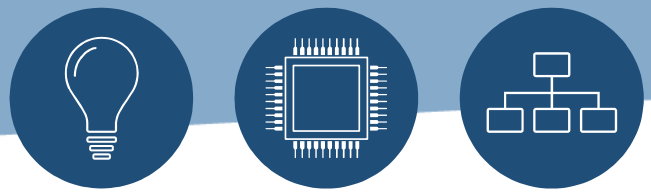
[일머리사관학교 디지털헬스케어]

EDA 활용

경남대학교 창의융합대학 교수 유현주
comjoo@kyungnam.ac.kr



강의 진행 안내

- **수업 자료 공유 URL**
  - https://github.com/yoohjoo/
  - 수업에 필요한 강의 자료 다운로드 저장

- **소통을 위한 메일 주소:** comjoo@kyungnam.ac.kr
  yoohjoo94@gmail.com

Contents

EDA활용

- 통계 기반 데이터 분석 : 와인 성분과 품질등급
- 상관분석 : 피마 인디언 당뇨 여부
  제로콜라와 기온속성 관계
  전복의 나이
- 회귀분석 : 연도별 날씨 데이터 회귀분석

와인 성분과 품질등급 데이터

통계 기반 데이터 분석

- 예제1: 와인 품질 예측하기

분석의 목표

와인 속성을 분석하여 품질 등급을 예측한다

분석 방법

기술 통계, 회귀 분석, T-검정, 히스토그램

데이터 준비

데이터 파일 병합

데이터 탐색

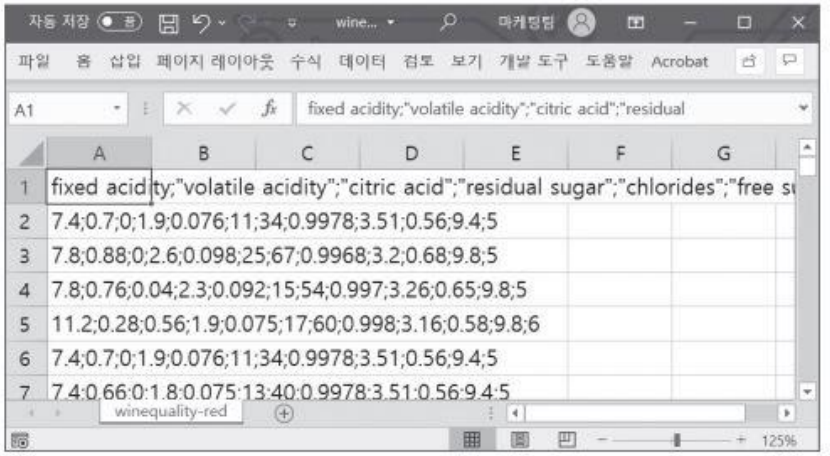
데이터 정보 확인	기술 통계량 확인

데이터 모델링

데이터를 두 그룹으로 비교 분석	품질 등급 예측

데이터 준비

- 데이터 수집 : 오픈 데이터 활용
  - 캘리포니아 어바인 대학의 머신러닝 저장소에서 제공하는 오픈 데이터를 사용
  - 다운로드한 파일은 홈 폴더의 Data 폴더를 만든 후에 저장



- 다운로드 CSV파일 데이터 정리
  - 현재 데이터 파일의 데이터 구분자를 확인
  - 세미콜론을 열 구분자 로 인식하도록 파일 불러오기 → 파일 백업 저장

데이터 준비

1. 데이터 파일 열어 데이터 정보 확인
   - 입력변수 : fixed acidity(고정산), volatile acidity(휘발산), citric acid(구연산), residual sugar(잔당), chlorides(염화물), free sulfur dioxide(유리 이산화황), total sulfur dioxide(총 이산화황), density(밀도), pH(pH), sulphates(황산염), alcohol(알콜) 11개 속성
   - 출력변수 : quality(품질등급)
2. 레드와인 데이터프레임과 화이트와인 데이터프레임의 내용을 불러온 후 ‘type’ 열을 추가하여 각 “red”, “white”로 표시
3. 각 데이터프레임 규모를 확인하기
4. 두 데이터프레임을 합치기 → 결합된 데이터프레임의 정보 확인
5. 결합된 데이터프레임을 파일로 백업 저장

데이터 탐색

- 기본 정보 확인 : wine.info()
  - 전체 샘플은 6,497개이고 속성을 나타내는 열은 13개, 각 속성의 이름은 type부터 qualit까지
  - 속성 중에서 실수 타입(float64)은 11개, 정수 타입(int64)은 1개(quality),
    객체 타입(object)이 1개(type)
  - 독립 변수(x)는 type부터 alcohol 까지 12개, 종속 변수(y)는 1개(quality)
- 함수로 기술 통계량 확인 :
  - wine.describe()
  - sorted(wine.quality.unique())
  - wine.quality.value_count()
- 그룹별 비교
  - wine.groupby('type')['quality'].describe()
  - wine.groupby('type')['quality'].agg(['mean', 'std'])

데이터 모델링

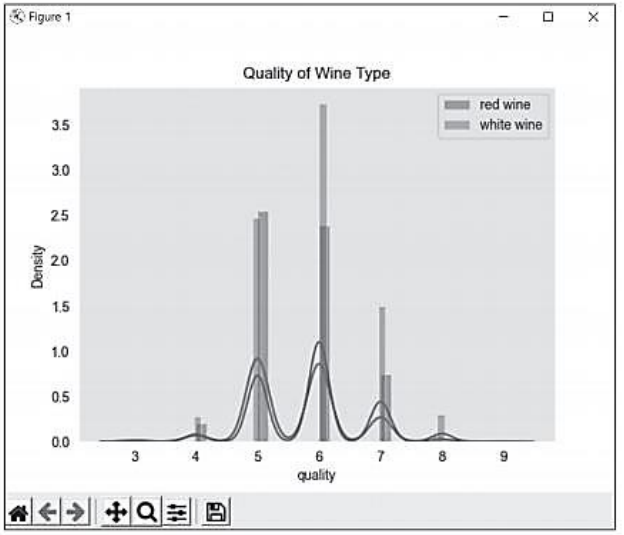
- t-검정과 회귀 분석으로 그룹 비교하기
  - t-검정을 위해서는 scipy 라이브러리 패키지를 사용
  - 회귀 분석을 위해서는 statsmodels 라이브러리 패키지를 사용

```
from scipy import stats
from statsmodels.formula.api import ols, glm
```

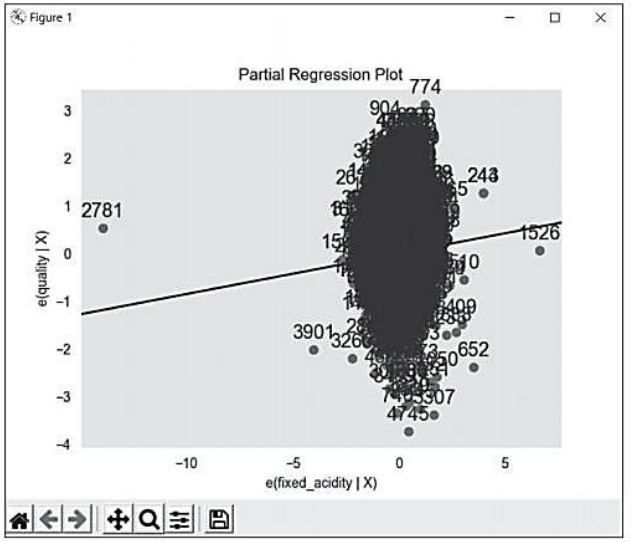
- 회귀분석 모델로 새로운 샘플의 품질 등급 예측하기
  - 샘플 데이터 만들기
  - 샘플로 ‘quality’ 변수 예측하기

데이터 시각화

- 와인 유형에 따른 품질 등급 히스토그램 그리기
  - seaborn 라이브러리 패키지 사용
  - 커널 밀도 추정(kde)을 적용한 히스토그램 그리기
  - x축: quality
  - y축: 확률 밀도 함수값



- 부분 회귀 플롯으로 시각화하기
  - 독립 변수가 2개 이상인 경우에는 부분 회귀 플롯을 사용하여 하나의 독립 변수가 종속 변수에 미치는 영향력을 시각화 함으로써 결과를 분석할 수 있음
  - 부분 회귀 계산을 위해 statsmodels.api를 로드
  - fixed_acidity가 종속 변수 quality에 미치는 영향력을 시각화



피마 인디언 당뇨병 여부 데이터

데이터 분석하기

- **피마 인디언 데이터 분석하기**
  - 비만은 유전일까?
  - 아니면 식습관 조절에 실패한 자신의 탓일까?
  - 비만이 유전 및 환경, 모두의 탓이라는 것을 증명하는 좋은 사례가 바로 미국 남서부에 살고 있는 피마 인디언의 사례
  - 피마 인디언은 1950년대까지만 해도 비만인 사람이 단 1명도 없는 민족이었음
  - 지금은 전체 부족의 60%가 당뇨, 80%가 비만으로 고통받고 있음
  - 이는 생존하기 위해 영양분을 체내에 저장하는 뛰어난 능력을 물려받은 인디언들이 미국의 기름진 패스트푸드 문화를 만나면서 벌어진 일

데이터 분석하기

- **피마 인디언 데이터 분석하기**
  - 모두 768명의 인디언으로부터 여덟 개의 정보와 한 개의 클래스를 추출한 데이터임을 알 수 있음

속성 / 클래스

	정보 1	정보 2	정보 3	…	정보 8	당뇨병 여부
1번째 인디언	6	148	72	…	50	1
2번째 인디언	1	85	66	…	31	0
3번째 인디언	8	183	64	…	32	1
…	…	…	…	…	…	…
768번째 인디언	1	93	70	…	23	0

샘플

- 샘플 수: 768
- 속성: 8
  - 정보 1(pregnant): 과거 임신 횟수
  - 정보 2(plasma): 포도당 부하 검사 2시간 후 공복 혈당 농도(mm Hg)
  - 정보 3(pressure): 확장기 혈압(mm Hg)
  - 정보 4(thickness): 삼두근 피부 주름 두께(mm)
  - 정보 5(insulin): 혈청 인슐린(2-hour, mu U/ml)
  - 정보 6(BMI): 체질량 지수(BMI, weight in kg/(height in m)^2)
  - 정보 7(pedigree): 당뇨병 가족력
  - 정보 8(age): 나이
- 클래스: 당뇨(1), 당뇨 아님(0)

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - 판다스(pandas)와 시본(seaborn) 라이브러리가 필요함

```
# 필요한 라이브러리를 불러옵니다.
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 깃허브에 준비된 데이터를 가져옵니다.
!git clone https://github.com/taehojo/data.git

# 피마 인디언 당뇨병 데이터셋을 불러옵니다.
df = pd.read_csv('./data/pima-indians-diabetes3.csv')
```

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - 판다스 라이브러리의 read_csv() 함수로 csv 파일을 불러와 df라는 이름의 데이터 프레임으로 저장
  - csv 파일에는 데이터를 설명하는 한 줄이 파일 맨 처음에 나옴, 이를 **헤더**(header)라 함

```
df.head(5)
```

  - 정상과 당뇨 환자가 각각 몇 명씩인지 조사해 보자
  - 불러온 데이터 프레임의 특정 칼럼을 불러오려면 df["칼럼명"]이라고 입력하면 됨
  - value_counts() 함수를 이용하면 각 컬럼의 값이 몇 개씩 있는지 알려 줌

```
df["diabetes"].value_counts()
```

  - 정보별 특징을 좀 더 자세히 알고 싶으면 describe() 함수를 이용

```
df.describe()
```

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - 정보별 샘플 수(count), 평균(mean), 표준편차(std), 최솟값(min), 백분위 수로 25%, 50%, 75%에 해당하는 값 그리고 최댓값(max)이 정리되어 보임

	pregnant	plasma	pressure	thickness	insulin	bmi	pedigree	age	diabetes
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - 각 항목이 어느 정도의 상관관계를 가지고 있는지 알고 싶다면 다음과 같이 입력

```
df.corr()
```

	pregnant	plasma	pressure	thickness	insulin	bmi	pedigree	age	diabetes
pregnant	1.000000	0.129459	0.141282	−0.081672	−0.073535	0.017683	−0.033523	0.544341	0.221898
plasma	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
pressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
thickness	−0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	−0.113970	0.074752
insulin	−0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	−0.042163	0.130548
bmi	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
pedigree	−0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
age	0.544341	0.263514	0.239528	−0.113970	−0.042163	0.036242	0.033561	1.000000	0.238356
diabetes	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - 상관관계를 그래프로 표현
  - 맷플롯립(matplotlib)은 파이썬에서 그래프를 그릴 때 가장 많이 사용되는 라이브러리
  - 이를 기반으로 조금 더 정교한 그래프를 그리게 해 주는 시본(seaborn) 라이브러리까지 사용해서 각 정보 간 상관관계를 가시화
  - 그래프의 색상과 크기를 정함

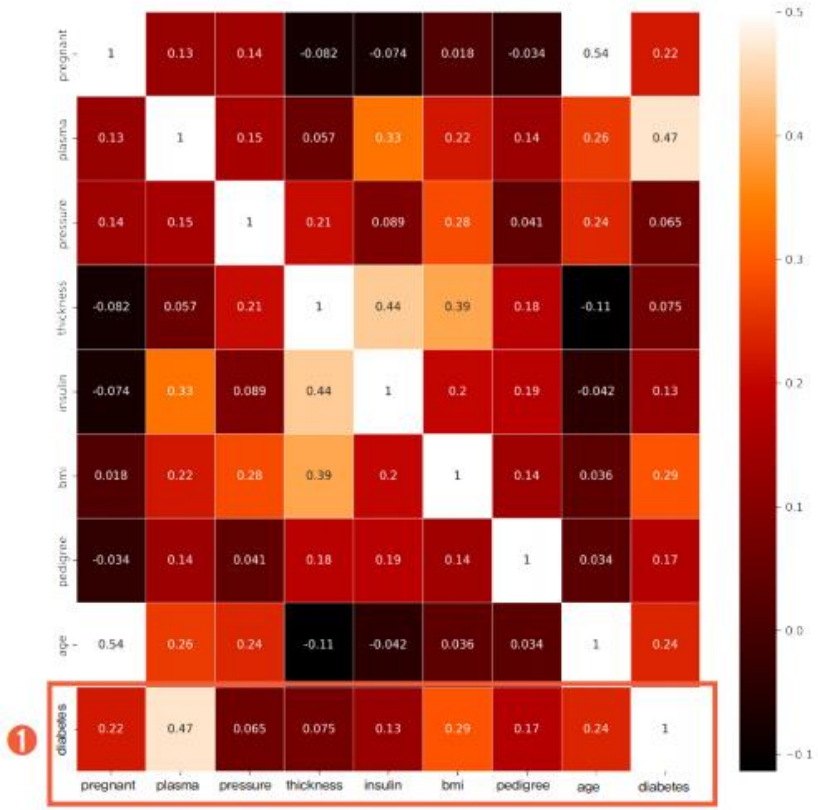
```
colormap = plt.cm.gist_heat    # 그래프의 색상 구성을 정합니다.
plt.figure(figsize=(12,12))    # 그래프의 크기를 정합니다.
```

  - 시본 라이브러리 중 각 항목 간 상관관계를 나타내는 heatmap() 표시
  - heatmap() : 두 항목씩 짝을 지은 후 각각 어떤 패턴으로 변화하는지 관찰
  - 두 항목이 전혀 다른 패턴으로 변화하면 0을,
    서로 비슷한 패턴으로 변할수록 1에 가까운 값을 출력

```
sns.heatmap(df.corr(), linewidths=0.1, vmax=0.5, cmap=colormap,
linecolor='white', annot=True)
plt.show()
```

데이터 분석하기

- **판다스를 활용한 데이터 조사**
  - vmax는 색상의 밝기를 조절하는 인자
  - cmap은 미리 정해진 맷플롯립 색상의 설정 값을 불러옴
    - 색상 설정 값은 https://matplotlib.org/users/colormaps.html에서 확인할 수 있음
  - 가장 눈여겨보아야 할 부분은 당뇨병 발병 여부를 가리키는
    ❶ diabetes 항목
  - diabetes 항목을 보면 pregnant부터 age까지 상관도가 숫자로 표시되어 있고, 숫자가 높을수록 밝은 색상으로 채워져 있음



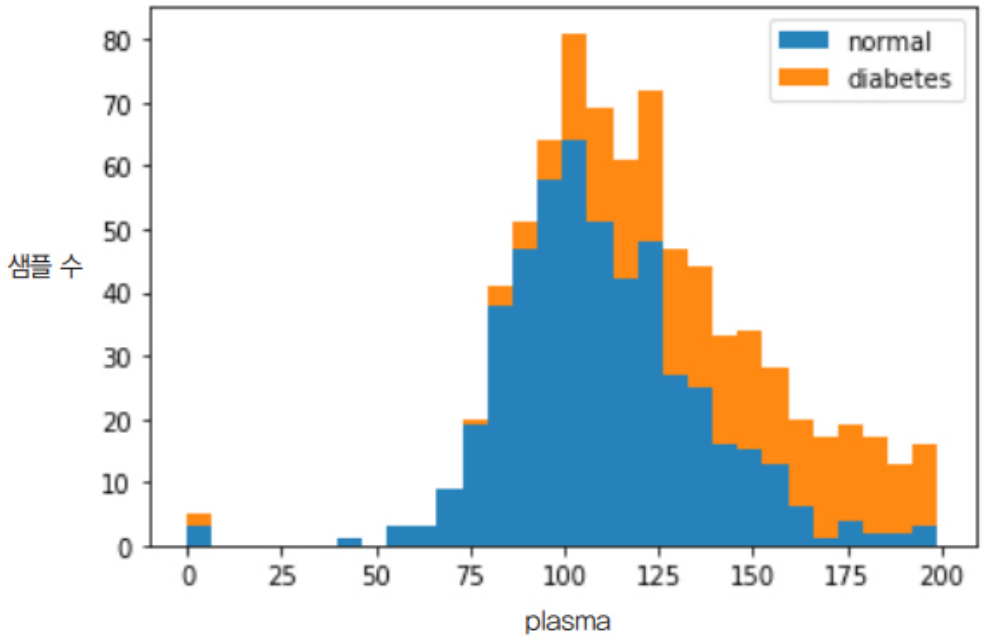
데이터 분석하기

- **중요한 데이터 추출하기**
  - plasma 항목(공복 혈당 농도)과 BMI(체질량 지수)가 우리가 예측하고자 하는 diabetes 항목과 상관관계가 높다는 것을 알 수 있음
  - 즉, 이 항목들이 예측 모델을 만드는 데 중요한 역할을 할 것으로 기대할 수 있음
  - 이 두 항목만 따로 떼어 내어 당뇨의 발병 여부와 어떤 관계가 있는지 확인
  - plasma를 기준으로 각각 정상과 당뇨 여부가 어떻게 분포되는지 확인
  - 히스토그램을 그려 주는 맷플롯립 라이브러리의 hist()를 이용

```
plt.hist(x=[df.plasma[df.diabetes==0], df.plasma[df.diabetes==1]], bins=30,
histtype='barstacked', label=['normal','diabetes'])
plt.legend()
```

데이터 분석하기

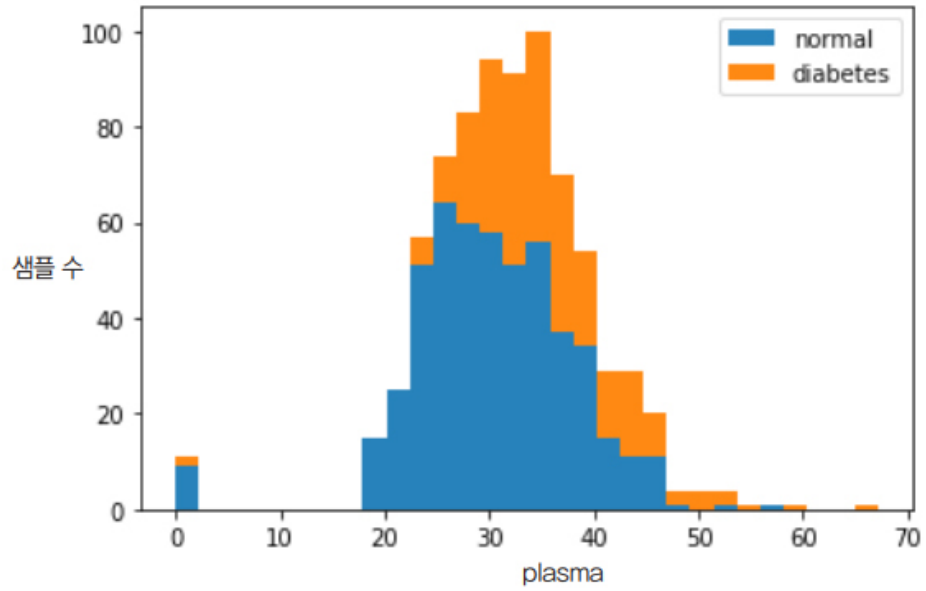
- **중요한 데이터 추출하기**
  - 가져오게 될 칼럼을 hist() 함수 안에 x축으로 지정
  - 여기서 df 안의 plasma 칼럼 중 diabetes 값이 0인 것과 1인 것을 구분해 불러오게 했음
  - bins는 x축을 몇 개의 막대로 쪼개어 보여 줄 것인지 정하는 변수
  - barstacked 옵션은 여러 데이터가 쌓여 있는 형태의 막대바를 생성하는 옵션
  - 불러온 데이터의 이름을 각각 normal(정상)과 diabetes(당뇨)로 정함
  - 실행시키면 그래프가 형성



데이터 분석하기

- **중요한 데이터 추출하기**
  - plasma 수치가 높아질수록 당뇨인 경우가 많음을 알 수 있음
  - BMI를 기준으로 각각 정상과 당뇨가 어느 정도 비율로 분포하는지 확인

```
plt.hist(x=[df.bmi[df.diabetes==0], df.bmi[df.diabetes==1]], bins=30,
histtype='barstacked', label=['normal','diabetes'])
plt.legend()
```



데이터 분석하기

- **중요한 데이터 추출하기**
  - BMI가 높아질 경우 당뇨의 발병률도 함께 증가하는 추세를 볼 수 있음
  - 이렇게 결과에 미치는 영향이 큰 항목을 발견하는 것이 데이터 전처리 과정 중 하나
  - 이 밖에도 데이터에 빠진 값이 있다면 평균이나 중앙값으로 대치하거나, 흐름에서 크게 벗어나는 이상치를 제거하는 과정 등이 데이터 전처리에 포함될 수 있음
  - SVM이나 랜덤 포레스트처럼 일반적인 머신 러닝에서는 데이터 전처리 과정이 성능 향상에 중요한 역할을 함

제로콜라와 기온속성 간의 관계

제로콜라와 기온속성 간의 관계

- (1) 문제정의

- 자판기 매니저는 자판기에 제로콜라를 얼마나 채워야 하는지 알 수 없다. 어느 날은 제로콜라가 품절되기도 하고 제로콜라만 남기도 한다. 이를 위해 제로콜라를 예측하 는 프로그램을 만들고자 한다. 이때 제로콜라를 예측할 수 있도록 하는 주요 특성이 무엇인지 찾아보자.

제로콜라와 기온속성 간의 관계

- (2) 데이터속성 설명

- 데이터셋은 총 5개의 속성으로 구성되어 있고 각 속성은 다음과 같은 특징을 가지고 있음

속성 명	속성 정보
date	날짜
mean	평균기온
min	최저기온
max	최고기온
zero-coke	제로콜라 판매량

제로콜라와 기온속성 간의 관계

- (3) 원본 데이터 관찰하기

- 데이터 정보와 내용 확인

데이터 정보 확인하기

```
1  import pandas as pd
2  df=pd.read_csv('/content/기온과제로콜라.csv')
3  df.info()
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   date        366 non-null    object
 1   mean        366 non-null    float64
 2   min         365 non-null    float64
 3   max         366 non-null    float64
 4   zero_coke   366 non-null    int64
dtypes: float64(3), int64(1), object(1)
memory usage: 14.4+ KB
```

제로콜라와 기온속성 간의 관계

- (2) 데이터속성 설명

데이터 내용 확인하기

```
1  df.head()
```

실행 결과

	date	mean	min	max	zero_coke
0	2021-09-24	21.8	16.7	26.9	58
1	2021-09-25	21.2	18.7	23.1	54
2	2021-09-26	22.6	18.4	27.3	77
3	2021-09-27	21.6	18.7	25.0	100
4	2021-09-28	21.5	19.6	24.4	77

제로콜라와 기온속성 간의 관계

- (3) 결측치 확인

결측치 확인

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

실행 결과

```
date          0
mean          0
min           1
max           0
zero_coke     0
dtype: int64
```

최저기온에 결측치가 1개 있음

제로콜라와 기온속성 간의 관계

- (4) 요약 통계값 확인

결측치 확인

```
1  df.describe()
```

실행 결과

	mean	min	max	zero_coke
count	366.000000	365.000000	366.000000	366.000000
mean	13.556011	9.522466	18.236612	41.893443
std	10.749479	11.146055	10.566594	16.789824
min	-12.100000	-15.500000	-7.300000	6.000000
25%	5.025000	0.700000	9.300000	30.000000
50%	14.700000	10.400000	20.150000	44.000000
75%	22.875000	18.900000	27.275000	54.000000
max	30.900000	27.400000	36.100000	100.000000

최소 6개에서 최대 100개까지 판매

제로콜라와 기온속성 간의 관계

- (5) 전처리하기

결측치 제거 및 확인

```
1  df=df.dropna(axis=0)
2  df.isnull().sum()
```

실행 결과

```
date        0
mean        0
min         0
max         0
zero_coke   0
dtype: int64
```

최저기온에 대한 결측치가 1개 있고, 해당 데이터 행을 삭제하더라도 전체 데이터에 큰 문제가 없을 것으로 판단되기에 결측치가 발생한 행을 삭제

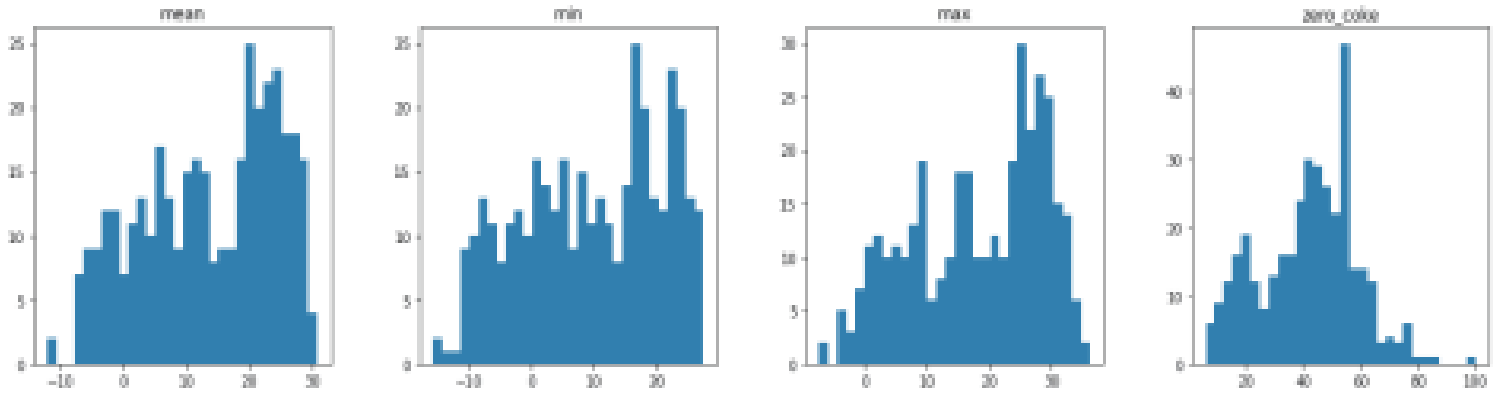
제로콜라와 기온속성 간의 관계

- (6) 시각화로 표현

- 히스토그램으로 속성별 분포 확인

```
1  import matplotlib.pyplot as plt
2
3  df.hist(figsize=(20, 10), grid=False, layout=(2, 4), bins = 30)
```

실행 결과



제로콜라와 기온속성 간의 관계

- (6) 시각화로 표현

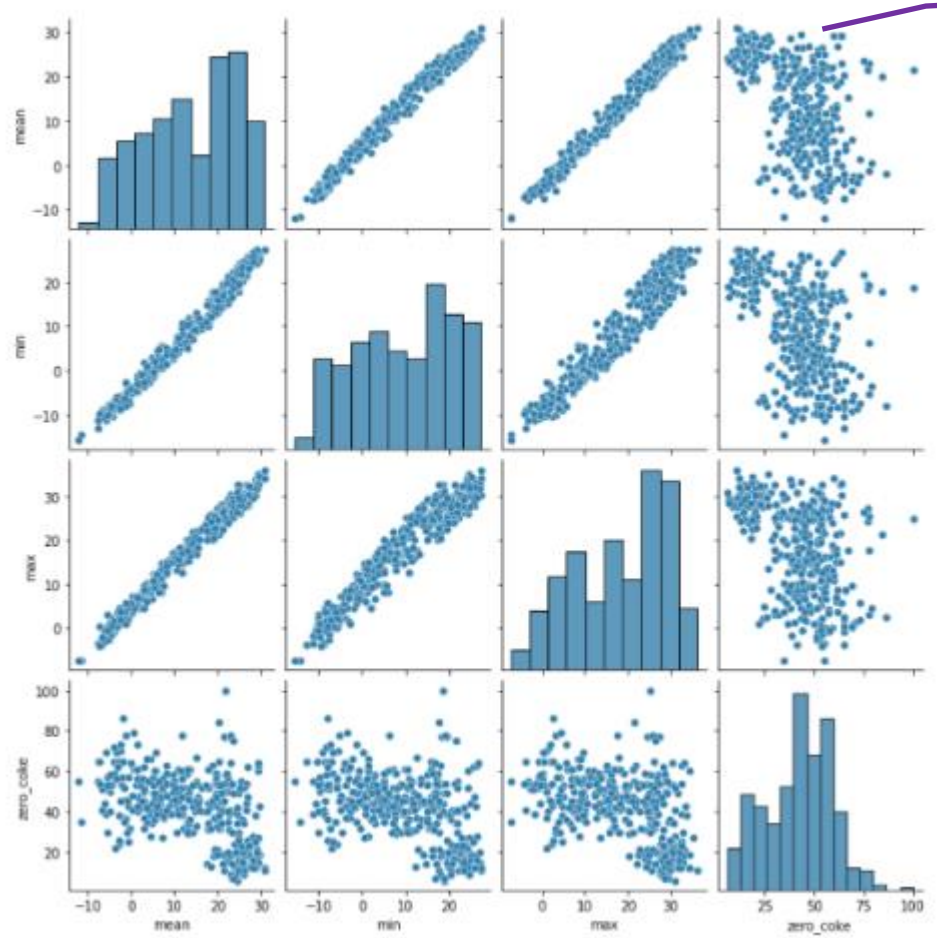
- pairplot()으로 각 속성의 관계 확인

각 속성 간의 관계

```
1  import numpy as np
2  # 수치형 데이터만 추출하여 num_features에 저장
3  num_features = df.select_dtypes(include=[np.number]).columns
   sns.pairplot(df[num_features])
```

제로콜라와 기온속성 간의 관계

실행 결과



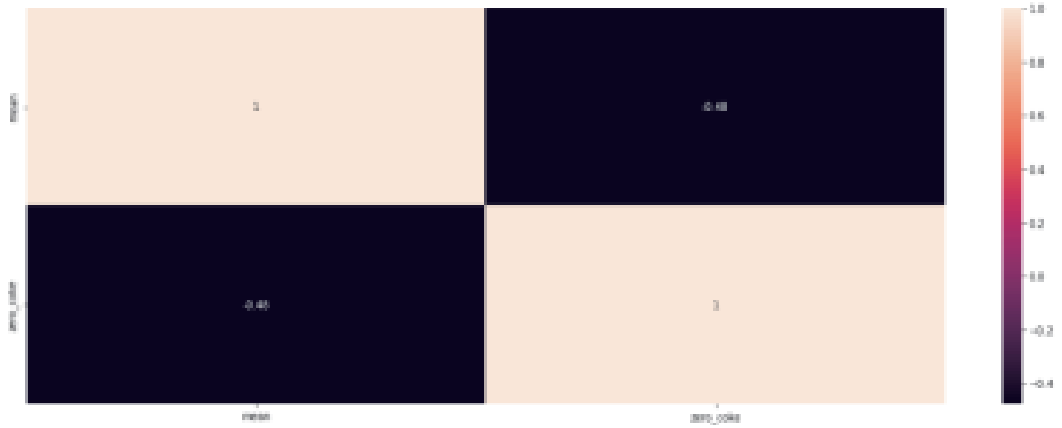
제로콜라는 최저기온, 최고기온, 평균기온 간의 음의 상관관계를 가지고 있는 분포를 보임

제로콜라와 기온속성 간의 관계

- (7) 상관관계 확인하기

```
1  plt.figure(figsize=(20, 7))
2  sns.heatmap(df[['mean', 'zero_coke']].corr(), annot=True)
```

실행 결과



- 제로콜라와 평균기온 간에는 음의 상관관계가 있는 것을 알 수 있다
- 겨울철에는 제로콜라의 판매량이 더 많 아지고 여름철에는 제로콜라의 판매량이 적어진다는 의미로도 해석

전복 측정 데이터

데이터탐색과 전처리 실습

- (1) 문제정의

- 전복의 나이를 확인하기 위해 껍질을 원뿔 모양으로 자르고, 염색하고, 현미경으로 고리의 수를 세어 확인해야 한다. 이렇게 진행하려면 너무나도 많은 시간이 필요하다. 더 쉽게 나이를 예측하기 위해 전복의 특성이 담긴 데이터를 이용하고자 한다. 전복의 특성이 담긴 데이터들을 머신러닝에 사용할 수 있도록 정리해보자.

데이터탐색과 전처리 실습

- (2) 데이터 속성 설명

- 전복 데이터셋은 총 9개의 속성이고 각 속성은 다음과 같은 특징을 가지고 있음

속성 명	속성 정보	속성 명	속성 정보
Rings	나이테 개수	Whole weight	전체 무게
Sex	성별(수컷(m), 암컷(f), 유아(i))	Shucked weight	순살 무게
Length	길이	Viscera weight	내장 무게
Diameter	직경	Shell weight	껍질 무게
height	두께		

데이터탐색과 전처리 실습

- (3) 원본 데이터 관찰하기

- 데이터 정보 확인하기

```
1  import pandas as pd
2
3  df=pd.read_csv('/content/abalone.csv', encoding='utf-8')
4  df.info()
5
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

수치만 있는 항목은 int64, 문 자열을 포함한 항목은 object

데이터탐색과 전처리 실습

- (3) 원본 데이터 관찰하기

- 결측치 확인

칼럼별 결손값 유무 확인

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

실행 결과

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

null 값이 존재하지 않는 것을 알 수 있음

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 수치형 데이터 통계

사분위수에서 50%에 해당하는 제2사분위수는 중앙값과 동일한 의미

통계량 확인

```
1  df.describe()
```

실행 결과

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 최빈값 확인

최빈값 확인

```
1  df.mode()
```

실행 결과

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	9.0
1	NaN	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

각 속성은 정규분포는 아니지만, 정규성에 가깝다는 것을 확인할 수 있다(중간값=평 균=최빈값)

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 이상치 제거 : 통계량에서 두께가 0인 것보다 큰 값으로만 데이터를 추출하여 이상치 제거

이상치 제거

```
1  df=df[df.Height>0]
2  df.Height.min()
```

실행 결과

```
0.01
```

이상치가 잘 제거되었는지 확인한 결과, 두께(Height)의 최솟값이 0.01인 것을 알 수 있음

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 나이테 수(Rings)에 따른 전복 개수 확인

나이테 수에 따른 전복의 개수

```
1  df.groupby('Rings').count()
```

실행 결과

Rings	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	15	15	15	15	15	15	15	15
4	57	57	57	57	57	57	57	57
5	115	115	115	115	115	115	115	115
6	258	258	258	258	258	258	258	258
7	391	391	391	391	391	391	391	391
8	567	567	567	567	567	567	567	567
9	689	689	689	689	689	689	689	689

가장 많이 차지하는 나이테 수는 9

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 나이테 수(Ring)에 따른 순살의 평균 무게 확인

```
df.groupby('Rings')['Shucked weight'].mean()
```

나이테 수가 29인 경우 가장 무겁고,
나이테 수가 1인 경우 가장 가벼움

Rings	
1	0.001000
2	0.004500
3	0.011767
4	0.024719
5	0.061696
6	0.123413
7	0.182657
8	0.293927
9	0.387938
10	0.447217
11	0.503977
12	0.472781
13	0.434638
14	0.427190
15	0.402471
16	0.421716
17	0.467052
18	0.446833
19	0.440625
20	0.458115
21	0.447464
22	0.405000
23	0.399444
24	0.600250
25	0.426500
26	0.384000
27	0.539000
29	0.705500

데이터탐색과 전처리 실습

- (4) 요약통계값 확인

- 나이테 수에 따른 전체 무게(Whole weight)와 껍질 무게(Shell weight) 확인

```
1  df.groupby('Rings')['Shucked weight', 'Whole weight', 'Shell
   weight'].mean()
```

- 껍질 무게가 가장 많이 나가는 나이테 수는 27
- 나이테 수가 작을수록 껍질 무게도 적어짐
- 전체 무게 또한 나이테 수가 작을수록 0에 가깝고, 나이테 수가 크면 1을 넘음

데이터탐색과 전처리 실습

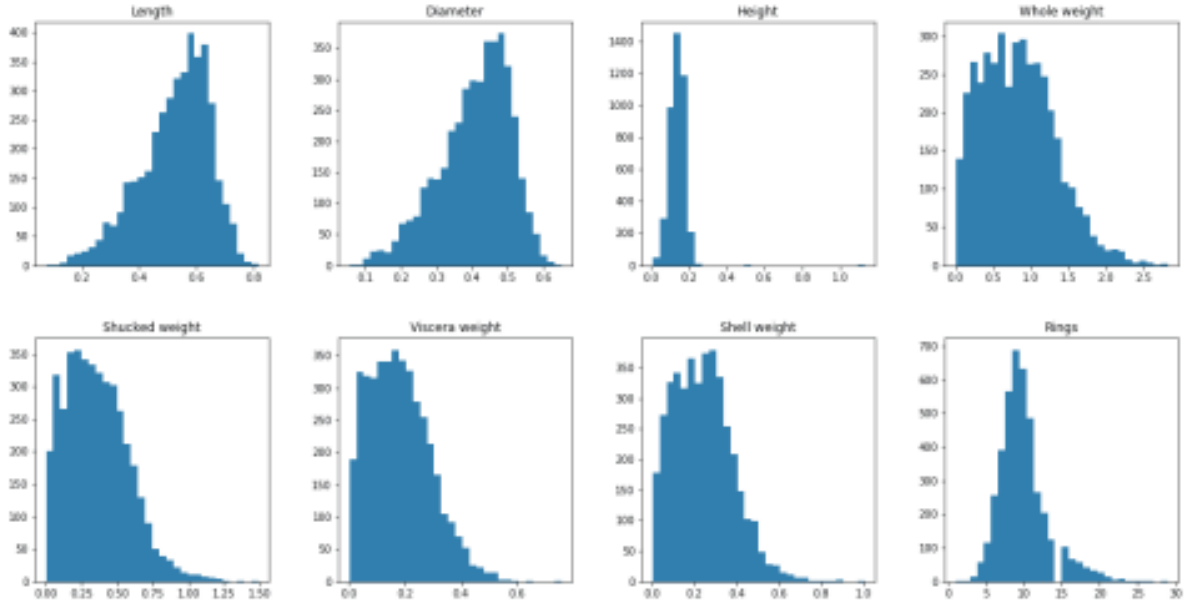
- (5) 시각화로 표현

- 히스토그램으로 속성별 분포 확인

속성별 데이터 분포

```
1  # 2행 4열로 각 속성의 히스토그램 출력
2  df.hist(figsize=(20, 10), grid=False, layout=(2, 4), bins = 30)
```

실행 결과



데이터탐색과 전처리 실습

- (5) 시각화로 표현

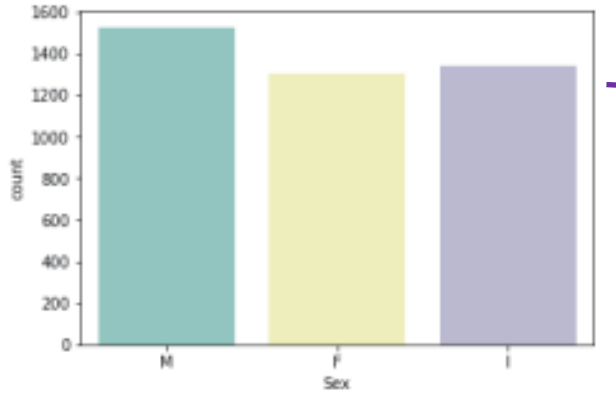
- countplot()으로 성별에 따른 데이터 분포 확인

성별의 분포

```
1  import seaborn as sns
2  sns.countplot(x = 'Sex', data = df, palette = "Set3")
```

실행 결과

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4l5df8d3l0>
```



수컷의 수가 가장 많고 암컷과
유아의 개수가 비슷 하게 구성

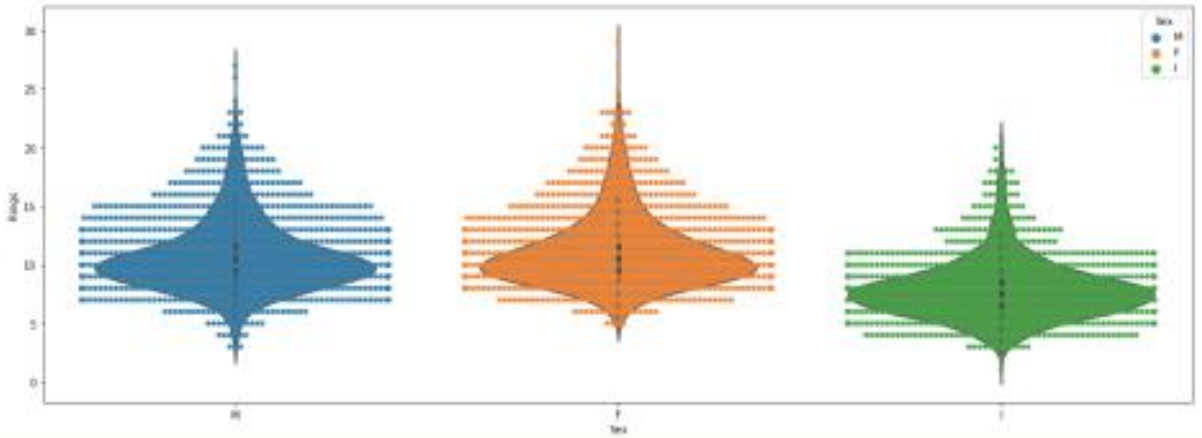
데이터탐색과 전처리 실습

- (5) 시각화로 표현

- swarmplot()과 violinplot()으로 성별에 따른 나이 분포 확인

```
1  plt.figure(figsize = (20, 7))
2  sns.swarmplot(x = 'Sex', y = 'Rings', data = df, hue = 'Sex')
3  sns.violinplot(x = 'Sex', y = 'Rings', data = df)
```

실행 결과



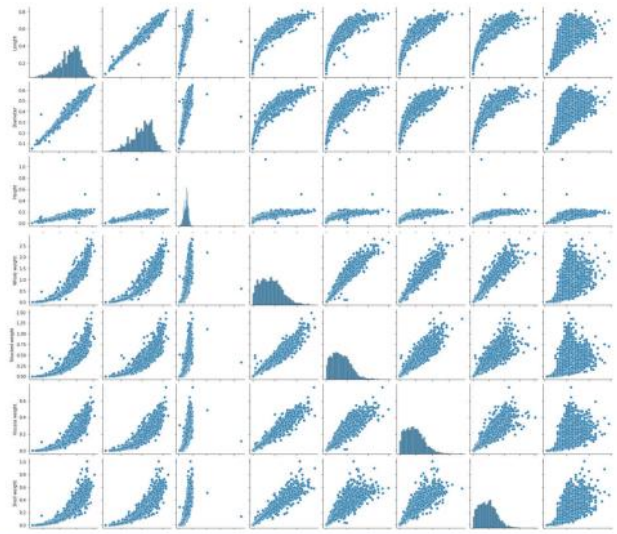
데이터탐색과 전처리 실습

- (5) 시각화로 표현
- pairplot()으로 각 속성의 변화 확인

각 속성 간의 변화

```
1  import numpy as np
2  # 숫자로 구성된 데이터 타입의 칼럼들만 추출하여 num_features에 저장
3  num_features = df.select_dtypes(include=[np.number]).columns
4  sns.pairplot(df[num_features])
```

실행 결과



길이(length)와 직경(diameter)은 강력한 상관관계가 있고, 전체 무게(whole weight)는 순살 무게, 내장 무게, 껍질 무게와 높은 상관관계가 있음

데이터탐색과 전처리 실습

- (5) 시각화로 표현

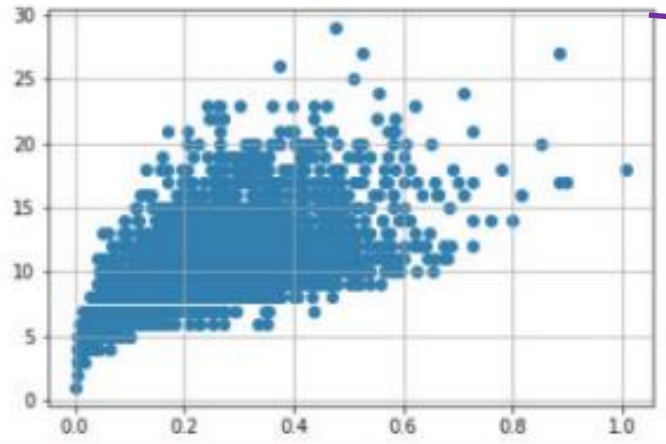
- scatterplot()으로 이상치 확인하고 제거

① 껍질 무게 이상치

껍질 무게의 이상치 확인

```
1  var = 'Shell weight'
2  plt.scatter(x = df[var], y = df['Rings'],)
3  plt.grid(True)
```

실행 결과



- 껍질 무게의 이상치는 0.6보다 크고 나이테 수는 25보다 작은 부분
- 껍질 무게가 0.8보다 크고 나이테 수가 25보다 큰 부분도 이상치

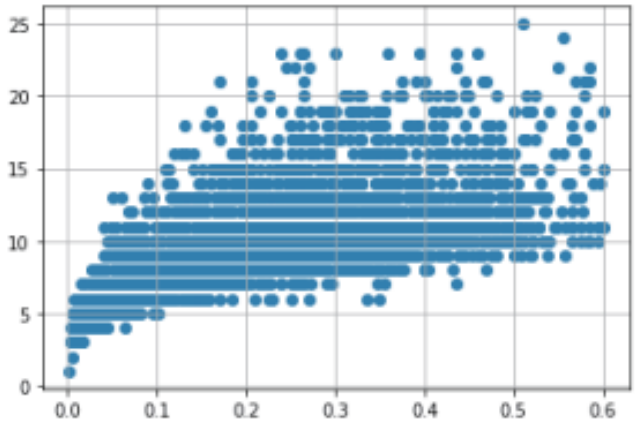
데이터탐색과 전처리 실습

- (5) 시각화로 표현

껍질 무게의 이상치 제거

```
1 df.drop(df[(df['Shell weight'] > 0.6) & (df['Rings'] < 25)].index,
  inplace=True)
2 df.drop(df[(df['Shell weight'] < 0.8) & (df['Rings'] > 25)].index,
  inplace=True)
3 df.drop(df[(df['Shell weight'] > 0.8) & (df['Rings'] > 25)].index,
  inplace=True)
4 var = 'Shell weight'
5 plt.scatter(x = df[var], y = df['Rings'],)
6 plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

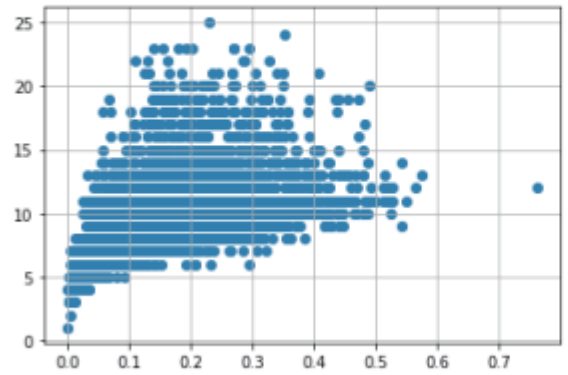
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

② 내장 무게 이상치

내장 무게의 이상치 확인

```
1  var = 'Viscera weight'
2  plt.scatter(x = df[var], y = df['Rings'],)
3  plt.grid(True)
```

실행 결과



- 내장 무게의 이상치는 0.5보다 크고 나이테 수는 20보다 작은 부분
- 내 장 무게가 0.5보다 작고 나이테 수가 25보다 크거나 같은 데이터도 이상치

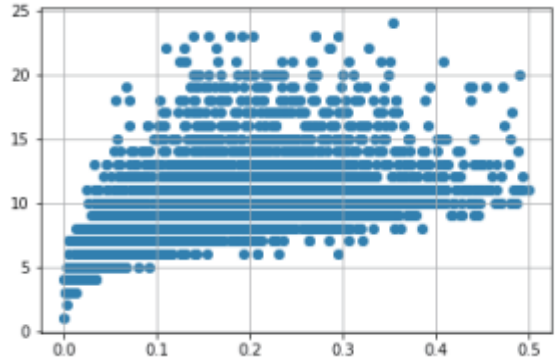
데이터탐색과 전처리 실습

- (5) 시각화로 표현

내장 무게의 이상치 확인

```
1 df.drop(df[(df['Viscera weight'] > 0.5) & (df['Rings'] < 20)].
  index, inplace=True)
2 df.drop(df[(df['Viscera weight'] < 0.5) & (df['Rings'] >= 25)].
  index, inplace=True)
3 var = 'Viscera weight'
4 plt.scatter(x = data[var], y = data['age'],)
5 plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

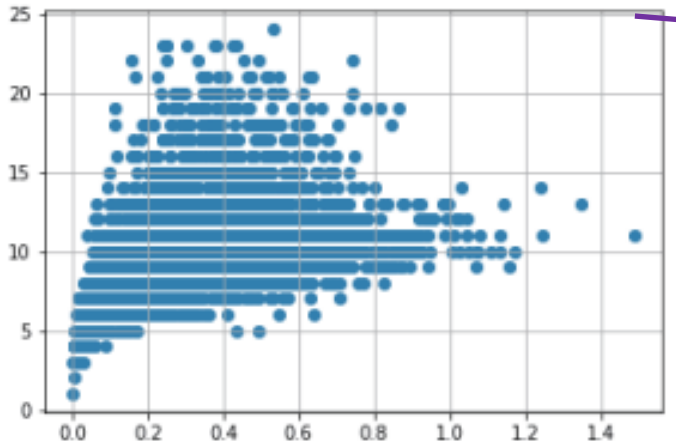
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

③ 순살 무게 이상치

순살 무게의 이상치 확인

```
1  var = 'Shucked weight'
2  plt.scatter(x = df[var], y = df['Rings'])
3  plt.grid(True)
```

실행 결과



- 순살 무게의 이상치는 1 이상이고 나이테 수는 20보다 작은 부분
- 순살 무게가 1보다 작고 나이테 수가 20보다 큰 데이터도 이상치

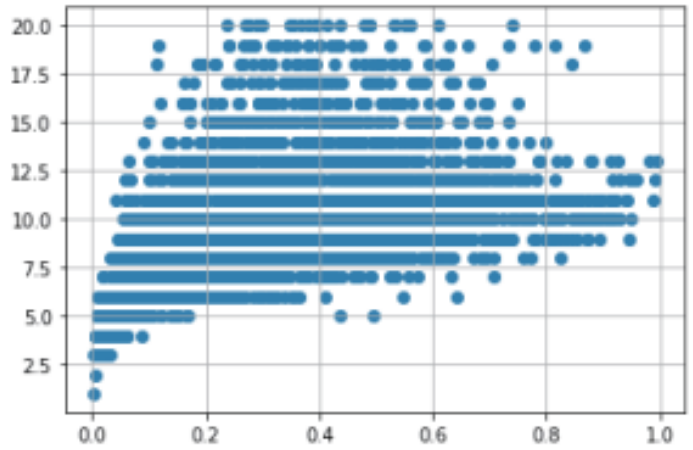
데이터탐색과 전처리 실습

- (5) 시각화로 표현

순살 무게의 이상치 확인

```
1 df.drop(df[(df['Shucked weight'] >= 1) & (df['Rings'] < 20)].
  index, inplace=True)
2 df.drop(df[(df['Shucked weight'] < 1) & (df['Rings'] > 20)].index,
  inplace=True)
3 var = 'Shucked weight'
4 plt.scatter(x = df[var], y = df['Rings'])
5 plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

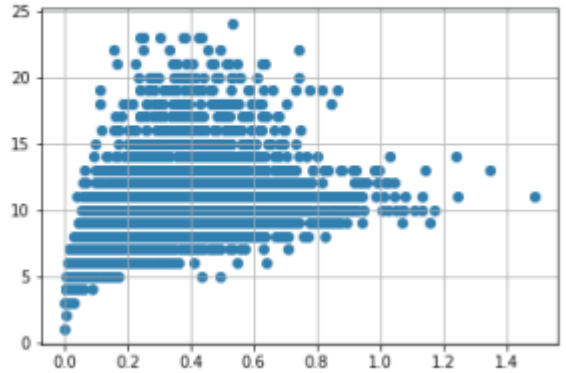
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

④ 전체 무게 이상치

전체 무게의 이상치 확인

```
1  var = 'Shucked weight'
2  plt.scatter(x = df[var], y = df['Rings'])
3  plt.grid(True)
```

실행 결과



- 전체 무게의 이상치는 2.0보다 크고 나이테 수는 20 이상인 부분

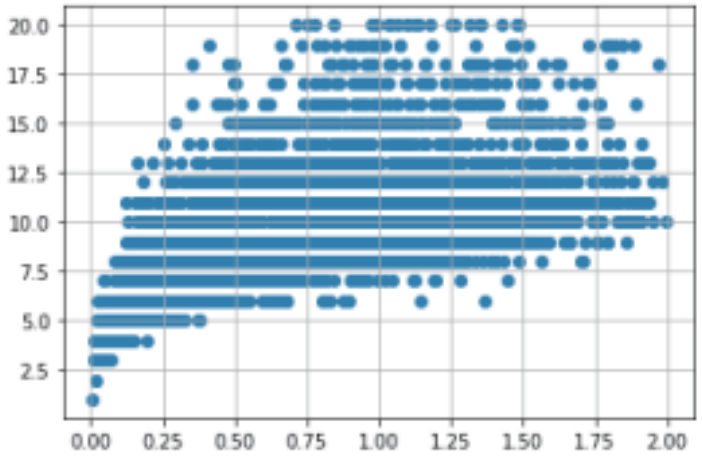
데이터탐색과 전처리 실습

- (5) 시각화로 표현

전체 무게의 이상치 제거와 확인

```
1  df.drop(df[(df['Whole weight'] > 2.0) & (df['Rings'] <= 20)].index,
   inplace=True)
2  var = 'Whole weight'
3  plt.scatter(x = df[var], y = df['Rings'])
4  plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

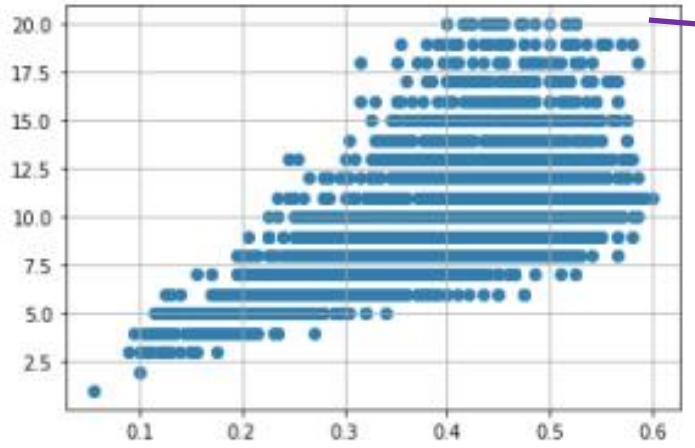
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

⑤ 직경의 이상치

직경의 이상치 확인

```
1  var = 'Diameter'
2  plt.scatter(x = df[var], y = df['Rings'])
3  plt.grid(True)
```

실행 결과



- 직경의 이상치는 0.1 이하이고 나이테 수는 2.5 미만인 부분

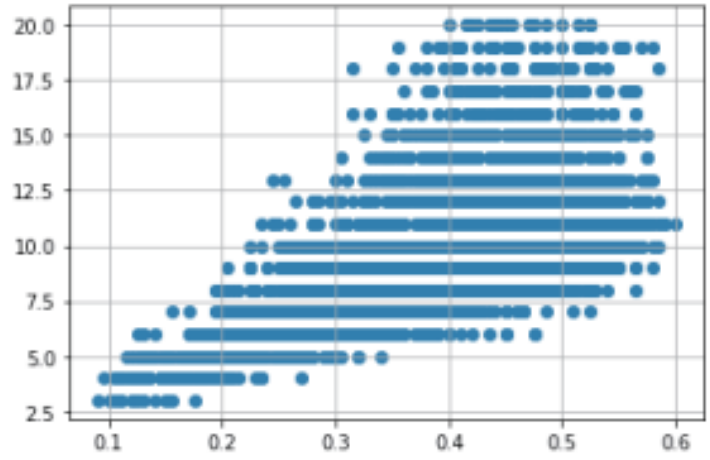
데이터탐색과 전처리 실습

- (5) 시각화로 표현

직경의 이상치 제거와 확인

```
1  df.drop(df[(df['Diameter'] <= 0.1) & (df['Rings'] < 2.5)].index,
   inplace=True)
2  var = 'Diameter'
3  plt.scatter(x = df[var], y = df['Rings'])
4  plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

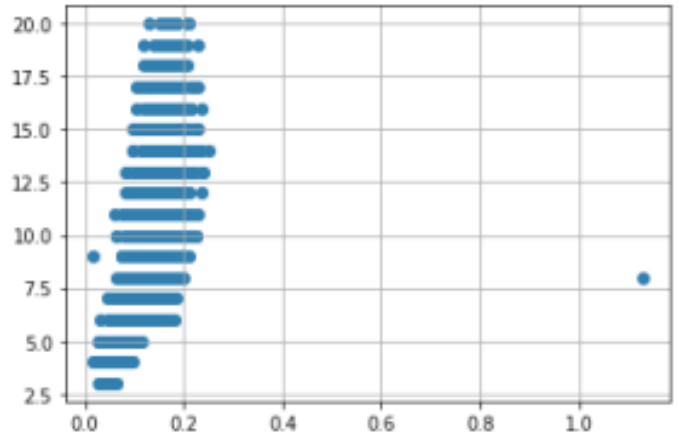
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

⑥ 높이의 이상치

높이의 이상치 확인

```
1  var = 'Height'
2  plt.scatter(x = df[var], y = df['Rings'])
3  plt.grid(True)
```

실행 결과



- 높이의 이상치는 0.4보다 크고 나이테 수는 10 미만인 부분

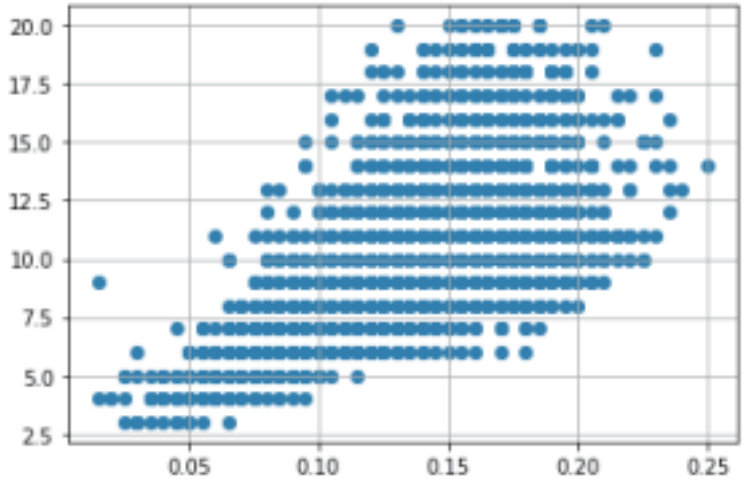
데이터탐색과 전처리 실습

- (5) 시각화로 표현

높이의 이상치 제거와 확인

```
1  df.drop(df[(df['Height'] > 0.4) & (df['Rings'] < 15)].index,
   inplace=True)
2  var = 'Height'
3  plt.scatter(x = df[var], y =df['Rings'])
4  plt.grid(True)
```

실행 결과



데이터탐색과 전처리 실습

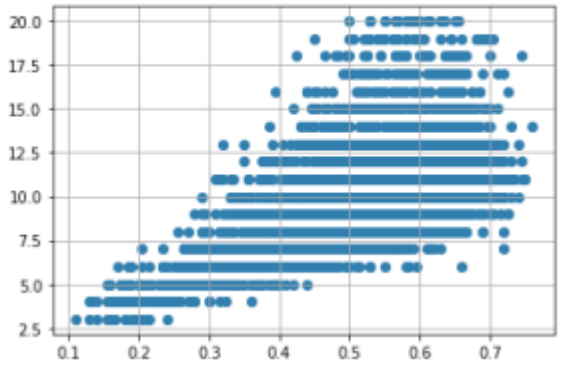
- (5) 시각화로 표현
- scatterplot()으로 이상치 확인하고 제거

⑦ 길이의 이상치

길이의 이상치 확인

```
1  var = 'Length'
2  plt.scatter(x = df[var], y = df['Rings'])
3  plt.grid(True)
```

실행 결과



- 길이 속성에서는 이상치가 없음

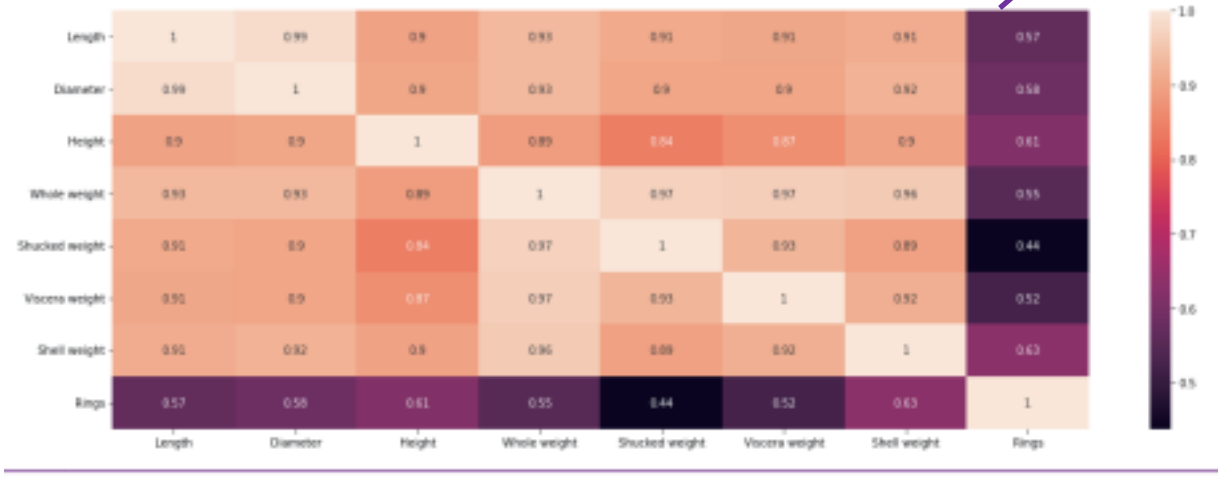
데이터탐색과 전처리 실습

- (5) 시각화로 표현
- heatmap()으로 전체 속성 간의 상관관계 확인

속성 간의 상관관계 확인

```
1  plt.figure(figsize=(20,7))
2  sns.heatmap(df[num_features].corr(), annot=True)
```

실행 결과



- 나이테 수와 길이, 직경, 높이, 전체 무게, 껍질 무게, 내장 무게는 높은 상관관계
- 순살 무게는 상관관계가 어느 정도 있음

연도별 날씨 데이터 회귀분석

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

[준비 파일: (2010-2020) weather.xlsx]

(1) 데이터셋 정보

상단 메뉴에서 [기후통계분석] → [통계분석]을 선택한 후 기온분석 페이지에서 원하는 지역(서울)과 기간(2010.1-2020.12)을 다음과 같이 설정한 후 날씨 데이터를 다운로드한다.

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

(2) 데이터필드의 이해

변수명	데이터형
날짜	날짜형
지점	정수형
평균기온(℃)	실수형
최저기온(℃)	실수형
최고기온(℃)	실수형

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

(3) 데이터 확인과 가공하기

1) 데이터 읽어 와서 확인하기

'날씨' 파일 읽어오기

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import seaborn as sns
5  seoul = pd.read_excel('/content/drive/MyDrive/(2010-2020) weather.xlsx')
6  seoul.head()
```

실행 결과

	날짜	지점	평균기온(℃)	최저기온(℃)	최고기온(℃)
0	2010-01-01	108	-7.6	-12.7	-3.6
1	2010-01-02	108	-3.6	-7.4	0.2
2	2010-01-03	108	-6.8	-10.5	-3.2
3	2010-01-04	108	-5.9	-8.0	-3.4
4	2010-01-05	108	-9.9	-12.3	-7.0

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

1) 데이터 읽어 와서 확인하기

데이터의 전체적인 구조 출력

```
1  seoul.info()
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4018 entries, 0 to 4017
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   날짜          4018 non-null   datetime64[ns]
 1   지점          4018 non-null   int64
 2   평균기온(℃)   4018 non-null   float64
 3   최저기온(℃)   4018 non-null   float64
 4   최고기온(℃)   4017 non-null   float64
dtypes: datetime64[ns](1), float64(3), int64(1)
memory usage: 157.1 KB
```

요약 통계량 확인하기

```
1  seoul.describe()
```

실행 결과

	지점	평균기온(℃)	최저기온(℃)	최고기온(℃)
count	4018.0	4018.000000	4018.000000	4017.000000
mean	108.0	12.965207	8.991015	17.699627
std	0.0	10.803691	10.932678	10.976719
min	108.0	-14.800000	-18.000000	-10.700000
25%	108.0	3.700000	-0.300000	8.200000
50%	108.0	14.250000	9.800000	19.600000
75%	108.0	22.800000	18.900000	27.400000
max	108.0	33.700000	30.300000	39.600000

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

2) 칼럼 삭제하기

지점 명 칼럼 삭제하기

```
1  seoul.drop('지점',axis=1,inplace=True)
2  seoul.head()
```

실행 결과

	날짜	평균기온(℃)	최저기온(℃)	최고기온(℃)
0	2010-01-01	-7.6	-12.7	-3.6
1	2010-01-02	-3.6	-7.4	0.2
2	2010-01-03	-6.8	-10.5	-3.2
3	2010-01-04	-5.9	-8.0	-3.4
4	2010-01-05	-9.9	-12.3	-7.0

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

3) 칼럼명 변경하기

칼럼명 변경하기

```
1  seoul.columns=['날짜','평균기온','최저기온','최고기온']
2  seoul.columns
```

실행 결과

```
Index(['날짜', '평균기온', '최저기온', '최고기온'], dtype='object')
```

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

4) 결측 데이터 확인 및 처리

결측 데이터 개수 확인

```
1  seoul.isnull().sum()
```

실행 결과

```
날짜      0
평균기온    0
최저기온    0
최고기온    1
dtype: int64
```

누락된 데이터 행 삭제하기

```
1  seoul.dropna(subset=['최고기온'],axis=0,inplace=True)
2  seoul.isnull().sum()
```

실행 결과

```
날짜      0
평균기온    0
최저기온    0
최고기온    0
dtype: int64
```

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

5) 칼럼 생성하기

칼럼 생성하기

```
1  seoul['년도']=seoul['날짜'].dt.year
2  seoul.head()
```

실행 결과

	날짜	평균기온	최저기온	최고기온	년도
0	2010-01-01	-7.6	-12.7	-3.6	2010
1	2010-01-02	-3.6	-7.4	0.2	2010
2	2010-01-03	-6.8	-10.5	-3.2	2010
3	2010-01-04	-5.9	-8.0	-3.4	2010
4	2010-01-05	-9.9	-12.3	-7.0	2010

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

6) 데이터 필터링

조건에 맞는 데이터 추출

```
1  conditions=(seoul['날짜'].dt.month==8) & (seoul['날짜'].dt.day==15)
2  conditions
3  seoul0815=seoul[conditions]
4  seoul0815
```

실행 결과

	날짜	평균기온	최저기온	최고기온	년도
226	2010-08-15	26.6	24.6	30.2	2010
591	2011-08-15	24.5	22.9	26.9	2011
957	2012-08-15	23.7	22.4	27.1	2012
1322	2013-08-15	28.7	25.8	32.4	2013
1687	2014-08-15	24.9	20.9	29.6	2014
2052	2015-08-15	27.1	23.1	30.8	2015
2418	2016-08-15	29.1	25.8	34.0	2016
2783	2017-08-15	21.9	20.8	24.0	2017
3148	2018-08-15	31.7	28.3	38.0	2018
3513	2019-08-15	25.9	23.9	28.6	2019
3879	2020-08-15	26.1	25.0	27.0	2020

날씨 데이터 예측하기

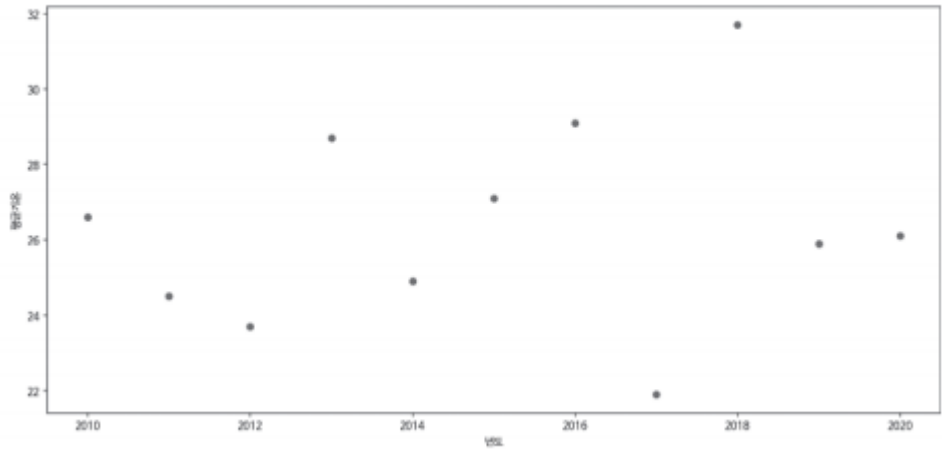
- [회귀분석] 날씨 데이터 분석하기

7) 데이터 시각화

산점도 그래프 그리기

```
1  fig = plt.figure(figsize=(15,7))
2  X = seoul0815[['년도']]
3  Y = seoul0815['평균기온']
4  plt.xlabel('년도')
5  plt.ylabel('평균기온')
6  plt.scatter(X,Y)
7  plt.show()
```

실행 결과



날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

(4) 회귀분석

과거 10년 날씨 데이터를 분석하고 회귀분석의 개념을 이해하기 위한 목적으로 현재의 날씨를 예측하는 분석 코딩을 실시하였다. 날씨 데이터 변수 간의 인과관계를 알아보고 특정 날짜의 기온을 예측해보고자 한다.

1) 회귀분석에 필요한 라이브러리와 모듈을 불러온다.

필요한 모듈 import

```
1  from sklearn.linear_model import LinearRegression
```

날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

2) 단순선형회귀분석

2022년 08월15일 기온 예측하기

```
1  model = LinearRegression()
2  X = seoul0815[['년도']]
3  Y = seoul0815['평균기온']
4  model.fit(X, Y)
5  result = model.predict([['2022']]) # [27.50818182]
6  print(result)
```

날씨 데이터 예측하기

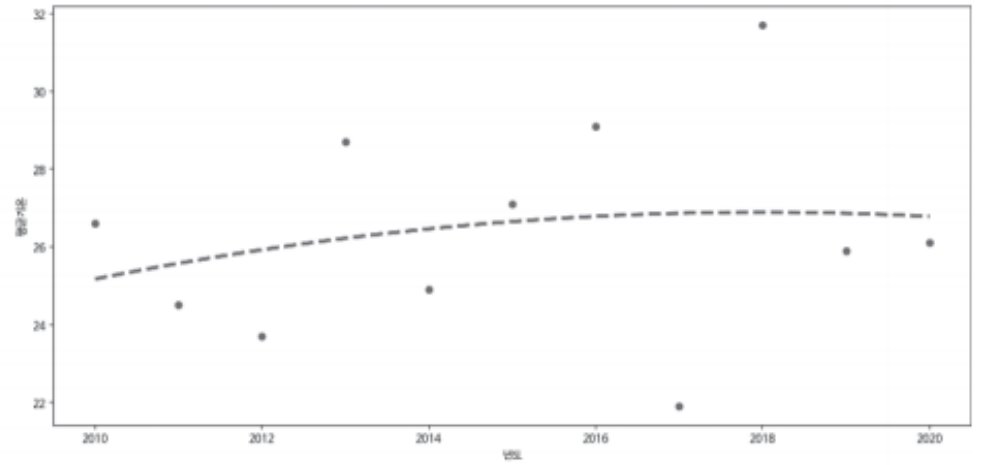
- [회귀분석] 날씨 데이터 분석하기

2) 단순선형회귀분석

08월15일 평균 기온의 산점도, 회귀선 출력하기

```
1  x=seoul0815['년도']
2  y=seoul0815['평균기온']
3  fp1 = np.polyfit(x, y,2)
4  f1 = np.poly1d(fp1)
5  fx = np.linspace(2010, 2020)
6  plt.figure(figsize=(15,7))
7  plt.scatter(x,y)
8  plt.plot(fx, f1(fx), ls='dashed', lw=3, color='g')
9  plt.xlabel('년도')
10 plt.ylabel('평균기온')
11 plt.show()
```

실행 결과



날씨 데이터 예측하기

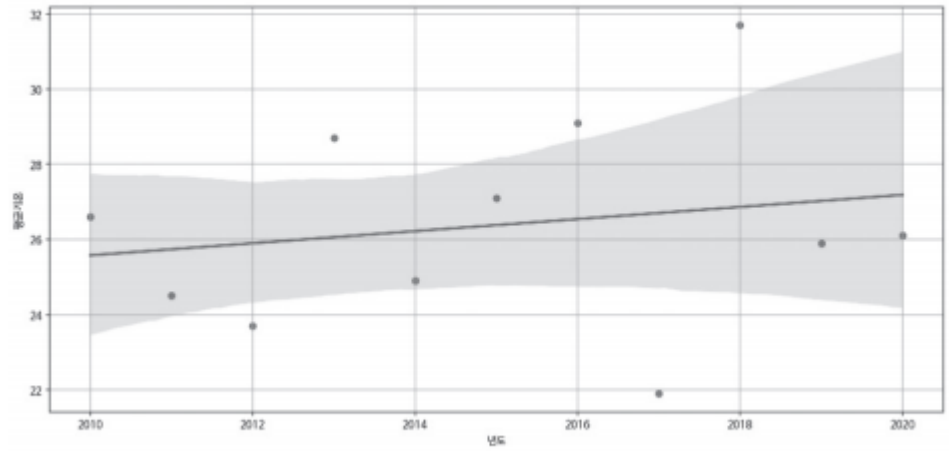
- [회귀분석] 날씨 데이터 분석하기

2) 단순선형회귀분석

시본 모듈로 추세선 넣기

```
1  fig = plt.figure(figsize=(15, 7))
2  sns.regplot(x='년도', y='평균기온', data=seoul0815)
3  plt.grid()
4  plt.show()
```

실행 결과



날씨 데이터 예측하기

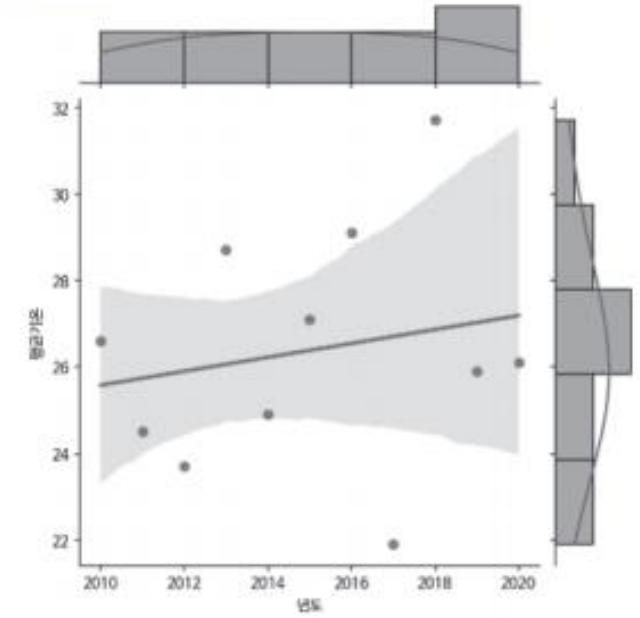
- [회귀분석] 날씨 데이터 분석하기

2) 단순선형회귀분석

산점도와 히스토그램 동시에 그리기

```
1  plt.figure(figsize=(15, 7))
2  sns.jointplot(x='년도', y='평균기온', data=seoul0815, kind='reg')
3  plt.show()
```

실행 결과



날씨 데이터 예측하기

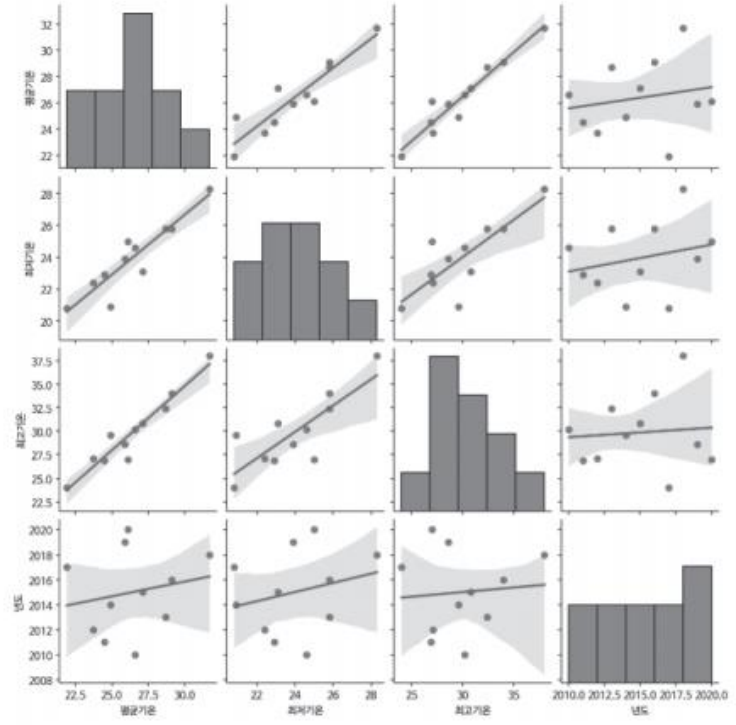
- [회귀분석] 날씨 데이터 분석하기

2) 단순선형회귀분석

산점도와 히스토그램 동시에 그리기

```
1  sns.pairplot(seoul0815,kind='reg')
2  plt.show()
```

실행 결과



날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

3) 다중선형회귀분석

다중선형회귀는 여러 개의 특성을 이용해 종속변수를 예측하기 때문에 일반선형회귀보다 더 좋은 성능을 기대할 수 있다.

2022년 08월 15일 기온 예측하기

```
1  from sklearn.linear_model import LinearRegression
2  model = LinearRegression()
3
4  X = seoul0815[['년도','최저기온','최고기온']]
5  Y = seoul0815['평균기온']
6
7  model.fit(X, Y)
8  result = model.predict([['2022',24,33]])
9  print(result)                  # [28.08381871] # 평균기온 예측됨
```

실행 결과

```
[28.08381871]
```

날씨 데이터 예측하기

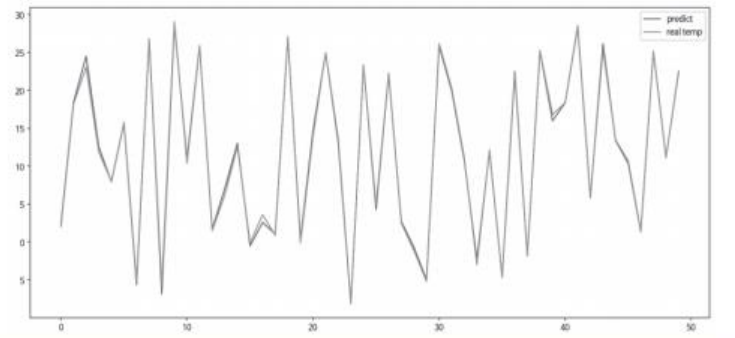
- [회귀분석] 날씨 데이터 분석하기

3) 다중선형회귀분석

다중선형회귀분석

```
1  from sklearn.linear_model import LinearRegression
2  from sklearn.model_selection import train_test_split
3  import seaborn as sns
4  plt.figure(figsize=(15,7))
5
6  # 데이터 준비: 속성(변수) 2가지 선택
7  X = seoul[['년도','최저기온','최고기온']]
8  Y = seoul['평균기온']
9
10 x_train, x_test, y_train, y_test = train_test_split(X,Y, train_
   size = 0.7, test_size = 0.3)
11 model = LinearRegression()
12 model.fit(x_train, y_train)
13
14 plt.plot(model.predict(x_test[:50]), label='predict')
15 plt.plot(y_test[:50].values.reshape(-1, 1), label='real temp')
16 plt.legend()
```

실행 결과



날씨 데이터 예측하기

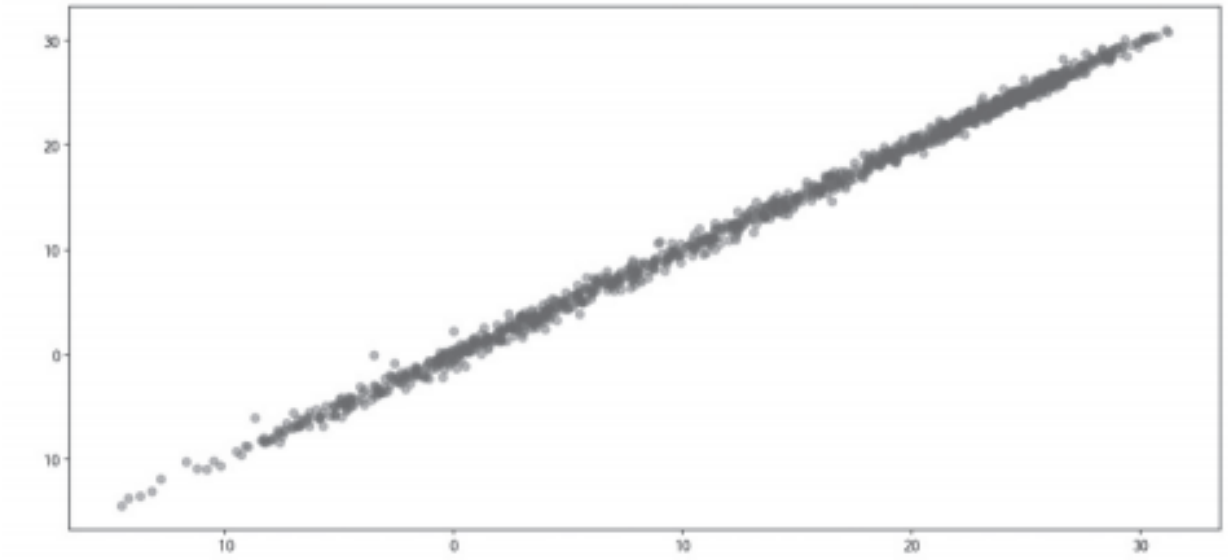
- [회귀분석] 날씨 데이터 분석하기

3) 다중선형회귀분석

다중선형회귀분석: 산점도 그래프 그리기

```
1  plt.figure(figsize=(15,7))
2  y_predict = model.predict(x_test)
3
4  plt.scatter(y_test, y_predict, alpha = 0.4)
5  plt.show()
```

실행 결과



날씨 데이터 예측하기

- [회귀분석] 날씨 데이터 분석하기

3) 다중선형회귀분석

다중선형회귀모델의 성능 측정

```
1  print(model.score(x_train,y_train))
```

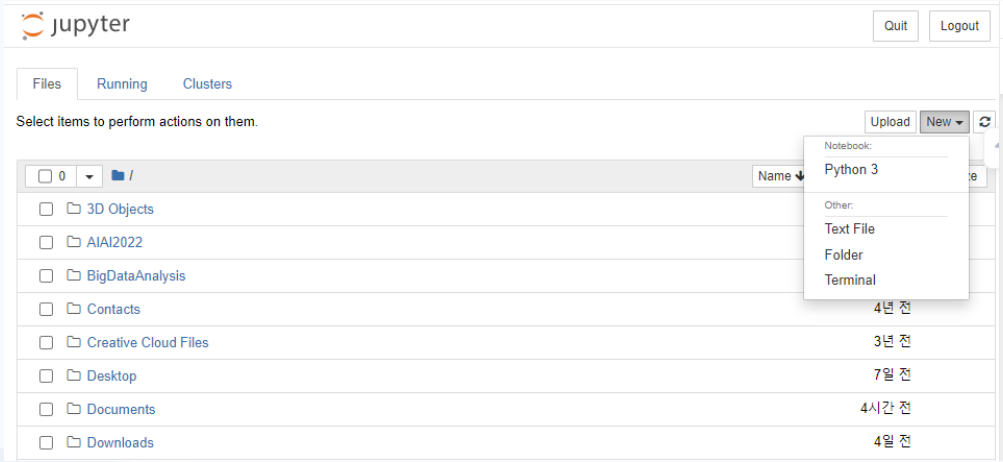
실행 결과

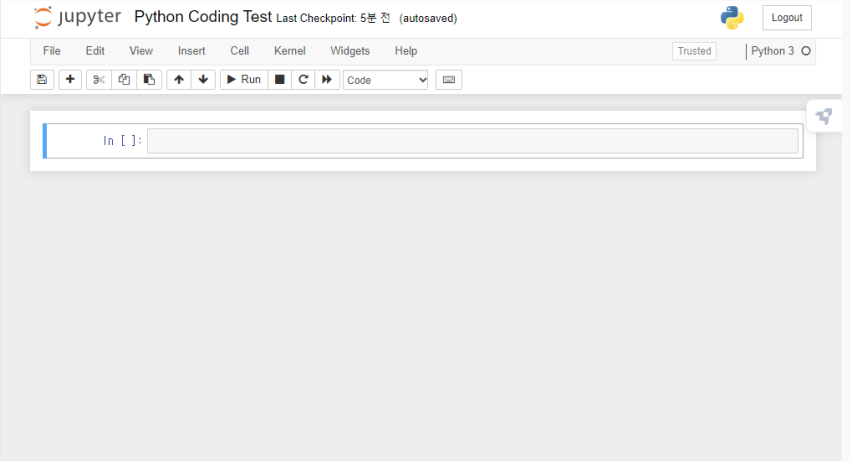
```
0.9976977104781903
```

우리가 모델링한 다중선형회귀모델은 약 0.99의 결정계수를 가지며, 이는 x 변수들이 y 변수에 미치는 영향이 99%로, x 변수들이 평균 기온값 변동의 99%를 설명할 수 있다는 뜻이다.

학습활동: Python 코딩 실습

Python 코드 소스 파일 작성 실습





Thank you!

See you next time.

담당교수 : 유 현 주

comjoo@uok.ac.kr