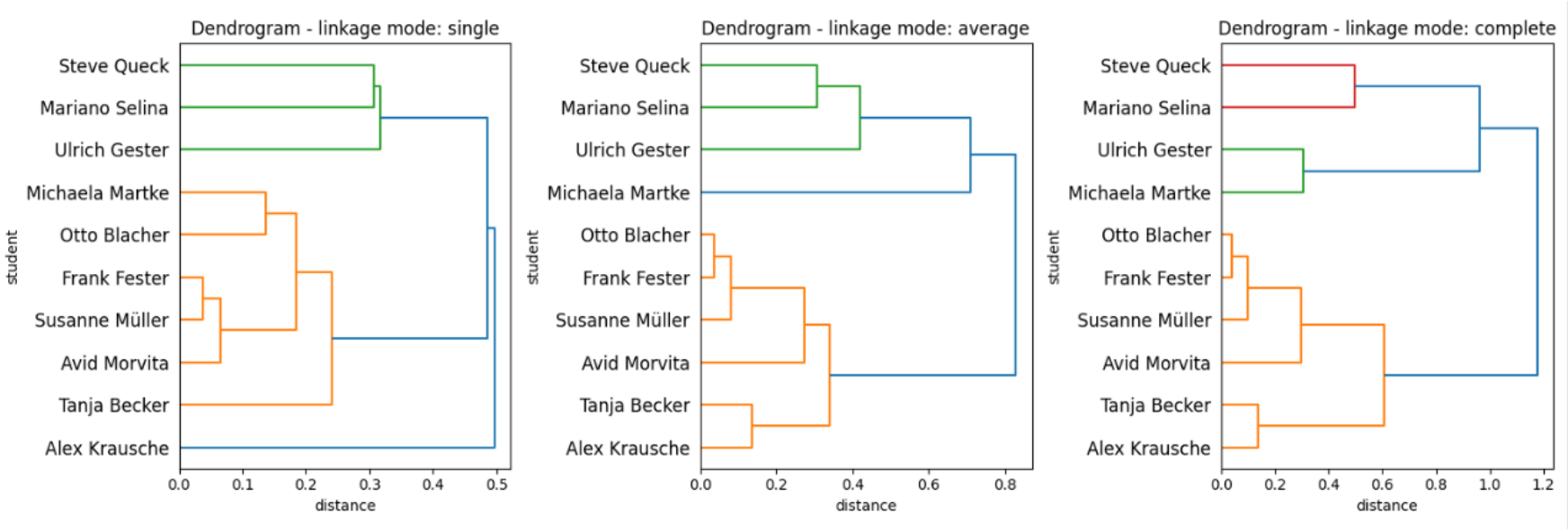
클러스터링 파이썬 연습 예제 – 학생 성적 데이터 분석

- 학생 데이터 파이썬 분석

① Single Linkage는 가장 작은 거리로 먼저 클러스터를 형성하여 유사한 학생들을 빠르게 묶는다.

② Average Linkage는 균형 잡힌 클러스터를 만들고, 이상치의 영향을 적절히 완화.

③ Complete Linkage는 가장 밀접한 클러스터링을 수행하지만, 그룹 간 거리가 더 커질 수 있다.



6.Quiz

Question 1

다음 데이터세트는 diamonds.csv” 이다. Min-Max 정규화 하라.

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31

6.Quiz

Answer 1

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("datasets/diamonds.csv")
df.head(2)
>>
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31

6.Quiz

Answer 1

```
df_nums = df.select_dtypes(include = "number")
arr_nums_nor = MinMaxScaler().fit_transform(df_nums)
arr_nums_nor[:2, ]
>>
array([[0.00623701, 0.51388889, 0.23076923, 0.        , 0.36778399,
        0.06757216, 0.07641509],
       [0.002079  , 0.46666667, 0.34615385, 0.        , 0.36219739,
        0.06519525, 0.07264151]])


df_nums_nor = pd.DataFrame(arr_nums_nor, columns = df_nums.columns)
df_nums_nor.head(2)
>>
```

	carat	depth	table	price	x	y	z
0	0.006237	0.513889	0.230769	0.0	0.367784	0.067572	0.076415
1	0.002079	0.466667	0.346154	0.0	0.362197	0.065195	0.072642

```
df_nums_nor.agg(["min", "max"])
>>
```

	carat	depth	table	price	x	y	z
min	0.0	0.0	0.0	0.0	0.0	0.0	0.0
max	1.0	1.0	1.0	1.0	1.0	1.0	1.0

6.Quiz

Answer 1

```
model_nor2 = MinMaxScaler().fit(df_nums)
arr_nums_nor2 = model_nor2.transform(df_nums)
arr_nums_nor2[:2, ]
>>
array([[0.00623701, 0.51388889, 0.23076923, 0.        , 0.36778399,
        0.06757216, 0.07641509],
       [0.002079  , 0.46666667, 0.34615385, 0.        , 0.36219739,
        0.06519525, 0.07264151]])

print(model_nor2.data_min_)
## array([2.00e-01, 4.30e+01, 4.30e+01, 3.26e+02, 0.00e+00, 0.00e+00, 0.00e+00])

print(model_nor2.data_max_)
## array([5.0100e+00, 7.9000e+01, 9.5000e+01, 1.8823e+04, 1.0740e+01, 5.8900e+01,
3.1800e+01])

print(model_nor2.data_range_)
## array([4.8100e+00, 3.6000e+01, 5.2000e+01, 1.8497e+04, 1.0740e+01, 5.8900e+01,
3.1800e+01])
```

6.Quiz

Answer 1

```
print(model_nor2.feature_names_in_)
## array(['carat', 'depth', 'table', 'price', 'x', 'y', 'z'], dtype=object)
>>
['carat' 'depth' 'table' 'price' 'x' 'y' 'z']

arr_nums_inv = model_nor2.inverse_transform(arr_nums_nor2)
arr_nums_inv[:2, ]
>>
array([[2.30e-01, 6.15e+01, 5.50e+01, 3.26e+02, 3.95e+00, 3.98e+00,
        2.43e+00],
       [2.10e-01, 5.98e+01, 6.10e+01, 3.26e+02, 3.89e+00, 3.84e+00,
        2.31e+00]])

df_nums_inv = pd.DataFrame(arr_nums_inv, columns = df_nums.columns)
df_nums_inv.head(2)
>>
```

	carat	depth	table	price	x	y	z
0	0.23	61.5	55.0	326.0	3.95	3.98	2.43
1	0.21	59.8	61.0	326.0	3.89	3.84	2.31

6.Quiz

Question 2

다음 데이터세트는 diamonds.csv” 이다. Standardization를 하라.

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31

6.Quiz

Answer 2

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

df = pd.read_csv("datasets/diamonds.csv")
df.head(2)
>>
```

	carat	cut	color	clarity	depth	table	price	x	y	z
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31

6.Quiz

Answer 2

```
df_nums = df.select_dtypes(include = "number")
arr_nums_nor = StandardScaler().fit_transform(df_nums)
arr_nums_nor[:2, ]
## array([[-1.19816781, -0.17409151, -1.09967199, -0.90409516, -1.58783745, -
1.53619556, -1.57112919],
##        [-1.24036129, -1.36073849,  1.58552871, -0.90409516, -1.64132529, -1.65877419, -
1.74117497]])
>>
array([[-1.19816781, -0.17409151, -1.09967199, -0.90409516, -1.58783745,
        -1.53619556, -1.57112919],
       [-1.24036129, -1.36073849,  1.58552871, -0.90409516, -1.64132529,
        -1.65877419, -1.74117497]])

df_nums_nor = pd.DataFrame(arr_nums_nor, columns = df_nums.columns)
df_nums_nor.head(2)
>>
carat       depth       table       price       x           y           z
0           -1.198168               -0.174092               -1.099672               -0.904095               -
1.587837 -1.536196               -1.571129
1           -1.240361               -1.360738               1.585529 -0.904095               -1.641325
            -1.658774               -1.741175
```

6.Quiz

Answer 2

```
model_nor3 = StandardScaler().fit(df_nums)
arr_nums_nor3 = model_nor3.transform(df_nums)
arr_nums_nor3[:2, ]
## array([[-1.19816781, -0.17409151, -1.09967199, -0.90409516, -1.58783745, -
1.53619556, -1.57112919],
##        [-1.24036129, -1.36073849,  1.58552871, -0.90409516, -1.64132529, -1.65877419, -
1.74117497]])
>>
array([[-1.19816781, -0.17409151, -1.09967199, -0.90409516, -1.58783745,
        -1.53619556, -1.57112919],
       [-1.24036129, -1.36073849,  1.58552871, -0.90409516, -1.64132529,
        -1.65877419, -1.74117497]])

pd.DataFrame([model_nor3.mean_,
              model_nor3.var_,
              model_nor3.scale_],
             index = ["mean", "var", "std"],
             columns = model_nor3.feature_names_in_)
>>
```

	carat	depth	table	price	x	y	z
mean	0.797940	61.749405	57.457184	3.932800e+03	5.731157	5.734526	3.538734
var	0.224682	2.052366	4.992856	1.591533e+07	1.258324	1.304447	0.498002
std	0.474007	1.432608	2.234470	3.989403e+03	1.121750	1.142124	0.705692

6.Quiz

Answer 2

```
arr_nums_inv = model_nor3.inverse_transform(arr_nums_nor3)
arr_nums_inv[:2, ]
## array([[2.30e-01, 6.15e+01, 5.50e+01, 3.26e+02, 3.95e+00, 3.98e+00, 2.43e+00],
##        [2.10e-01, 5.98e+01, 6.10e+01, 3.26e+02, 3.89e+00, 3.84e+00, 2.31e+00]])
>>
array([[2.30e-01, 6.15e+01, 5.50e+01, 3.26e+02, 3.95e+00, 3.98e+00,
        2.43e+00],
       [2.10e-01, 5.98e+01, 6.10e+01, 3.26e+02, 3.89e+00, 3.84e+00,
        2.31e+00]])
```

6.Quiz

Question 3

mtcars 데이터셋(mtcars.csv)의 qsec 컬럼을 최소최대 척도(Min-Max Scale)로 변환한 후 0.5보다 큰 값을 가지는 레코드 수를 구하시오.

```
import pandas as pd # pandas import

df = pd.read_csv('datasets/mtcars.csv') # df에 mtcars.csv를 읽어 데이터프레임으로 저장
df.head() # df의 앞에서 5개 데이터 출력
```

>>

Unnamed: 0	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3

6.Quiz

Answer 3

방법 1 sklearn의 MinMaxScaler를 이용해서 변환

```
from sklearn.preprocessing import MinMaxScaler # sklearn의 min-max scaler import

scaler = MinMaxScaler() # MinMaxScaler 객체 생성

# scaler에 데이터를 넣어서 모델을 만들고 값을 덮어씌움
df['qsec'] = scaler.fit_transform(df[['qsec']])

# qsec가 0.5보다 큰 데이터만 색인해서 길이를 구함
answer = len(df[df['qsec']>0.5])
print(answer)
>>
            9
```

6.Quiz

Answer 3

방법 2
Min-Max Scale을 수식으로 만들어서 변환시킴

```
df['qsec'] = (df['qsec'] - df['qsec'].min()) / (df['qsec'].max() - df['qsec'].min())

answer = len(df[df['qsec']>0.5])
print(answer)
```

>>

9

『 3과목 :』
데이터 마트와 데이터 전처리

- Data Mart & Data Preprocessing
- Data Structures
- Data Gathering(Collect, Acquisition), Data Ingestion
- Data Invest & Exploratory Data Analysis, Data Visualization
- Data Cleansing (정제)
- Data Integration (통합)
- Data Reduction (축소)
- Data Transformation (변환)
- **Feature Engineering & Data Encoding**
- Cross Validation & Data Splitting
- Data Quality Assessment and Model Performance Evaluation
- 『3과목』 Self 점검

개요

학습목표

- 이 워크샵에서는 특성 공학(Feature Engineering)과 데이터 인코딩(Data Encoding),데이터 축소(Data Reduction에 대해 알 수 있습니다.

눈높이 체크

- 특성 공학(Feature Engineering)과 데이터 인코딩(Data Encoding) 에 대해 들어보셨나요?

1. Feature Engineering

특성 공학(Feature Engineering)?

- 특성 공학(Feature Engineering)은 머신러닝 모델에 입력으로 제공될 특성(또는 변수)을 만들거나 변형하는 과정.

  - 이를 통해 모델의 성능을 향상시키고, 데이터로부터 더 유용한 정보를 추출할 수 있다.
  - 특성 공학은 모델의 성능을 향상시키고 데이터의 정보를 최대한 활용하기 위해 매우 중요한 단계. 하지만 도메인 지식과 실험을 통한 탐색이 필요. 데이터의 특성을 잘 이해하고, 문제에 적합한 특성 공학 기법을 적용하는 것이 중요.

1. Feature Engineering

특성 공학(Feature Engineering)?

- 일반적인 특성 공학의 목표는 다음과 같다.
  - 새로운 특성 생성: 기존의 특성을 기반으로 새로운 특성을 만들어내는 것입니다. 예를 들어, 날짜 데이터에서 요일, 월, 계절 등을 추출하거나, 길이에 관련된 특성을 만들거나, 텍스트 데이터에서 단어 수 또는 패턴을 추출하는 등이 있습니다.
  - 특성 변형: 기존의 특성을 변형하여 새로운 관점에서 데이터를 표현합니다. 예를 들어, 로그, 제곱근, 표준화, 정규화 등을 사용하여 데이터의 분포를 조정할 수 있습니다.
  - 차원 축소: 고차원의 데이터를 저차원으로 변환하여 더 간결하고 효과적인 데이터를 생성합니다. 주성분 분석(PCA), t-SNE 등의 기법을 사용하여 차원을 축소할 수 있습니다.
  - 외부 데이터 사용: 외부 데이터를 활용하여 새로운 특성을 추가하는 것도 중요한 특성 공학의 한 부분입니다. 예를 들어, 지리적 데이터를 활용하여 거리 기반 특성을 만드는 등이 있습니다.

1. Feature Engineering

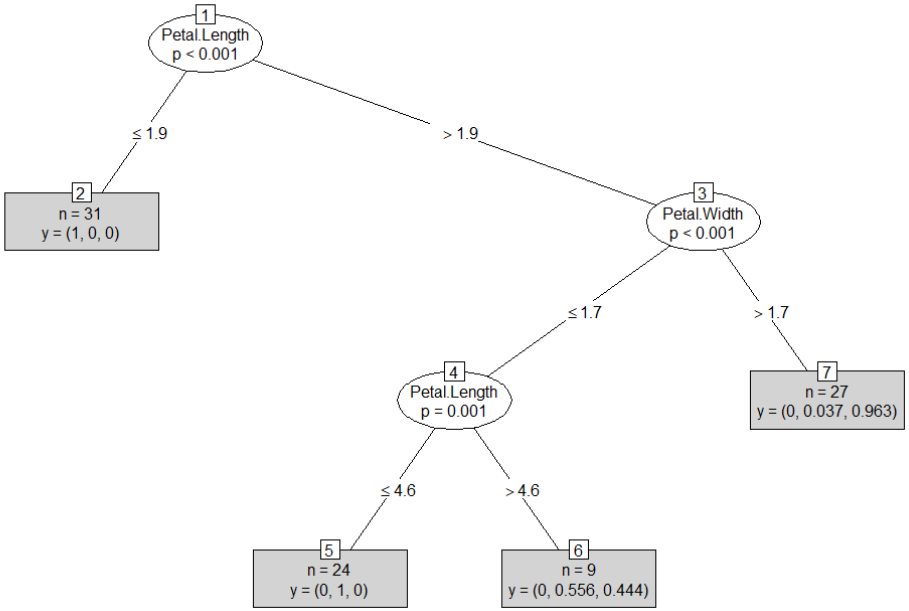
특징 값의 종류

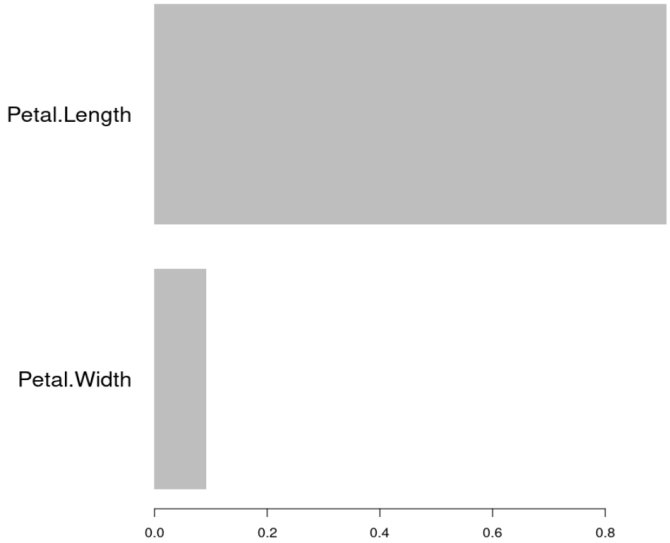
- 수치형 특징
  - 예) iris의 네 개 특징은 실수
  - 거리 개념이 있음
  - 실수 또는 정수 또는 이진값
- 범주형 특징
  - 학점, 수능 등급, 혈액형, 지역 등
  - 순서형: 학점, 수능 등급 등
    - 거리 개념이 있음. 순서대로 정수를 부여하면 수치형으로 취급 가능
  - 이름형
    - 혈액형, 지역 등으로 거리 개념이 없음
    - 보통 원핫one-hot 코드로 표현. 예) A형(1,0,0,0), B형(0,1,0,0), O형(0,0,1,0), AB형(0,0,0,1)

1. Feature Engineering

특징 공간에서 데이터 분포

- petal width(수직 축)에 대해 Setosa는 아래쪽, Virginica는 위쪽에 분포→ petal Length 특징은 분별력discriminating power이 뛰어남
- sepal width 축은 세 부류가 많이 겹쳐서 분별력이 낮음
- 전체적으로 보면, 세 부류가 3차원 공간에서 서로 다른 영역을 차지하는데 몇 개 샘플은 겹쳐 나타남





1. Feature Engineering

피처 영향력 살펴보기

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import matplotlib.pyplot as plt

# Load data
df_train = pd.read_csv("datasets/titanic_train.csv")
df_test = pd.read_csv("datasets/titanic_test.csv")

# Preprocessing: fill missing values and encode categorical variables
df_train['age'].fillna(df_train['age'].median(), inplace=True)
df_train['fare'].fillna(df_train['fare'].median(), inplace=True)
df_train['embarked'].fillna(df_train['embarked'].mode()[0], inplace=True)

# Convert categorical variables to dummy/indicator variables
df_train = pd.get_dummies(df_train, columns=['pclass', 'gender', 'embarked'], drop_first=True)
```

1. Feature Engineering

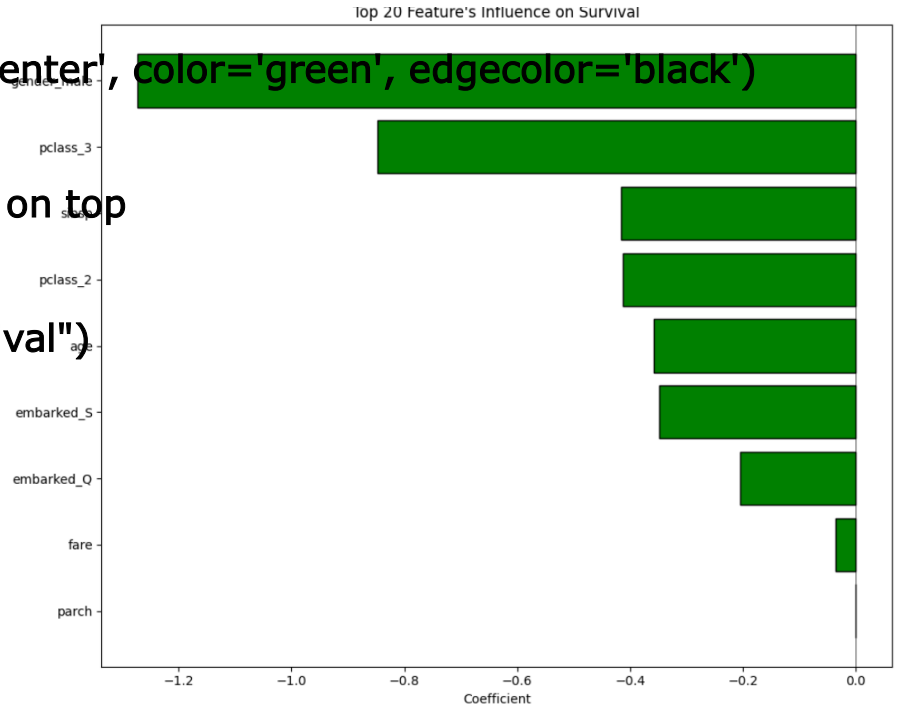
피처 영향력 살펴보기

```
# Define feature set and target variable
X = df_train.drop(['survived', 'name', 'ticket', 'cabin', 'body', 'home.dest'], axis=1,
errors='ignore')
y = df_train['survived']

# Split the data (optional, if you want to evaluate on a test set)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a logistic regression model pipeline
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('logisticregression', LogisticRegression(max_iter=1000))
])

# Fit the model
pipeline.fit(X_train, y_train)

# Get feature coefficients
cols = X.columns.tolist()
coef = pipeline.named_steps['logisticregression'].coef_[0]
coef_df = pd.DataFrame({'Feature': cols, 'Coefficient': coef})
```

1. Feature Engineering

피처 영향력 살펴보기

```
# Select the top 20 features by absolute coefficient
top_coef_df =
coef_df.reindex(coef_df.Coefficient.abs().sort_values(ascending=False).index).head(20)

# Plot the top features by coefficient
plt.rcParams['figure.figsize'] = [10, 8]
fig, ax = plt.subplots()
y_pos = np.arange(len(top_coef_df))
ax.barh(y_pos, top_coef_df.Coefficient, align='center', color='green', edgecolor='black')
ax.set_yticks(y_pos)
ax.set_yticklabels(top_coef_df['Feature'])
ax.invert_yaxis()  # Highest importance feature on top
ax.axvline(0, color='black', linewidth=0.5)
ax.set_xlabel('Coefficient')
ax.set_title("Top 20 Feature's Influence on Survival")

plt.tight_layout()
plt.show()
```



2. Data Encoding

원핫 인코딩(One-Hot Encoding):

- 원핫 인코딩(One-Hot Encoding)은 범주형 데이터를 머신러닝 알고리즘이 이해할 수 있는 형태로 변환하는 방법 중 하나. 주로 범주형 변수의 각 범주를 새로운 이진 특성으로 변환.
- 파이썬에서 pandas나 scikit-learn을 사용하여 원핫 인코딩을 수행. 아래는 pandas를 사용하여 원핫 인코딩을 수행하는 예제 코드입니다.

2. Data Encoding

원핫 인코딩(One-Hot Encoding):

```
import pandas as pd

# 샘플 데이터셋
data = {
    'color': ['red', 'blue', 'green', 'red', 'yellow']
}

# DataFrame 생성
df = pd.DataFrame(data)

# 원핫 인코딩 적용
df_encoded = pd.get_dummies(df['color'], prefix='color')

# 변환된 데이터 확인
print("변환된 데이터:")
print(df_encoded)
```

>>

변환된 데이터:

	color_blue	color_green	color_red	color_yellow
0	False	False	True	False
1	True	False	False	False
2	False	True	False	False
3	False	False	True	False
4	False	False	False	True

2. Data Encoding

원핫 인코딩(One-Hot Encoding):

- 위 코드에서 pd.get_dummies() 함수를 사용하여 'color' 열을 원핫 인코딩. 각 범주는 새로운 이진 특성으로 변환되었으며, 변환된 데이터셋은 df_encoded에 저장되어 출력.
- 실제로는 범주형 변수의 종류와 데이터에 따라 적절한 원핫 인코딩 방식을 선택. 인코딩된 데이터는 기존 변수보다 차원이 증가할 수 있으므로, 데이터 크기와 모델 성능에 영향을 미칠 수 있음.

2. Data Encoding

데이터 인코딩(Data Encoding)

- 데이터 인코딩은 주로 범주형 데이터를 모델링하거나 분석하기 쉬운 형태로 변환하는데 사용.

  - 라벨 인코딩 (Label Encoding): 범주형 변수를 숫자형으로 변환하는 기법입니다. 각 범주형 변수의 고유한 값들을 숫자로 매핑하여 변환합니다. 이를 통해 모델이 이해할 수 있는 형태로 변환하지만, 값 사이의 순서나 관계를 시사하는 것은 아닙니다. 예를 들어, {고양이, 개, 새}와 같은 범주형 변수를 {0, 1, 2}와 같은 숫자로 변환합니다.
  - 더미 변수화 (Dummy Variable Creation 또는 One-Hot Encoding):범주형 변수를 이진형 변수로 변환하는 기법입니다. 각 범주에 대한 새로운 이진형 변수(0 또는 1)를 생성합니다. 원-핫 인코딩이라고도 불리며, 각 범주를 별도의 열로 만들어 해당 범주에 해당하는 열은 1로 표시하고, 나머지 열은 0으로 표시합니다. 이 방법은 범주 간의 관계나 순서를 고려하지 않고, 각 범주를 독립적으로 처리할 수 있도록 합니다.

2. Data Encoding

데이터 인코딩(Data Encoding)

- 라벨 인코딩은 순서가 있는 범주형 데이터에 적합하며, 더미 변수화는 순서가 없는 범주형 데이터에 적합합니다. 데이터와 모델의 요구 사항에 따라 적절한 데이터 인코딩 방법을 선택하여 사용해야 합니다.

2. Data Encoding

라벨 인코딩

- 라벨 인코딩은 범주형 데이터를 숫자로 변환하는 과정.
  - 파이썬에서 LabelEncoder를 사용하여 라벨 인코딩을 수행할 수 있다. sklearn.preprocessing 모듈에 포함되어 있다.
  - 아래 코드는 categories 리스트의 각 범주를 라벨 인코딩하여 숫자로 변환. fit_transform 메서드를 사용하여 변환을 수행하고, 변환된 결과를 encoded_labels에 저장. 결과는 각 범주에 대해 할당된 숫자.

```
from sklearn.preprocessing import LabelEncoder

# 범주형 데이터 예시
categories = ['고양이', '개', '고양이', '새', '개']

# LabelEncoder 객체 생성
label_encoder = LabelEncoder()

# 라벨 인코딩 수행
encoded_labels = label_encoder.fit_transform(categories)
print(encoded_labels)

>>
[1 0 1 2 0]
```

2. Data Encoding

더미 변수화 (Dummy Variable Creation 또는 One-Hot Encoding)

- 파이썬에서 더미 변수화 또는 원-핫 인코딩을 수행하는 가장 흔한 방법은 pandas 라이브러리의 get_dummies() 함수를 사용하는 것. 이 함수를 사용하여 범주형 변수를 이진형 변수로 변환할 수 있다.
  - 아래 코드는 pet 열을 가진 데이터프레임을 생성하고, pd.get_dummies() 함수를 사용하여 pet 열을 더미 변수로 변환. concat() 함수를 사용하여 원본 데이터프레임과 더미 변수를 합쳐서 df_encoded라는 새로운 데이터프레임을 생성.
  - 더미 변수화를 통해 각 범주가 이진형 변수로 변환되며, 해당 범주에 속할 경우 1로 표시되고 속하지 않을 경우 0으로 표시. 이렇게 변환된 데이터프레임은 머신러닝 모델에 바로 사용될 수 있다.

2. Data Encoding

더미 변수화

```
import pandas as pd

# 범주형 데이터 예시
data = {'pet': ['고양이', '개', '새', '고양이', '개']}
df = pd.DataFrame(data)

# 더미 변수화 (원-핫 인코딩)
dummy_variables = pd.get_dummies(df['pet'])

# 원본 데이터프레임과 더미 변수를 합치기
df_encoded = pd.concat([df, dummy_variables], axis=1)
print(df_encoded)
>>
```

```
    pet     개    고양이     새
0  고양이  False   True  False
1    개   True  False  False
2    새  False  False   True
3  고양이  False   True  False
4    개   True  False  False
```

```
      pet  고양이  개  새
0  고양이    1  0  0
1    개    0  1  0
2    새    0  0  1
3  고양이    1  0  0
4    개    0  1  0
```

THANK YOU.