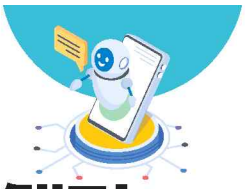


03 기계학습을 이용한 문제 해결

✚ **학습 목표** •기계학습을 활용하여 해결할 수 있는 문제와 그렇지 않은 문제를 구분할 수 있다.
•기계학습을 이용해 사회 문제를 해결할 수 있다.

✚ **학습 요소** •문제 해결 과정, 회귀, 분류, 군집

생각 깨우기 다음 그림을 보고 질문에 답해 보자.

Q 우리 주변의 문제를 인공지능을 이용해 어떻게 해결할 수 있을까?

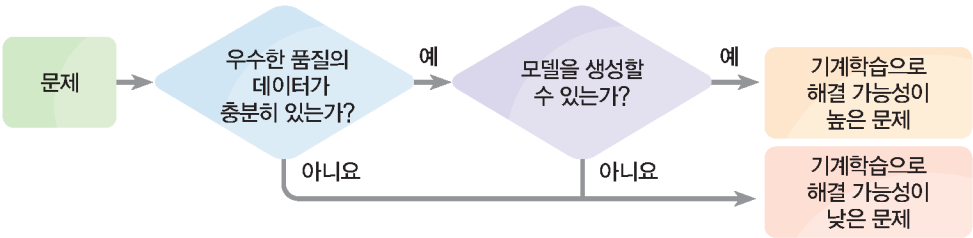
1 | 기계학습과 해결 가능한 문제

01 기계학습으로 해결할 수 있는 문제란?

기계학습을 이용해 문제를 해결하기 위해서는 문제 상태를 정확히 파악하고 정의할 수 있어야 한다. 주어진 상황의 현재 상태와 목표 상태가 무엇인지, 활용해야 하는 핵심 요소는 무엇인지 등을 정확히 분석해야 문제 해결 방향을 명확히 설정할 수 있으므로 문제 정의 단계는 매우 중요하다.

정의한 문제가 기계학습을 통해 해결할 수 있는 문제인지 또한 확인해야 한다. 기계학습으로 해결할 수 있는 문제인지 여부는 훈련 데이터의 양과 질, 활용하는 프로그램의 성능, 앞으로 기술 발전 양상에 따라 달라질 수 있으므로 명확하게 정의하기는 어렵다. 그러나 충분한 양과 우수한 품질을 갖춘 데이터가 있는지, 이를 활용해 적합한 모델을 생성할 수 있는지 여부에 따라 기계학습으로 해결할 가능성이 높은 문제와 낮은 문제를 구분할 수 있다.

스팸 메일을 일일이 확인하고 삭제하는 불편을 해결하기 위해 자동으로 스팸 메일을 분류하는 프로그램을 만든다고 가정해 보자. 스팸 메일에 자주 들어가는 단어, 스팸 메일을 자주 보내는 메일 주소 등의 데이터를 충분히 확보할 수 있고, 사용되는 단어들의 빈도와 특징을 분석하고 학습해 모델을 생성할 수 있으므로 기계학습으로 해결할 수 있는 가능성이 높다.



△ 기계학습으로 해결할 수 있는 문제인지 판단하기

Tip 분류 모델을 이용한 '스팸 메일 판별하기'

1단계: 스팸 메일을 일일이 확인하고 삭제하는 것이 불편함.

2단계: 다량의 스팸 메일과 일반 메일의 주소, 내용, 핵심 단어, 문장 등의 속성을 포함한 데이터를 수집하고 전처리함.

3단계: 스팸 메일인지 아닌지 구분해야 하므로 분류 모델을 선정함.

4단계: 학습을 통해 분류 모델이 생성되면 메일이 입력되었을 때 자동으로 기준에 맞추어 스팸 메일인지 아닌지 구분함.

5단계: 스팸 메일을 잘 구분하는지 평가 및 보완함.

02 기계학습 모델을 이용한 문제 해결 과정

기계학습을 이용해 문제를 해결하는 과정은 다음과 같다.

우선, 해결하고자 하는 문제가 무엇인지 문제 상황을 분석하고 현재 상태와 목표 상태를 정리한 다음 문제를 정의한다. 이후 인공지능 학습을 위해 목적에 맞는 데이터를 탐색하고 수집해야 한다. 또한 수집한 데이터가 학습에 적합한 데이터인지 검토하고, 데이터의 오류, 공백을 수정하거나 데이터 형태를 학습에 적합한 형태로 재구조화하는 데이터 전처리를 한다. 이후 문제를 해결하려는 목표에 맞는 기계학습의 유형과 적용할 알고리즘을 선정한다.

전처리가 완료된 데이터와 선정한 기계학습 알고리즘을 통해 인공지능이 패턴이나 규칙을 발견해 기계학습 모델을 생성하면, 학습된 기계학습 모델에 문제 해결을 위해 사용할 데이터를 적용해 결과(정보) 값을 출력한다. 이후에는 사용한 기계학습 모델이 적합했는지, 원하는 결과가 나오는지 그 성능을 평가하고 보완한다.

해 보기 1 글로벌 공공선을 위한 인공지능 알아보기

근래 지속가능발전목표(SDGs)에 대한 중요성이 대두되면서, 이를 달성하기 위해 인공지능을 활용하는 사례가 증가하고 있다. 인터넷에서 인공지능이 SDGs를 이용해 글로벌 공공선을 실천하는 방법에 관한 영상을 찾아보고 다음 활동을 해 보자.

*공공선: 개인을 위한 선(善)에 대비(對比)되는 말로서, 개인을 포함하는 사회 전체를 위한 선을 뜻한다.

지속가능발전목표(SDGs)

"2030년까지 세계의 빈곤을 종식하고 지구상의 모든 사람이 평등하고 평화로운 사회에서 살 수 있도록 함"을 목표로 하는 미래를 위한 17가지 목표로 구성된 유엔(UN)의 행동 계획을 의미한다.

참고 영상

https://youtu.be/ND7pvShNdIg
https://www.youtube.com/watch?v=0M5qKKLmC-I

지속가능한 발전을 달성하기 위해 국제 사회에서 채택한 17개 지속가능발전목표(SDGs: Sustainable Development Goals)를 따라 우리나라도 국제 사회의 책임있는 일원으로서 국제 사회의 공동 목표 달성에 기여하고, 한국 사회에 처한 여러 문제들을 해결하기 위해 한국형 지속가능발전목표, 즉 K-SDGs를 수립하였다.

1 영상을 통해 알게 된 인공지능 활용 사례에 대해 정리하고, 친구에게 설명해 보자.

영상을 통해 알게 된 AI	관련 목표	
	인공지능의 기능	
	기대 효과	

2 인공지능의 힘을 사회적 이익을 위해 어떻게 활용할 수 있을지 주제를 선정해 아이디어를 제안해 보자.

2 | 기계학습으로 문제 해결하기

일상생활의 다양한 문제 상황을 어떻게 기계학습으로 해결할 수 있는지 알아보자. 회귀, 분류, 군집 모델을 이용해 여러 문제를 해결해 보자.

문제 1 자동차의 CO_2 배출량 예측하기

문제 상황 **자동차는 많은 양의 배기가스를 배출한다. 배기가스의 주성분인 이산화 탄소(CO_2)는 지구 온난화를 일으키는 온실가스 중 하나다. 자동차의 정보를 이용해 이산화 탄소 배출량을 예상해 볼 수 있을까? 자동차의 연료량, 가스 중량 등의 여러 속성을 통해 이산화 탄소 배출량을 예측해 보자.**

생각해 보기

1. 자동차가 배출하는 이산화 탄소 양을 파악하려면 자동차의 어떤 정보를 알아야 할까?
2. 속성들을 통해 이산화 탄소 배출량을 예상해 보는 것이 기계학습으로 해결 가능한 문제인가? 그렇게 생각한 이유는 무엇인가?
3. 이산화 탄소 배출량을 예측하기 위해 어떤 기계학습 유형을 사용해야 할까?

1단계 문제 정의

문제를 해결하기 위해 기계학습으로 해결할 수 있는 문제로 정의해 보자.

자동차의 연료량과 가스 중량 속성을 이용해 이산화 탄소 배출량을 예측해 보자.

2단계 데이터 탐색 및 전처리

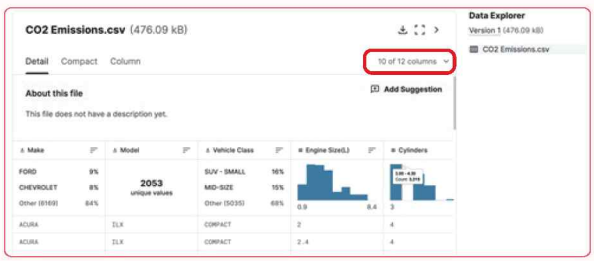
① 데이터 수집하기

필요한 데이터를 받을 수 있는 사이트에서 자동차의 이산화 탄소(CO2) 배출량과 관련된 속성이 들어 있는 데이터를 내려받는다.

Tip 데이터 수집 방법

캐글 사이트(kaggle.com)에서 'CO2 emission'을 검색한다. 'Data Card' 항목의 데이터셋 정보를 살펴보면 필요한 속성을 포함한 데이터셋임을 파악할 수 있다.

붉은색 부분을 클릭해 문제 정의 단계에서 선정한 핵심 속성들만 선택해 조회할 수 있다. 엔진 크기, 도심에서의 연료 소비량, 고속도로에서의 연료 소비량, 이산화 탄소 배출량 등을 체크해 핵심 속성만 담은 데이터셋을 조회하고, 필요한 속성이 모두 있다면 csv 파일을 내려받는다.



출처 https://www.kaggle.com/datasets/bhuviranga/co2-emissions

필요한 데이터를 UCI 사이트에서도 받을 수 있다. UCI는 "University of California, Irvine"의 약자로, 데이터 과학 및 기계 학습 연구를 위한 데이터셋을 제공하는 사이트다. 데이터셋에 대한 설명과 다운로드 링크를 제공한다.

데이터셋(data-set)

컴퓨터에서 사용할 수 있게 저장된 데이터들의 집합체이다.

csv 파일

여러 개의 필드를 쉼표(,)로 구분한 데이터 파일로 스프레드시트 형식으로 쉽게 조작 가능하다.

데이터 파악 시 사용하는 명령어

함수	기능
head()	앞부분 출력
tail()	뒷부분 출력
shape	행, 열 개수 출력
info()	변수 속성 출력
describe()	요약 통계량 출력

속성 설명

- Make: 제조업체명
- Model: 모델명
- Vehicle class: 차량 등급
- Engine Size: 엔진 크기
- Cylinders: 실린더 수(기통 수)
- Transmission: 변속기
- Fuel type: 연료 종류
- Fuel Consumption City: 도시에서의 연료 소비량
- Fuel Consumption Hwy: 고속도로 연료 소비량
- Fuel Consumption Comb (L/100km): 평균 연료 소비량(리터당 주행 거리)
- Fuel Consumption Comb (mpg): 평균 연료 소비량(갤런당 주행 거리)
- CO2 Emissions: 이산화탄소 배출량

- L/100km: 리터당 주행 가능한 거리. 즉, 주행 가능한 거리를 리터로 나누어서 계산하며 값이 낮을수록 연비가 높다.
- mpg: 갤런당 주행 가능한 거리. 즉, 주행 가능한 거리를 갤런으로 나누어서 계산하며 값이 높을수록 연비가 높다.

❷ 데이터 불러오기

컴퓨터에 저장된 데이터를 코랩에 업로드하기 위해 google.colab에서 files라는 패키지를 불러온 후, upload() 함수를 통해 myfile이라는 이름으로 저장한다.

❶ 'CO2_data.csv' 파일을 불러와 코랩에 업로드하고 myfile에 저장한다.

프로그램

```
1 #코랩 환경에서 실습에 사용할 데이터 파일 불러오기
2 from google.colab import files
3 #코랩에 파일 불러와 myfile 변수에 저장하기
4 myfile = files.upload()
```

실행 결과

설명 코드를 실행하면 그림과 같이 '파일 선택' 버튼이 생성된다. 컴퓨터의 csv 파일을 선택해 업로드를 완료하면 '100% done'이라는 말과 함께 파일명이 뜬다.

❷ 데이터 처리에 필요한 판다스(pandas) 라이브러리를 불러와 pd라는 별칭으로 줄여 사용한다. 그 후 업로드한 데이터의 내용을 읽어 df로 저장하고, head()를 이용하여 내용을 확인한다.

프로그램

```
1 # pandas 라이브러리를 불러와 pd라는 별칭 지정하기
2 import pandas as pd
3 # csv 파일의 내용을 읽어와 df라는 변수에 저장하기
4 df = pd.read_csv("CO2 Emissions.csv")
```

❸ df.head( )를 이용해 df에 저장한 데이터 중 상위 5개 데이터를 화면에 나타내 데이터의 내용을 확인한다. head( ) 함수는 상위 일부 행의 데이터를 확인할 수 있는 함수로, ( ) 안에 숫자를 적지 않으면 기본적으로 5개 행의 데이터를 보여준다.

프로그램

```
1 # head() 함수를 이용해 데이터셋의 상위 5줄 확인하기
2 df.head()
```

실행 결과

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)	Fuel Consumption Hwy (L/100 km)	Fuel Consumption Comb (L/100 km)	Fuel Consumption Comb (mpg)	CO2 Emissions(g/km)
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244

설명 csv 파일이 어떤 형태로 구성되어 있는지 확인해 보니, 자동차 제조업체명(Make), 모델명(Model), 엔진 크기(Engine Size) 등의 속성으로 구성된 정형 데이터인 것을 알 수 있다.

❸ 결측치 확인 및 제거

❶ df.info()는 데이터 속성의 자료형과 자료 개수(행의 개수)를 알려 주는 함수이다. 이를 통해 데이터의 정보를 확인할 수 있다.

프로그램

```
#데이터 속성의 자료형과 속성별 데이터 개수 알아보기
df.info()
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7385 entries, 0 to 7384
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Make                            7385 non-null   object
 1   Model                           7385 non-null   object
 2   Vehicle Class                   7385 non-null   object
 3   Engine Size(L)                  7385 non-null   float64
 4   Cylinders                       7385 non-null   int64
 5   Transmission                    7385 non-null   object
 6   Fuel Type                       7385 non-null   object
 7   Fuel Consumption City (L/100 km)  7385 non-null   float64
 8   Fuel Consumption Hwy (L/100 km)   7385 non-null   float64
 9   Fuel Consumption Comb (L/100 km)  7385 non-null   float64
 10  Fuel Consumption Comb (mpg)     7385 non-null   int64
 11  CO2 Emissions(g/km)             7385 non-null   int64
dtypes: float64(4), int64(3), object(5)
memory usage: 692.5+ KB
```

설명 info()의 결과를 보아 7,385개의 데이터로 이루어진 것을 알 수 있으며, 각 속성은 모두 정수형, 실수형 수치값 또는 문자형으로 이루어졌다는 것을 알 수 있다. Dtype에서, int64는 정수형 수치값, float64는 실수형 수치값, object는 그 외 문자형 등의 값을 의미한다. 또한 non-null이라는 것으로 보아 결측치가 없는 데이터라고 할 수 있다.

3단계 기계학습 유형과 알고리즘 선정

적합한 기계학습 유형을 선정해 보고, 그 이유를 친구와 이야기해 보자.

- 적합한 기계학습 유형: 회귀
- 이유: 예 엔진 크기와 연료 소비량과 관련된 속성을 이용해 CO2 배출량이라는 특정 수치값을 예상하고자 하므로 지도학습의 회귀를 이용한다.
- 독립 변수와 종속 변수 선정: 예 Engine Size(엔진 크기)와 Fuel Consumption Comb(L/100km)(평균 연료 소비량)를 독립 변수로 선정하고, CO2 Emissions(이산화 탄소 배출량) 속성을 종속 변수로 선정한다.

사용할 데이터는 많은 속성을 가진 정형 데이터이다. 이 중 실습 목적에 적합한 몇 가지 속성을 선정해 기계학습에 활용해야 정확도 높은 모델을 만들 수 있다. 여러 속성 중에 엔진 크기(Engine Size), 평균 연료 소비량(Fuel Consumption Comb(L/100km))

결측치
데이터가 존재하지 않거나 누락된 값 또는 상태이다.

이상치
일반적인 데이터 패턴에서 벗어나거나 다른 값들과 비교했을 때 두드러지는 독특한 값이다.

Tip
데이터의 정보를 확인하고자 할 때, len()을 통해서도 데이터 개수(행의 개수)를 파악할 수 있다. isna().sum() 함수 또한 속성마다 결측치가 몇 개 있는지를 속성별 결측치 개수의 합을 표시함으로써 알려 준다.

info() 함수 외에도 describe 함수를 통해 데이터 개수, 수치 데이터의 평균, 분산, 최솟값, 최댓값 등의 정보를 파악할 수 있다.

독립 변수와 종속 변수
독립 변수는 다른 변수에 영향을 미치는 변수이며, 종속 변수는 독립 변수에 의해 영향을 받는 변수이다.

선형 회귀 알고리즘
선형 회귀 알고리즘은 데이터의 관계를 이해하고 예측하는 데 사용된다. 이 알고리즘은 데이터가 일직선적인 관계를 가진다고 가정하며, 이를 통해 데이터의 패턴을 설명하고 예측한다. 선형 회귀 알고리즘은 데이터의 특성과 목표 값 사이의 관계를 분석하여 최적의 선을 찾아내는 과정을 거치며, 이렇게 찾아낸 선은 데이터의 특성을 가장 잘 설명하는 선으로 새로운 데이터에 대한 예측을 할 수 있다. 따라서 선형 회귀 알고리즘은 간단한 예측 모델을 만들 때 자주 사용된다.

속성의 값을 이용해 이산화 탄소 배출량(CO2 Emissions) 속성의 값을 구하려고 한다. 이러한 상황에서는 지도학습의 회귀가 적절하다. 그 중에서도 회귀의 대표적인 알고리즘인 선형 회귀 알고리즘을 이용해 기계학습 모델을 만들어 보자.

4단계 기계학습을 통한 모델 생성

학습 모델을 만들기 위해 데이터셋의 여러 속성을 독립 변수와 종속 변수로 나누어 각각을 X와 y에 저장한다. 또한 데이터를 훈련 데이터와 테스트 데이터로 나누어 각각을 X_train, X_test, y_train, y_test로 저장한다.

데이터는 2차원 배열이므로 대문자 X를, 타깃인 종속 변수는 1차원 배열이므로 소문자 y를 사용하는 경우가 많다.

❶ 학습할 속성이 될 독립 변수 X와 타깃(목표) 속성이 될 종속 변수 y를 나누어 저장한다.

독립 변수 X에는 여러 가지 속성들이 포함될 것이므로 '[]' 기호를 2번 사용한다. 즉, [] 안에 속성 이름이 리스트의 형태로 들어가는 것이다.

프로그램

```
# 학습할 속성이 될 X(독립 변수)와 타깃(목표) 속성이 될 y(종속 변수)를 나누어 저장하기
X = df[['Engine Size(L)', 'Fuel Consumption Comb (L/100 km)']]
y = df['CO2 Emissions(g/km)']
```

❷ 훈련 데이터와 테스트 데이터를 나눈다. 기존의 데이터셋을 훈련 데이터와 테스트 데이터로 나눈다. 데이터셋을 X_train, X_test, y_train, y_test라는 이름으로 분할해 저장한다. 이때 훈련 데이터와 테스트 데이터의 비율은 7:3으로 분할하며, random_state는 데이터를 임의로 지정해 분할하되, 이후 같은 숫자를 입력해 같은 명령을 실행할 때는 같은 조합의 데이터로 구성할 수 있도록 한다.

사이킷런(Scikit-Learn)
기계학습 관련 파이썬 라이브러리로 여러 가지 기계학습 알고리즘이 포함된 모듈과 예시 데이터로 구성되어 있다.

프로그램

```
# 훈련 데이터와 테스트 데이터를 나누기
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size=0.3, random_state=10)
```

❸ 선형 회귀 모델을 이용해 학습을 진행하기 위해 사이킷런 라이브러리에서 선형 회귀(Linear Regression)와 관련된 함수들을 불러온다. 이후 선형 회귀를 위한 알고리즘을 linear_model이라는 이름으로 불러오고, 훈련 데이터인 X_train, y_train을 학습시켜 linear_model이라는 모델을 만든다.

독립 변수 개수, 종속 변수 개수, 관계 등에 따라 회귀 모델을 생성하는 다양한 알고리즘이 있다. 단순 선형 회귀, 다중 선형 회귀, 비선형 회귀 모델 등이 그 예이다.

프로그램

```
# 선형 회귀 모델을 이용해 학습을 진행하기
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression()          # 선형 회귀 모델 생성
linear_model.fit(X_train, y_train)         # 선형 회귀 모델 학습
```

실행 결과

```
▾ LinearRegression
LinearRegression()
```

4 직접 생성한 linear_model 모델을 이용해 X_test 데이터에 대한 y 속성의 기댓값을 예측한다. X_test 안의 속성값들에 따른 y 값을 모델을 이용해 예측한 결과를 화면에서 확인한다.

프로그램

```
1 # 학습 모델을 이용해 X_test 데이터에 대한 y 기댓값을 예측한 결과를 linear_predict 변수에 저장하기
2 linear_predict = linear_model.predict(X_test)
3 # 예측 결과가 들어 있는 linear_predict 값 확인해 보기
4 linear_predict
```

실행 결과

```
array([317.44207752, 279.08799323, 338.86253731, ..., 243.03871947,
       306.34999328, 254.23990494])
```

설명 X_test에 저장되어 있던 테스트 데이터의 독립 변수들에 따른 이산화 탄소 배출량 예측값을 linear_predict라는 이름의 변수로 저장하였다. 그리고 내용을 확인해 보니 X_test 속 데이터들의 순서대로 예측값이 약 317, 279, 338, ... 254로 나타나는 것을 확인할 수 있다.

5 가상의 독립 변수 값들을 넣어 예측값을 확인해 볼 수도 있다.

프로그램

```
1 # 가상의 독립 변수 값을 넣어 예측값이 어떻게 되는지 확인해 보기
2 # 예) 엔진 크기가 3, 평균 연료 소비량이 7.5일 경우의 이산화 탄소 배출량 예측해 보기
3 linear_model.predict([[3.0,7.5]])
```

실행 결과

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not
have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(array([200.71150049])
```

설명 예를 들어 엔진 크기(Engine size) 속성의 값이 3.0, 평균 연료 소비량(Fuel Consumption Comb(L/100km)) 속성의 값이 7.5인 자동차의 경우, 학습 모델이 예측한 이산화 탄소 배출량(CO2 Emissions) 속성 값은 약 200이다.

5단계 성능 평가

모델을 생성하고, 모델을 더 나은 방향으로 개선하기 위해서는 모델 성능 평가가 필요하다. 회귀 모델의 성능 평가에 활용되는 대표적인 값으로 R2 score이 있다. R2 결정계수라고도 하는 이 값은 실젯값과 예측값, 평균값 사이의 편차를 점수로 매긴 것이다. 0~1까지의 수로 나타내며 1에 가까울수록 예측력이 좋다는 것을 의미한다.

코랩(Colab)에서는 print 명령을 사용하지 않아도 마지막 오브젝트에 대해 자동으로 출력하는 기능이 있다.

잠깐

predict() 함수에 대괄호[]를 2개 붙이는 이유?

predict() 함수는 2차원 배열을 사용하기 때문에 입력하고자 하는 가상의 테스트 데이터 또한 2차원 배열의 형태가 되어야 한다. 따라서 대괄호 2개를 사용함으로써 데이터의 형태를 바꿀 수 있다.

사이킷런에는 기계학습 모델의 성능 평가를 하기 위한 함수가 포함되어 있다. R2 score를 계산하기 위한 함수인 r2_score 함수를 사용한다.

프로그램

```
from sklearn.metrics import r2_score
r2_score(y_test,linear_predict)
```

실행 결과

```
0.9846084439201376
```

문제 2 수업 태도에 따른 학업 성취 수준 구분하기

문제 상황 **흔히 학업 성적을 향상하기 위해서는 수업에 열심히 참여해야 한다고 한다. 과연 수업에 적극적으로 참여하는지 여부가 성적과 관련이 있을까? 성별, 수업 시간 내 발표를 위해 손을 든 횟수, 수업 자료 조회 횟수, 공지 확인 횟수, 토론 참여 횟수, 결석 횟수 등의 속성을 통해 상위, 중위, 하위 성적으로 구분해 보자.**

생각해 보기

1. 상위, 중위, 하위 성적군 결정에 영향을 미치는 요소에는 무엇이 있을까?
2. 속성들을 통해 성적군을 분류해 보는 것이 기계학습으로 해결 가능한 문제인가? 그렇게 생각한 이유는 무엇인가?
3. 상위, 중위, 하위 성적으로 구분하기 위해 어떤 기계학습 유형을 사용하면 좋을까?

1단계 문제 정의

문제를 해결하기 위해 기계학습으로 해결할 수 있는 문제로 정의해 보자.

발표 시도 횟수(손을 든 횟수), 수업 자료 조회 횟수(자료 접근 횟수), 결석 횟수 등의 속성을 이용해 상위, 중위, 하위 성적군으로 분류해 본다.

2단계 데이터 탐색 및 전처리

❶ 데이터 수집하기

캐글(Kaggle) 사이트(http://www.kaggle.com)에서 제공하는 데이터를 사용한다. “Students’ Academic Performance”를 검색하고 화면 하단의 다운로드 버튼을 눌러 데이터를 내려받는다.

캐글(Kaggle)

전 세계 데이터 과학자들이 데이터 분석 대회를 개최하고 분석 내용을 공유하는 커뮤니티로, 데이터 분석을 위한 데이터셋과 무료 강의 등을 제공한다.

❷ 데이터 불러오기

❶ 'xAPI-Edu-Data.csv' 파일을 불러와 코랩에 업로드하고 myfile에 저장한다.

프로그램

```
from google.colab import files                    # 코랩에 파일 업로드하기
myfile = files.upload()
```

실행 결과

```
파일 선택  xAPI-Edu-Data.csv
• xAPI-Edu-Data.csv(text/csv) - 38026 bytes, last modified: 2023. 12. 3. - 100% done
Saving xAPI-Edu-Data.csv to xAPI-Edu-Data.csv
```

❷ 데이터 처리에 필요한 pandas 라이브러리를 pd라는 별칭으로 불러온다. 그리고 업로드한 데이터의 내용을 df라는 변수에 불러와 저장한다.

프로그램

```
import pandas as pd    # 데이터 처리를 위한 pandas 라이브러리 설치하기
# csv 파일의 데이터를 불러와 df라는 변수로 저장하기
df = pd.read_csv("xAPI-Edu-Data.csv")
```

❸ df.head( )를 이용해 df에 저장한 데이터 중 상위 5개 데이터를 화면에 나타내 데이터의 내용을 확인한다. head( ) 함수는 상위 일부 행의 데이터를 확인할 수 있는 함수다.

프로그램

```
# head() 함수를 이용해 데이터셋의 상위 5줄 확인하기
df.head()
```

실행 결과

	gender	NationalITy	PlaceofBirth	StageID	GradeID	SectionID	Topic	Semester	Relation	raisedhands	VisITedResources	AnnouncementsVi
0	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	15	16	
1	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	20	20	
2	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	10	7	
3	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	30	25	
4	M	KW	KuwaIT	lowerlevel	G-04	A	IT	F	Father	40	50	

설명 csv 파일이 어떤 형태로 구성되어 있는지 확인해 보니, 성별(gender), 손을 든 횟수(raisedhands), 자료 접근 횟수(visitedResources) 등의 속성으로 이루어져 있는 정형 데이터인 것을 알 수 있다.

❸ 결측치 확인 및 제거

❶ len( )은 데이터 개수(행의 개수), df.info( )는 데이터 속성의 자료형과 자료 개수를 알려 준다. 이 두 함수를 통해 데이터의 정보를 확인할 수 있다.

판다스 라이브러리

판다스(Pandas)는 데이터 조작 및 분석을 위한 파이썬 라이브러리로, 표 형식의 데이터를 쉽게 다루도록 해 준다. 주로 데이터 프레임(DataFrame)과 시리즈(Series)라는 두 가지 주요 데이터 구조를 다루며, 데이터를 불러오거나 정렬, 필터링, 그룹화, 통계 계산, 시각화하는 등의 기능을 포함한다.

내 컴퓨터에 있는 데이터 파일을 선택해 업로드를 완료하면 '100% done'이라는 말과 데이터 파일의 이름이 뜬다.

속성 설명

- gender: 성별(남:M, 여:F)
- Nationality: 국적
- PlaceofBirth: 출생지
- StageID: 학교급 정보
- GradeID: 학년 정보
- SectionID: 분반 정보
- Topic: 학습 과목
- Semester: 학기
- Relation: 보호자
- raisedhands: 손을 든 횟수
- visitedResources: 자료 접근 횟수
- AnnouncementsView: 공지 확인 횟수
- Discussion: 토론 참여 횟수
- ParentAnsweringSurvey: 학부모 설문 참여도
- ParentschoolSatisfaction: 학부모 학교 만족도
- StudentAbsenceDays: 결석 횟수(7회 이상: 1, 미만: 0)
- Class: 성적 등급(L: 낮음, M: 보통, H: 높음)

프로그램

```
# len 함수를 이용해 데이터셋이 총 몇 개의 행으로(데이터로) 이루어져 있는지 알아보기
len(df)
```

실행 결과

```
480
```

프로그램

```
# info 함수를 이용해 데이터 속성의 자료형과 자료 개수 알아보기
df.info()
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   gender                    480 non-null    object
 1   NationalITy               480 non-null    object
 2   PlaceofBirth              480 non-null    object
 3   StageID                   480 non-null    object
 4   GradeID                   480 non-null    object
 5   SectionID                 480 non-null    object
 6   Topic                     480 non-null    object
 7   Semester                  480 non-null    object
 8   Relation                  480 non-null    object
 9   raisedhands               480 non-null    int64
 10  VisITedResources          480 non-null    int64
 11  AnnouncementsView         480 non-null    int64
 12  Discussion                480 non-null    int64
 13  ParentAnsweringSurvey     480 non-null    object
 14  ParentschoolSatisfaction  480 non-null    object
 15  StudentAbsenceDays        480 non-null    object
 16  Class                     480 non-null    object
dtypes: int64(4), object(13)
memory usage: 63.9+ KB
```

설명 info()의 결과를 보아 480개의 데이터로 이루어진 것을 알 수 있으며, 각 속성은 모두 정수형, 실수형 수칫값 또는 문자형으로 이루어졌다는 것을 알 수 있다. Dtype에서, int64는 정수형 수칫값, float64는 실수형 수칫값, object는 그 외 문자형 등의 값을 의미한다. 또한 non-null이라는 것으로 보아 결측치가 없는 데이터라고 할 수 있다.

잠깐

데이터는 기계학습 모델의 성능을 결정하는 데 중요한 영향을 미친다. 특히, 핵심 속성 선정이 잘못되어 해결할 문제와 관련 없는 데이터 속성이 학습 과정에 반영된다면 오히려 인공지능의 잘못된 판단을 이끌어낼 수 있다. 따라서 여러 데이터 속성 중 결과와 상관관계가 높은 속성들을 적절히 선별해 내는 작업이 필요하다.

2 isna( ).sum( )을 통해 결측치가 있는지 확인한다. 이번에 사용할 정형 데이터에는 결측치가 없는 것을 알 수 있다.

프로그램

```
# isna().sum() 함수를 이용해 속성별 결측치 개수의 합 확인하기
df.isna().sum()
```

drop( )을 이용해 필요 없는 속성을 데이터 프레임에서 삭제할 수도 있다.

실행 결과

```
gender                      0
NationalITy                 0
PlaceofBirth                0
StageID                     0
GradeID                     0
SectionID                   0
Topic                       0
Semester                    0
Relation                    0
raisedhands                 0
VisITedResources            0
AnnouncementsView           0
Discussion                  0
ParentAnsweringSurvey       0
ParentschoolSatisfaction    0
StudentAbsenceDays          0
Class                       0
dtype: int64
```

❸ 문제 정의 단계에서 선정한 핵심 속성인 손을 든 횟수, 자료 접근 횟수, 공지 확인 횟수, 토론 참여 횟수, 결석 횟수, 성적 등급만 포함하는 데이터 프레임을 df1이라는 변수에 저장한다.

프로그램

```
# 특정 속성들의 데이터들만 뽑아 df1이라는 이름의 데이터 프레임 구성하기
df1 = df[['raisedhands','VisITedResources', 'StudentAbsenceDays', 'Class']]
```

프로그램

```
# head()함수를 이용해 df1의 상위 5개 데이터 확인하기
df1.head()
```

실행 결과

	raisedhands	VisITedResources	StudentAbsenceDays	Class
0	15	16	Under-7	M
1	20	20	Under-7	M
2	10	7	Above-7	L
3	30	25	Above-7	L
4	40	50	Above-7	M

replace()는 replace(현재 내용, 대체할 내용)의 형태로 사용하지만, 여러 문자열 세트를 동시에 바꾸고 싶다면 replace({현재 내용1: 대체할 내용 1, 현재 내용2: 대체할 내용 2})와 같이 한 번에 적용 가능하다.

❹ 학습에 적합한 데이터 형태로 바꾸기 위해 모든 속성의 값을 수치형 데이터로 바꾼다. 그리고 head( ) 함수를 이용해 상위 5개 데이터를 확인해 잘 바뀌었는지 점검한다.

프로그램

```
# 기계학습에 활용하기 위해 replace()를 이용해 문자형 데이터를 수치형 데이터로 변환하기
df1['StudentAbsenceDays'] = df1['StudentAbsenceDays'].replace({'Under-7':0,'Above-7':1})
df1['Class'] = df1['Class'].replace({'L':0,'M':1,'H':2})
```

잠깐

기계학습 알고리즘은 문자형 데이터 속성을 가지고는 속성 간의 상관관계를 파악할 수 없어 가설 함수를 만들 수 없다. 따라서 데이터를 수치형 데이터로 변환해야 한다.

아래부터 몇 개의 데이터를 확인하고 싶을 경우, head() 대신 tail()을 사용할 수 있다.

프로그램

```
1 df1.head()
```

실행 결과

	raisedhands	VisITedResources	StudentAbsenceDays	Class
0	15	16	0	1
1	20	20	0	1
2	10	7	1	0
3	30	25	1	0
4	40	50	1	1

3단계 기계학습 유형과 알고리즘 선정

적합한 기계학습 유형을 선정해 보고, 그 이유를 친구와 이야기해 보자.

대표적인 분류 알고리즘은 189쪽 알고 가기를 참고하자.

- 적합한 기계학습 유형: 분류
- 이유: 예 여러 요인들을 통해 학습 수준을 '상위, 중위, 하위' 세 항목 중에서 어느 항목에 해당하는지를 파악하고자 한다. 또한 학습 데이터에 '상위, 중위, 하위'로 분류된 레이블이 포함되어 있으므로 지도학습을 적용할 수 있다. 따라서 지도학습의 분류를 이용한다.
- 독립 변수와 종속 변수 선정: 예 독립 변수는 손을 든 횟수, 자료 접근 횟수, 결석 횟수로 선정하고, 종속 변수는 성적 등급으로 선정한다.

분류 모델은 알고리즘에 따라 여러 가지로 나뉘는데, 그 중 가장 대표적인 로지스틱 회귀 알고리즘을 이용해 다중 분류 모델을 만들어 본다.

4단계 기계학습을 통한 모델 생성

학습 모델을 만들기 위하여 데이터셋의 여러 속성을 독립 변수와 종속 변수로 나누어 각각을 X와 y에 저장한다. 또한 데이터를 훈련 데이터와 테스트 데이터로 나누어 각각을 X_train, X_test, y_train, y_test로 저장한다.

1 학습할 속성이 될 독립 변수 X와 타깃(목표) 속성이 될 종속 변수 Y를 나누어 저장한다.

drop() 함수는 특정 데이터를 삭제하는 함수이다. axis=1은 세로 방향으로 drop 함수를 적용하겠다는 의미로, 특정 속성의 데이터를 삭제하게 된다.

프로그램

```
1 # 학습할 속성이 될 X(독립 변수)와 타깃(목표) 속성이 될 y(종속 변수)를 나누어 저장하기
2 X = df1.drop(labels = ['Class'],axis = 1)
3 y = df1['Class']
```

설명 df1에서 Class라는 이름의 속성을 찾아 세로 방향으로 삭제 기능을 수행한다. 즉, Class 속성에 해당하는 데이터들을 데이터셋에서 삭제하여 X 데이터에는 나머지 속성들의 데이터만 남게 된다. 종속 변수인 y에는 Class 속성에 해당하는 데이터만 저장된다.

❷ 수치 데이터 정규화: raisedhands, discussion 등의 속성들이 다른 속성보다 훨씬 큰 값을 가지고 있는 것처럼, 속성별로 수칫값의 크기가 다를 수 있다. 이러한 경우에는 기계학습 결과에 속성들이 얼마나 큰 영향을 미치는지, 얼마나 큰 관계가 있는지를 정확히 파악하기 어렵다. 따라서 기계학습 모델의 정확도를 높이기 위해 속성들의 수칫값을 0부터 1 사이의 범위로 정규화한다.

정규화

속성 값들을 0.0~1.0 사이의 실숫값으로 변환해 범위(scale)를 일정 수준으로 맞추는 과정이다. 범위가 큰 속성이 다른 속성보다 결과에 더 큰 영향을 미치는 것을 방지하기 위해 진행한다.

프로그램

```
# Min-Max 정규화를 위해 데이터 프레임을 배열로 변환하기
from sklearn.preprocessing import MinMaxScaler
data_array = X.values

# Min-Max 정규화 객체 생성하기
scaler = MinMaxScaler()

# 데이터 정규화하기
normalized_data = scaler.fit_transform(data_array)

# 정규화된 데이터를 다시 데이터 프레임으로 변환하기
X = pd.DataFrame(normalized_data, columns = X.columns)
```

위의 정규화 코드만 실행하면 결과가 화면에 출력되지 않으니, 데이터 프레임의 형태를 보고 싶다면 head() 함수를 사용해 상위 5줄을 출력해 확인한다.

프로그램

```
X.head()
```

(위 코드에 이어서) head()를 이용해 내용 확인

실행 결과

	raisedhands	VisITedResources	StudentAbsenceDays
0	0.15	0.161616	0.0
1	0.20	0.202020	0.0
2	0.10	0.070707	1.0
3	0.30	0.252525	1.0
4	0.40	0.505051	1.0

❸ 기존의 데이터셋을 훈련 데이터와 테스트 데이터로 나눈다. 데이터셋을 X_train, X_test, y_train, y_test라는 이름으로 분할해 저장한다. 이때 훈련 데이터와 테스트 데이터의 비율은 7:3으로 분할하며, random_state는 데이터를 임의로 지정해 분할하되, 이후 같은 숫자를 입력해 같은 명령을 실행할 때는 같은 조합의 데이터로 구성할 수 있도록 해 준다.

사이킷런 모듈

사이킷런(Scikit-Learn) 라이브러리는 파이썬 프로그래밍 언어 기반의 기계학습을 할 때 사용하는 라이브러리다. 기계학습 알고리즘을 제공하며 분류, 회귀 등 다양한 모델을 만들 때 사용한다.

프로그램

```
# 훈련 데이터와 테스트 데이터를 나누기
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
random_state=10)
```

로지스틱 회귀

선형 모델의 일종으로, 데이터를 분류할 때 직선 형태의 결정 경계를 사용한다. 클래스를 구분하는 직선을 찾고, 그 선을 기준으로 데이터를 분류한다.

4 로지스틱 회귀 모델을 이용해 학습을 진행한다.

프로그램

```
# 로지스틱 회귀 모델을 이용해 학습을 진행하기
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression()
logistic_model.fit(X_train,y_train)
```

실행 결과

```
▾ LogisticRegression
LogisticRegression()
```

5 모델을 이용해 X_test 데이터에 대한 y 기댓값을 예측한다.

프로그램

```
# 학습 모델을 이용해 X_test의 데이터에 대한 y 기댓값을 예측한 결과 logistic_predict에 저장하기
logistic_predict = logistic_model.predict(X_test)
```

프로그램

```
# 예측 결과가 담긴 logistic_predict 변수 확인하기
logistic_predict
```

실행 결과

```
array([1, 1, 1, 0, 1, 2, 2, 2, 1, 0, 2, 1, 1, 1, 1, 2, 0, 2, 1, 2, 0, 1,
       0, 2, 1, 0, 2, 1, 1, 2, 2, 2, 0, 2, 2, 0, 1, 1, 0, 2, 1, 0, 1, 0,
       1, 2, 2, 1, 1, 2, 1, 0, 1, 1, 1, 2, 2, 0, 1, 2, 2, 0, 2, 1, 0, 1,
       1, 2, 1, 2, 1, 0, 1, 1, 1, 1, 0, 0, 2, 2, 0, 2, 0, 0, 1, 2, 1, 2,
       2, 1, 2, 0, 1, 1, 2, 1, 2, 0, 2, 0, 0, 1, 2, 1, 1, 1, 1, 0, 1, 0,
       1, 2, 2, 0, 0, 1, 0, 2, 1, 0, 0, 1, 1, 1, 1, 2, 0, 1, 2, 2, 1, 2,
       0, 1, 0, 0, 0, 1, 0, 2, 1, 0, 2, 1])
```

설명 X_test에 저장되어 있던 테스트 데이터의 독립 변수들에 따른 성적 수준 예측값을 logistic_predict라는 이름의 변수로 저장하였다. 그리고 내용을 확인해 보니 X_test 속 데이터들의 순서대로 예측값이 1,1,1,0,1,...로 나타나는 것을 확인할 수 있다. 즉, X_test 속 첫 번째 데이터에 해당하는 학생은 성적 등급이 1이라는 의미이므로 'M' 등급에 해당한다.

Tip

프로그램

```
print(logistic_predict)
```

실행 결과

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

185쪽에서 replace()를 통해 'L' 등급을 0으로, 'M' 등급을 1로, 'H' 등급을 2로 변환했던 것을 떠올려 보자.

print(logistic_predict)로 결과를 확인하는 것도 가능하다.

5단계 성능 평가

모델을 생성하고, 모델을 더 나은 방향으로 개선하기 위해서는 모델 성능 평가가 필요하다. 성능 평가에 활용되는 대표적인 통계값으로 정확도(Accuracy)가 있다. 정확도란

전체 예측 건수에서 정답을 예측한 건수에 대한 비율을 말한다. 분류 모델의 정확도가 높을수록 테스트 데이터에 대한 결과를 정확하게 예측할 수 있는 모델이다. 정확도가 낮아 성능이 떨어질 경우에는 훈련 데이터를 재수집하거나 알고리즘 구현 코드를 수정함으로써 모델의 정확도를 높여야 한다. 혼동 행렬을 이용해 성능을 평가할 수도 있다.

혼동 행렬은 '오차 행렬'이라고도 부르며, 분류 모델의 성능을 파악하기 위한 값을 한눈에 보기 쉽게 표의 형태로 나타낸다.

프로그램

```
1 from sklearn.metrics import accuracy_score
2 accuracy_score(y_test, logistic_predict)
```

실행 결과

```
0.7291666666666666
```

프로그램

```
1 from sklearn.metrics import confusion_matrix
2 confusion_matrix(y_test, logistic_predict)
```

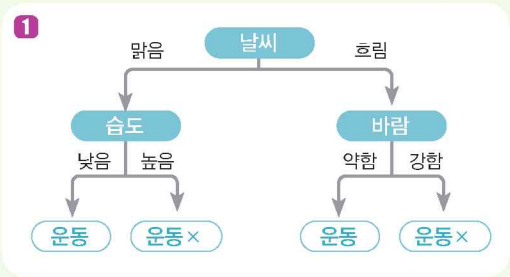
실행 결과

```
array([[32,  6,  1],
       [ 6, 46, 18],
       [ 0,  8, 27]])
```

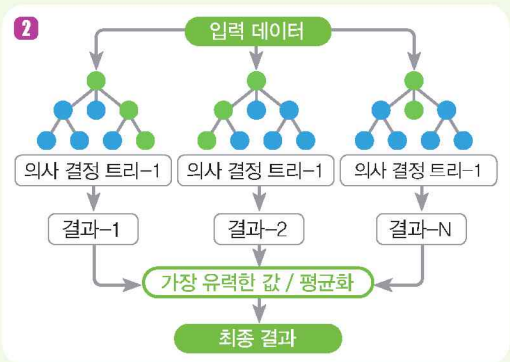
같은 분류 학습을 진행하더라도 사용하는 알고리즘에 따라 학습 결과와 성능은 다양해질 수 있다. 예를 들어, 로지스틱 회귀를 이용해 약 0.72의 정확도가 나왔다고 해도 다른 알고리즘을 활용하면 정확도가 달라질 수 있다. 따라서 때에 따라 성능이 가장 높은 알고리즘을 선택해 학습에 이용하는 것이 중요하다.

알고 가기 대표적인 분류 알고리즘

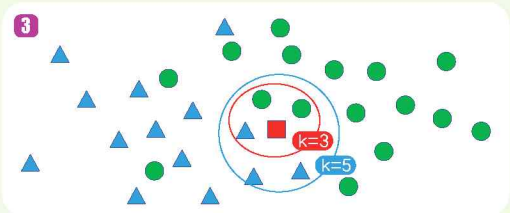
1 의사 결정 트리: 의사 결정 규칙과 그 결과들을 트리 형태로 표현하고 특정 기준에 따라 데이터를 마지막 기준까지 계속 분할해 나가는 모델이다. 특정 기준이 의사 결정 트리의 노드(node, 가지)마다 주어지며 특정 기준을 만족하는지 여부에 따라 노드에서 데이터가 둘로 분리되어 자식 노드로 데이터가 전달된다. 이런 데이터 분리 과정은 자식이 없는 리프 노드(leaf node)에 도달할 때까지 계속된다.



2 랜덤 포레스트(Random Forest): 여러 개의 의사 결정 트리 결과 예측 값 중에 가장 많이 나온 값을 최종 결과로 결정하는 모델이다. 랜덤 포레스트는 하나의 의사 결정 트리가 가지고 있는 장점은 그대로 가지면서 단점은 보완한다는 점에서 모델의 성능을 높일 수 있다. 이렇게 여러 의사 결정 트리를 이용해 성능을 높이는 랜덤 포레스트는 여러 학습 모델을 결합해 판단을 내리는 기계 학습 방법인 '앙상블 기법'의 한 예이다.



3 K-최근접 이웃 알고리즘(K-NN): 새로운 데이터를 분류하기 위해 가장 가까이에 있는 K개의 학습 데이터가 가장 많이 속한 그룹을 따라 분류하는 모델이다. 간단하고 효율적인 알고리즘
이지만 K의 값을 잘 선정해야 할 필요가 있다.



문제 3 자전거 대여소 위치 분석을 통해 관리소 위치 선정하기

문제 상황 **A 지역은 지속 가능한 발전을 실현하기 위해 공공 자전거 사업을 운영하고 있다. 공공 자전거 대여소를 지역 곳곳에 설치하고 사람들이 자유롭게 대여 및 반납할 수 있도록 하는 것이다. 그런데, 자전거의 대수가 늘어나고 사용량이 늘다 보니 각 대여소를 관리하기 위한 구간별 대여 관리소를 만들어야 한다는 의견이 대두되었다. 기존 자전거 대여소의 위치를 위도, 경도를 이용해 파악하고 적합한 위치를 찾아 관리소를 설치하고자 한다. 관리소 개수가 많으면 대여소까지의 접근은 쉬워지지만 건설, 유지 비용이 증가할 수 있다. 적절한 위치를 선정해 관리소를 설치하려면 어디에 설치하는 것이 좋을까?**

생각해 보기

1. 자전거 대여소의 위치를 가까운 곳끼리 묶는 것이 기계학습으로 해결 가능한 문제인가? 그렇게 생각한 이유는 무엇인가?
2. 가까운 곳끼리 묶고 그에 따른 구간별 관리소 위치를 선정하기 위해 어떤 기계학습 유형을 사용해야 할까?

1단계 문제 정의

문제를 해결하기 위하여 기계학습으로 해결할 수 있는 문제로 정의해 보자.

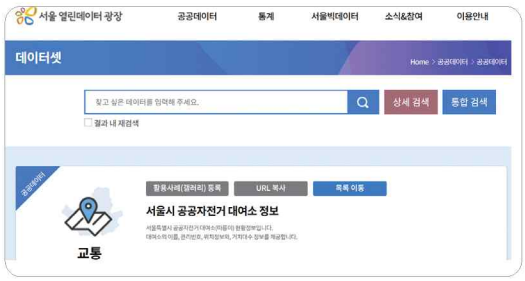
자전거 대여소의 위치를 위도와 경도를 이용해 가까운 곳끼리 묶어 2개의 그룹으로 나누고, 이를 관리할 관리소의 적절한 설치 위치를 선정해 보자.

공공 데이터 사이트와 이용 방법은 59쪽을 참고하자.

2단계 데이터 탐색 및 전처리

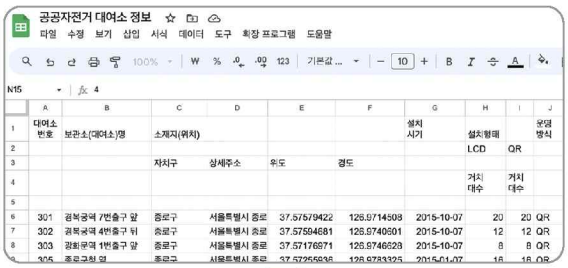
① 데이터 수집하기

1. 서울시 열린 데이터 광장 사이트에서 '공공자전거 대여소 정보'를 검색해 데이터를 내려받는다.



서울 열린 데이터 광장(https://data.seoul.go.kr/) 사이트는 서울과 관련된 다양한 분야의 공공 데이터가 제공되는 데이터 포털 사이트이다. 국내 데이터를 찾고자 한다면 캐글보다 공공 데이터 포털, 서울 열린 데이터 광장, 통합 데이터 지도, kosis 등의 국내 사이트를 활용하는 것이 좋다.

2. 다운로드한 데이터를 엑셀, 한셀, 구글 스프레드시트 등의 응용 프로그램으로 열고, 필요 없는 상위 5행을 삭제한 후 1행을 추가해 각 열의 속성을 적절한 속성명으로 지정하고 저장한다. 여기서는 속성 순서대로 '대여소 번호', '대여소명', '자치구', '상세 주소', '위도', '경도', '설치 시기', 'LCD 거치대수', 'QR 거치대수', '운영 방식'으로 지정한다.



	A	B	C	D	E	F	G	H	I	J
1	대여소번호	대여소명	자치구	상세주소	위도	경도	설치시기	LCD거치대수	QR거치대수	운영방식
2	301	경복궁역 7번출구 앞	종로구	서울특별시 종로	37.57579422	126.9714508	2015-10-07	20	20	QR
3	302	경복궁역 4번출구 뒤	종로구	서울특별시 종로	37.57594681	126.9740601	2015-10-07	12	12	QR
4	303	광화문역 1번출구 앞	종로구	서울특별시 종로	37.57176971	126.9746628	2015-10-07	8	8	QR
5	305	종로구청 옆	종로구	서울특별시 종로	37.57255936	126.9783325	2015-01-07	16	16	QR
6	307	서울역사박물관 앞	종로구	서울특별시 종로	37.56999969	126.9710999	2015-10-07	11	11	QR
7	308	광화문 SK허 앞	종로구	서울특별시 종로	37.56996918	126.973938	2015-10-07	20	31	QR

구글 스프레드시트로 데이터 확인하고 속성명 수정하기

구글 스프레드시트로 작업한 결과를 csv 파일로 다운로드하려면 [파일] – [다운로드] – '쉼표로 구분된 값(.csv)'을 클릭하면 컴퓨터의 '다운로드' 폴더에 저장된다.

❷ 데이터 불러오기

❶ '공공자전거 대여소 정보.csv' 파일을 불러와 코랩에 업로드하고 myfile에 저장한다.

프로그램

```
1 from google.colab import files
2 myfile=files.upload()
```

실행 결과

```
파일 선택  공공자전거 대여소 정보.csv
• 공공자전거 대여소 정보.csv(text/csv) - 292267 bytes, last modified: 2023. 10. 16. - 100% done
Saving 공공자전거 대여소 정보.csv to 공공자전거 대여소 정보.csv
```

한글이 담긴 csv 파일이 인코딩 방식으로 인해 종종 불러오기가 되지 않는 경우가 있다. 이럴 경우 아래와 같이 인코딩 옵션을 지정해 주면 오류가 해결된다.
df=pd.read_csv("공공자전거 대여소 정보.csv", encoding='cp949')

❷ 데이터 처리에 필요한 pandas 라이브러리를 pd라는 별칭으로 불러온다. 그리고 업로드한 데이터의 내용을 df라는 변수에 불러와 저장한다

프로그램

```
1 # 데이터 처리를 위한 pandas 라이브러리를 불러오고 pd라는 별칭 지정하기
2 import pandas as pd
3 # csv 파일의 데이터를 불러와 df라는 이름의 변수로 저장하기
4 df=pd.read_csv("공공자전거 대여소 정보.csv",encoding='cp949')
```

인코딩(Encoding) 방식
컴퓨터 안에서 텍스트를 기계가 이해할 수 있는 언어로 표현하는 방법을 인코딩이라고 한다. 'cp949' 방식 또한 한글을 읽기 위해 인코딩하는 방법 중 하나다.

❸ df.head( )를 이용해 df에 저장한 데이터 중 상위 5개 데이터를 화면에 나타내 데이터의 내용을 확인한다. head( ) 함수는 상위 일부 행의 데이터를 확인할 수 있는 함수다.

프로그램

```
1 # head() 함수를 이용해 불러온 데이터 내용의 상위 5줄 확인하기
2 df.head()
```

	번호	보관소명	자치구	상세주소	위도	경도	설치시기	LCD거치대수	QR거치대수	운영방식
0	301	경복궁역 7번출구 앞	종로구	서울특별시 종로구 사직로 지하130 경복궁역 7번출구 앞	37.575794	126.971451	2015-10-07	20.0	20.0	QR
1	302	경복궁역 4번출구 뒤	종로구	서울특별시 종로구 사직로 지하130 경복궁역 4번출구 뒤	37.575947	126.974060	2015-10-07	12.0	12.0	QR
2	303	광화문역 1번출구 앞	종로구	서울특별시 종로구 세종대로 지하189 세종로공원	37.571770	126.974663	2015-10-07	8.0	8.0	QR
3	305	종로구청 옆	종로구	서울특별시 종로구 세종로 84-1	37.572559	126.978332	2015-01-07	16.0	16.0	QR
4	307	서울역사박물관 앞	종로구	서울특별시 종로구 새문안로 55 서울역사박물관 앞	37.570000	126.971100	2015-10-07	11.0	11.0	QR

설명 csv 파일을 확인해 보니, 보관소명, 자치구, 상세 주소, 위도, 경도 등의 속성으로 이루어져 있는 정형 데이터인 것을 알 수 있다.

❸ 결측치 확인 및 제거

❶ len( )은 데이터 개수(행의 개수), df.info( )는 데이터 속성의 자료형과 자료 개수를 알려 준다. 이 두 함수를 통해 데이터의 정보를 확인할 수 있다.

프로그램

```
# len 함수를 이용해 데이터셋이 총 몇 개의 행으로(데이터로) 이루어져 있는지 알아보기
len(df)
```

실행 결과

```
2749
```

프로그램

```
# info 함수를 이용해 데이터 속성의 자료형과 자료 개수 알아보기
df.info()
```

실행 결과

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2749 entries, 0 to 2748
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   번호        2749 non-null   int64
 1   보관소명      2749 non-null   object
 2   자치구       2749 non-null   object
 3   상세주소      2749 non-null   object
 4   위도        2749 non-null   float64
 5   경도        2749 non-null   float64
 6   설치시기      2749 non-null   object
 7   LCD거치대수   1315 non-null   float64
 8   QR거치대수    1582 non-null   float64
 9   운영방식      2749 non-null   object
dtypes: float64(4), int64(1), object(5)
memory usage: 214.9+ KB
```

❷ isna( ).sum( )을 통해 결측치가 있는지 확인한다. 이번에 사용할 정형 데이터에는 'LCD 거치대수', 'QR 거치대수' 속성에 결측치가 있는 것을 알 수 있다.

프로그램

```
# isna().sum() 함수를 이용해 속성별 결측치 개수의 합 확인하기
df.isna().sum()
```

실행 결과

```
번호          0
보관소명        0
자치구         0
상세주소        0
위도          0
경도          0
설치시기        0
LCD거치대수    1434
QR거치대수     1167
운영방식        0
dtype: int64
```

❸ 문제 정의 단계에서 선정한 핵심 속성인 위도, 경도만 포함하는 데이터 프레임을 df1이라는 변수에 저장한다. 그리고 head() 함수를 이용해 df1의 상위 5행의 데이터를 확인해 본다.

프로그램

```
# 필요한 속성인 위도, 경도 속성을 df1이라는 변수에 저장하기
df1=df[['위도', '경도']]
```

프로그램

```
# head()를 이용해 df1의 상위 5행 확인하기
df1.head()
```

실행 결과

	위도	경도
0	37.575794	126.971451
1	37.575947	126.974060
2	37.571770	126.974663
3	37.572559	126.978332
4	37.570000	126.971100

데이터 프레임

판다스에서 지원하는 데이터 구조로, 표 형식으로 데이터를 저장하고 관리할 수 있다. 데이터 관리에 필요한 여러 가지 기능을 포함하고 있다.

drop( )을 이용해 필요 없는 속성을 데이터 프레임에서 삭제할 수도 있다.

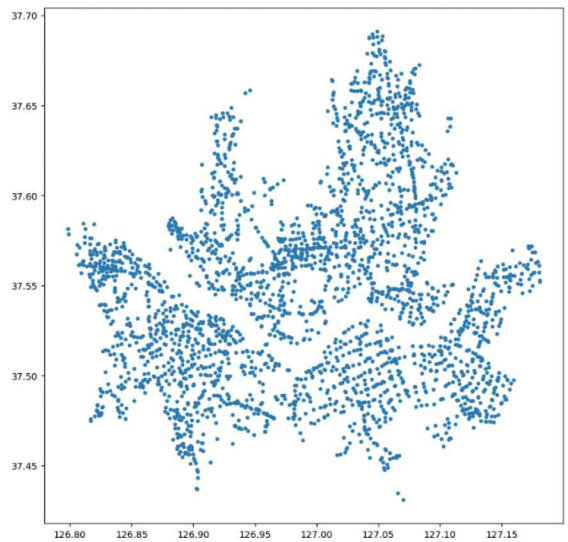
❹ 위도, 경도 속성의 데이터를 알고 있기 때문에 이를 산점도 형태로 시각화할 수 있다. 맷플롯립(matplotlib) 라이브러리를 불러온 후 scatter()를 이용해 산점도를 그린다. 가로 축을 경도, 세로 축을 위도로 설정하고 점의 크기는 10으로 설정한다.

프로그램

```
# 시각화를 위한 맷플롯립(matplotlib) 라이브러리를 불러와 plt라는 별칭 지정하기
import matplotlib.pyplot as plt

# 자전거 대여소의 위치를 산점도 그래프로 그리기
plt.figure(figsize=(10,10))
plt.scatter(df['경도'], df['위도'], s=10)
plt.show()
```

실행 결과



Tip

데이터 시각화를 위한 라이브러리인 맷플롯립(matplotlib)을 사용하면 위도와 경도 데이터를 이용해 각 대여소의 위치를 산점도로 시각화할 수 있다.

seaborn(시본) 라이브러리와 matplotlib(맷플롯립)을 기반으로 더 다양한 그래프, 색상 테마, 통계용 그래프 제공 등의 기능이 추가된 시각화 라이브러리를 활용할 수도 있다.

산점도에 대한 설명은 65쪽을 참고하자.

설명 산점도의 가로 축이 경도, 세로 축이 위도 값이기 때문에 각 대여소 위치가 지도 위의 점처럼 표시되었다. 서울 지역 대여소에 관련된 데이터이므로 시각화 결과가 서울 지역의 형태로 나타난다.

3단계 기계학습 유형과 알고리즘 선정

• 적합한 기계학습 유형: 군집

• 그 이유: 예 데이터 속 정보를 이용해 비슷한 위도, 경도를 가진 점들끼리 그룹을 만들고 중간 지점을 찾고자 하는 상황이다. 개수가 많으므로 육안이나 개별적인 계산으로 관리소 위치 선정을 하는 것은 어렵다. 따라서 비지도학습의 군집을 이용한다.

사용할 데이터는 위도, 경도 속성을 가진 데이터이며 레이블이 정해져 있지 않은 데이터이므로 군집에 활용할 수 있다. 군집의 대표적인 알고리즘인 k-평균(k-means) 알고리즘을 이용해 군집의 개수를 두 개로 설정한 모델을 만들어 본다.

k-평균 알고리즘이란 주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 레이블이 없는 데이터를 입력받아 각 데이터에 레이블을 할당함으로써 군집을 만든다.

4단계 기계학습을 통한 모델 생성

✿ K의 값에 따라 기계학습 모델의 성능이 달라질 수 있다.

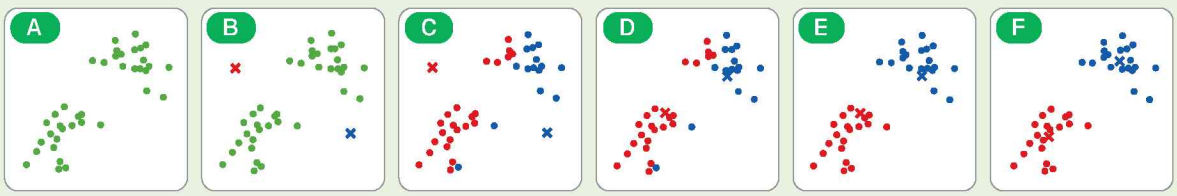
❶ k-평균 알고리즘을 이용하기 위해 사이킷런 라이브러리의 kmeans 모듈을 불러온다. 군집의 개수는 2개로 설정하고, df1을 학습시켜 군집 학습 모델을 생성한다.

프로그램

```
1 from sklearn.cluster import KMeans                    # K-Means 군집을 위한 모듈 불러오기
2 KMeans = KMeans(n_clusters = 2, random_state = 0)     # 군집의 개수는 2개로 설정하기
3 clus_model=KMeans.fit(df1)                            # fit()을 이용해 데이터 학습하기
```

알고 가기 k-평균(k-means) 알고리즘이란?

군집화의 목표는 서로 유사한 데이터끼리 같은 군집으로 묶는 것이다. 그룹 개수(K)를 정한 후 각 데이터로부터 그 데이터가 속한 군집 중심까지의 평균 거리를 최소화시킬 수 있는 군집으로 분류한다.



A 의 원래 데이터 상태에서 B 와 같이 K개의 임의의 중심점을 정한다.

C 임의의 중심점에서 각 데이터를 가장 가까운 중심점으로 할당한다.

D 군집으로 지정된 데이터를 기반으로 해당 군집의 중심점 위치를 재설정한다.

E 재설정된 중심점 위치까지의 거리를 기준으로 각 데이터를 다시 군집으로 분류하고, 분류된 군집의 중심점 위치를 재설정한다. 이 과정을 반복하다가 중심점 위치가 바뀌지 않으면 반복을 중지하고 군집을 확정한다.

❷ 두 개의 군집으로 나눈 결과를 저장할 group이라는 속성(열)을 추가하고, head( ) 함수를 이용해 군집 결과를 확인한다. groupby( )를 이용해 group 속성의 결과가 같은 것끼리 묶고 count( )를 이용해 개수를 확인한다.

프로그램

```
1 # 군집 결과를 저장할 열을 'group'으로 추가하기
2 df1['group']=clus_model.labels_
```

프로그램

```
1 # head() 함수를 이용해 df1의 상위 5행 데이터 확인하기
2 df1.head()
```

실행 결과

	위도	경도	group
0	37.575794	126.971451	0
1	37.575947	126.974060	0
2	37.571770	126.974663	0
3	37.572559	126.978332	0
4	37.570000	126.971100	0

프로그램

```
1 # 그룹별 데이터 개수 확인하기
2 df1.groupby('group').count()
```

실행 결과

group	위도	경도
0	1250	1250
1	1499	1499

설명 앞서 만든 clus_model 이라는 모델을 이용해 데이터들을 2개의 군집으로 구분하였다. 그리고 그 결과를 group이라는 속성명으로 df1에 추가하였다. group 0에는 총 1250개의 위도, 경도 데이터가 있으며 group 1에는 1499개의 위도, 경도 데이터가 있다는 것으로 보아, group 0에는 1250개의 대여소 데이터가 속하며, group 1에는 1499개의 대여소 데이터가 속한다는 것을 알 수 있다.

❸ 두 개의 군집이 형성되었다면, 각각의 군집의 평균값을 구해 두 군집의 중심점을 찾을 수 있다. 해당 위도, 경도의 위치가 각 군집의 관리 대여소 설치 위치가 된다.

프로그램

```
1 # 그룹별(군집별) 평균값을 구해 중심점 찾기
2 df1.groupby('group').mean()
```

실행 결과

group	위도	경도
0	37.534555	126.903564
1	37.557585	127.064734

설명 group 0과 group 1로 구분된 데이터들의 각각의 위도, 경도의 평균값을 구하였다. group 0의 데이터들의 위도 평균값은 약 37.53, 경도 평균값은 약 126.90이었다. 또한 group 1의 데이터들의 위도 평균값은 약 37.55, 경도 평균값은 약 127.06이었다. 즉, 이 두 위치가 두 그룹을 관리하는 관리소의 위치로 적절하다는 결론을 낼 수 있다.

❹ 두 군집으로 분류된 데이터들의 평균값, 즉 관리소의 위치를 직접 산점도에 시각화해 확인해 보자. 이를 위해 ❸에서 구한 평균값을 df2라는 새로운 변수에 저장한 다. 그 후 맷플롯립(matplotlib)을 이용해 다시 산점도를 표시하되 두 개의 군집과 df2에 대해 서로 다른 색 지정 옵션을 추가해 보자.

프로그램

```
1 # 그룹들의 평균값을 df2라는 이름의 변수로 저장하기
2 df2 = df1.groupby('group').mean()
```

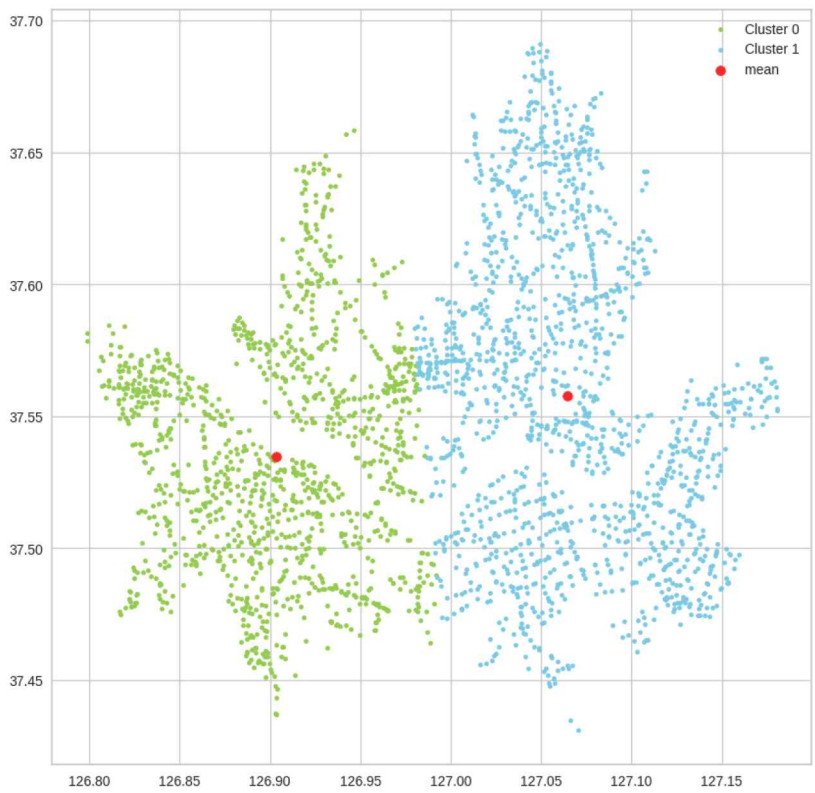
프로그램

```
# 군집 결과, 평균값을 색으로 구분해 산점도로 시각화하기
plt.figure(fig size = (10, 10))

# group 0의 대여소 위치를 연한 녹색(yellowgreen)으로 시각화하기
plt.scatter(df1[df1['group'] == 0]['경도'], df1[df1['group'] == 0]['위도'], s = 10, c = 'yellowgreen', label = 'Cluster 0')
# group 1의 대여소 위치를 하늘색(skyblue)으로 시각화하기
plt.scatter(df1[df1['group'] == 1]['경도'], df1[df1['group'] == 1]['위도'], s = 10, c = 'skyblue', label = 'Cluster 1')
# 두 군집의 평균값이 저장된 df2를 빨간색(red)으로 시각화하기
plt.scatter(df2['경도'], df2['위도'], s = 50, c = 'red', label = 'mean')

# 산점도를 표시하기
plt.legend()
plt.show()
```

실행 결과



설명 모든 데이터가 가까운 것들끼리 2개의 군집으로 나뉘어 연한 녹색, 하늘색으로 구분되어 표시되었다. 또한 각 군집의 중심점을 빨간색으로 표현해 한눈에 보기 쉽게 산점도에 표현하였다. 관리소 위치는 붉은 점 2개의 위도, 경도에 따라 설치하면 된다는 결론을 내릴 수 있다.

5단계 성능 평가

군집은 비지도학습이기 때문에 결과와 정답을 비교할 수 없어 실루엣 계수를 이용해 성능을 평가한다. 실루엣 계수는 개별 데이터가 할당된 군집 내 데이터와 얼마나 가깝게 군집되어 있는지, 그리고 다른 군집에 있는 데이터와는 얼마나 멀리 분리되어 있는지를 수치로 나타낸다. 군집이 잘 분리되었다는 것은 동일한 군집 내에서의 데이터는 서로 가깝게 위치해 있으며 다른 군집과의 거리는 멀음을 의미한다. 실루엣 계수 값이 크다면 성능이 좋은 것으로 해석한다.

실루엣 계수는 −1에서 1 사이의 값을 가지며 1에 가까울수록 근처 군집과 멀리 떨어져 있음을, 0에 가까울수록 근처 군집과 가까움을 의미한다. −(마이너스)이면 아예 다른 군집에 데이터가 할당됐음을 의미한다.

프로그램

```
# 사이킷런에서 실루엣 점수를 계산하는 함수 불러오기
from sklearn.metrics import silhouette_score

# 실루엣 점수 계산하기
silhouette_avg = silhouette_score(df1, df1['group'])
print("실루엣 점수:", silhouette_avg)
```

실행 결과

```
실루엣 점수: 0.9143281769153793
```

소단원 1분 요약

❶ 기계학습을 이용해 문제를 해결하기 위해서는 문제 상태를 정확히 파악하고 정의할 수 있어야 한다. 정의한 문제가 기계학습을 통해 해결할 수 있는 문제인지도 확인해야 한다.

❷ 기계학습을 이용해 문제를 해결하는 과정: [1단계] 문제 정의 → [2단계] 데이터 탐색 및 전처리 → [3단계] 기계학습 유형과 알고리즘 선정 → [4단계] 기계학습을 통한 모델 생성 → [5단계] 성능 평가

소단원 자기 평가

평가 항목	평가 기준		
	잘함	보통	노력
1. [과정·기능] 기계학습으로 해결할 수 있는 문제의 유형을 비교할 수 있다.			
2. [가치·태도] 사회문제를 해결하기 위해 기계학습을 적극적으로 적용하는 자세를 올바르게 인식할 수 있다.			