IT 파이썬. 딥러닝

인공지능 모델학습-예측

강사 이수현

목차

머신러닝 문제해결 시험점수 예측

시험 점수를 예측하다 작업 순서

- **연속적인 수치 예측**

문제 정의하기 시험 점수 예측

데이터 불러오기 학생 점수 데이터 셋

데이터 탐색 및 시각화하기 속성 탐색
속성 간 상관관계 분석하기 데이터 나누기

모델 생성하기 선형 회귀 모델

모델 학습하기

모델 평가 및 예측하기

문제 정의하기

python™

문제 상황 이야기하기

인공지능은 학생이 선택한 과목 수, 학생의 하루 평균 공부한 시간만으로 어떻게 학생의 시험 점수를 예측할 수 있을까요?

1. 내가 선택한 과목이 5개이고, 하루 평균 공부한 시간이 4시간이면 시험 점수가 얼마나 나올까요?

문제 정의하기

문제 해결에 필요한 정보

문제 해결 과정에서 필요한 정보를 미리 살펴봅시다.

- 1. 이 활동에 필요한 데이터 셋은 무엇이고, 이 데이터 셋은 어디에서
  다운로드할 수 있나요?
- 데이터 셋은 학생 시험 점수 데이터 셋으로, 캐글에서 다운로드할 수 있습니다.

문제 정의하기

문제 해결에 필요한 정보

데이터 셋은 학생 시험 점수 데이터 셋으로, 캐글에서 다운로드 할 수 있습니다.

- 학생이 선택한 과목 수와 공부 시간으로 시험 성적을 예측할 수 있도록 데이터 탐색 후 전처리 과정이 필요합니다. 학생 시험 점수 속성에 영향을 미치는 속성을 찾기 위해 각 속성 간 상관관계를 분석하여 시각화하기, 학습 및 평가를 위한 훈련 데이터와 테스트 데이터 나누기 등을 합니다.

문제 정의하기

3. 모델 생성에서 사용할 모델을 찾아볼까요?

- 여기서는 학생 시험 점수 예측을 위해 선형 회귀 모델을 사용합니다.

4. 선형 회귀 모델 성능을 나타내는 평가 지표는 무엇일까요?

- 테스트 데이터의 예측값과 실젯값을 비교하여 얼마나 차이가 나는지를 평가하는 지표로 결정계수를 산출합니다.

문제 정의하기

5. 분류 모델 성능과 회귀 모델 성능을 평가하는 방법은 어떻게 다른가요?

분류 모델 성능은 예측값과 실젯값을 비교해서 몇 개의 데이터를 맞췄는지를 평가하고, 회귀 모델 성능은 예측값과 실젯값을 비교해서 오차가 얼마나 나는지를 평가합니다.

데이터 불러오기

데이터 셋 소개하기

학생 점수 데이터 셋은 선형 회귀를 포함하여 기본적인 회귀 문제를 학습하기에 매우 단순한 데이터이다.

그러나 학생의 점수를 한정된 속성으로 예측하기 때문에 실제로 이 모델을 사용하기에는 한계가 있다.

학생 점수 데이터 셋

- 데이터: 총 1000개의 데이터
- 속성: 코스 수(number_courses)
  하루 평균 공부한 시간(time_study)
  학생의 점수(Marks)

데이터 불러오기

python™

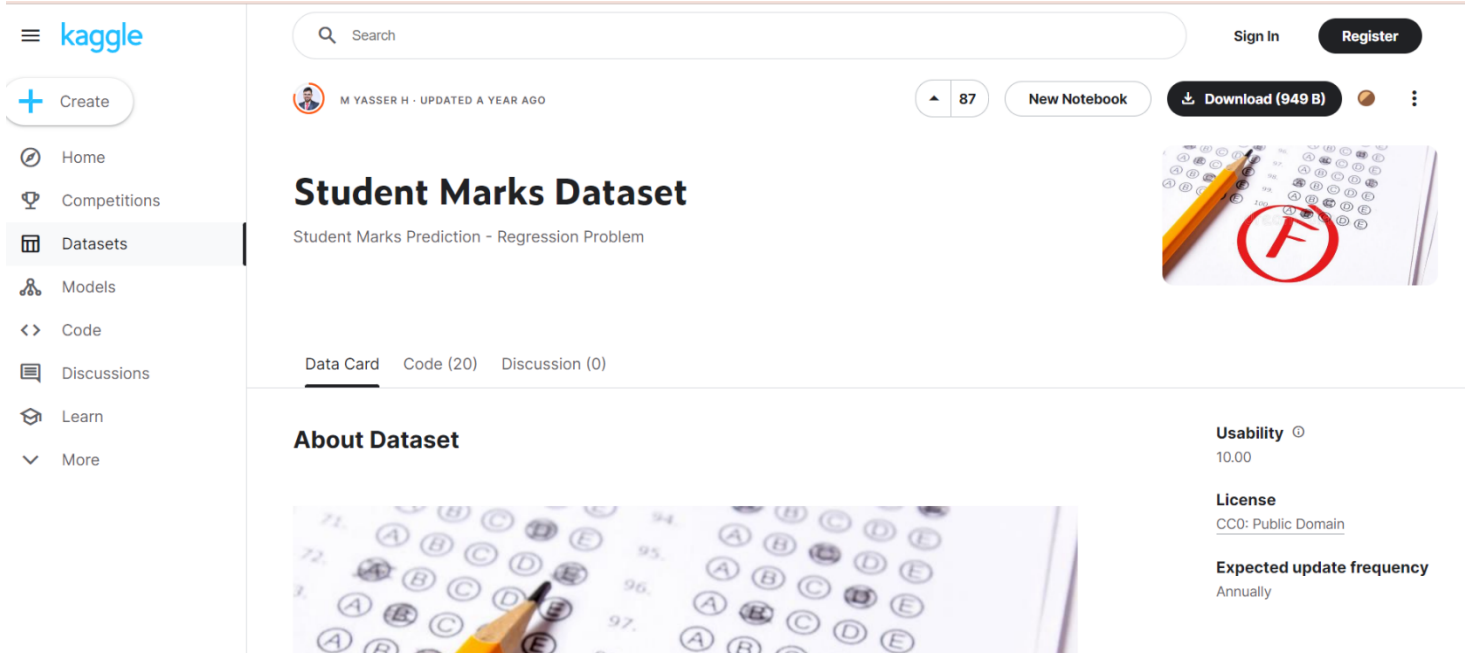
학생 점수 데이터 속성

1	number_courses	time_study	Marks
2	3	4.508	19.202
3	4	0.096	7.734
4	4	3.133	13.811
5	6	7.909	53.018
97	6	3.561	19.128
98	3	0.301	5.609
99	4	7.163	41.444
100	7	0.309	12.027
101	3	6.335	32.357

데이터 불러오기

데이터 셋 불러오기

- 학생 점수 데이터 셋은 캐글(kaggle.com)에서 수집할 수 있다.
  캐글 검색창에서 'Student Marks Dataset'을 검색하여 아래의 데이터 셋을 다운로드 한다. 다운로드한 파일의 압축을 풀면'Student_Marks.csv'를 확인할 수 있다.



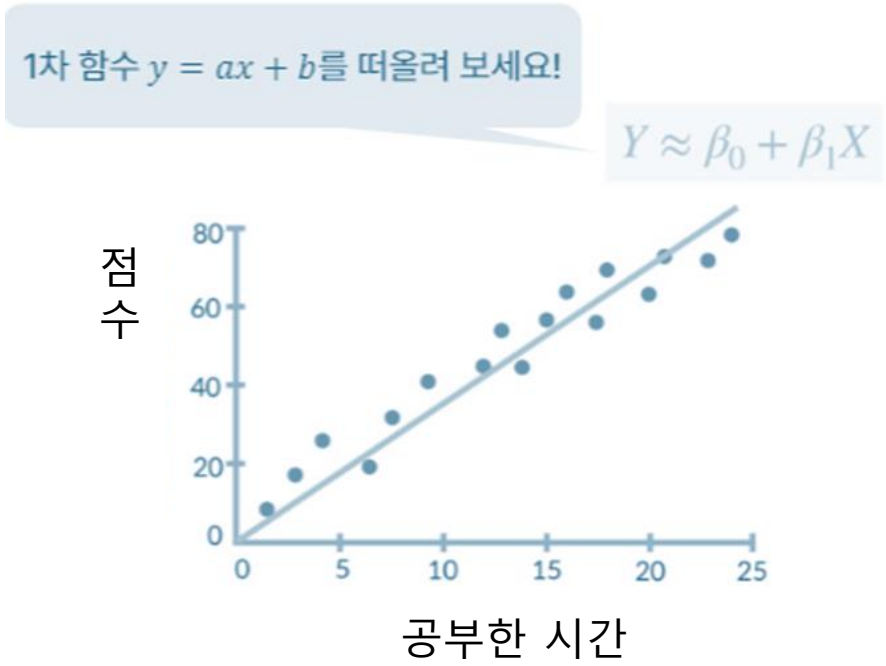
참고: https://www.kaggle.com/datasets/yasserh/student-marks-dataset

회귀 분석

- **주어진 데이터**
  - X:공부한 시간, Y: 점수
  - 가정

$$Y \approx \beta_0 + \beta_1 X$$

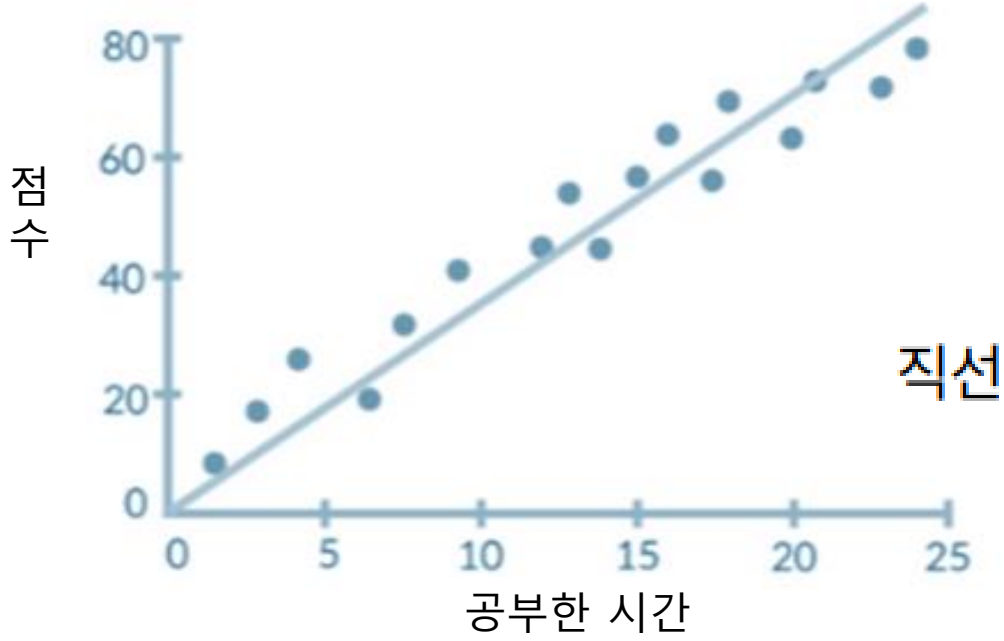
적절한 β_0 , β_1 값을 찾자
> 데이터를 가장 잘 설명하는 모델



회귀 분석은 데이터를 가장 잘 설명하는 모델을 찾아
입력값에 따른 미래 결과값을 예측하는 알고리즘

회귀 분석의 목표

$$Y \approx \beta_0 + \beta_1 X$$

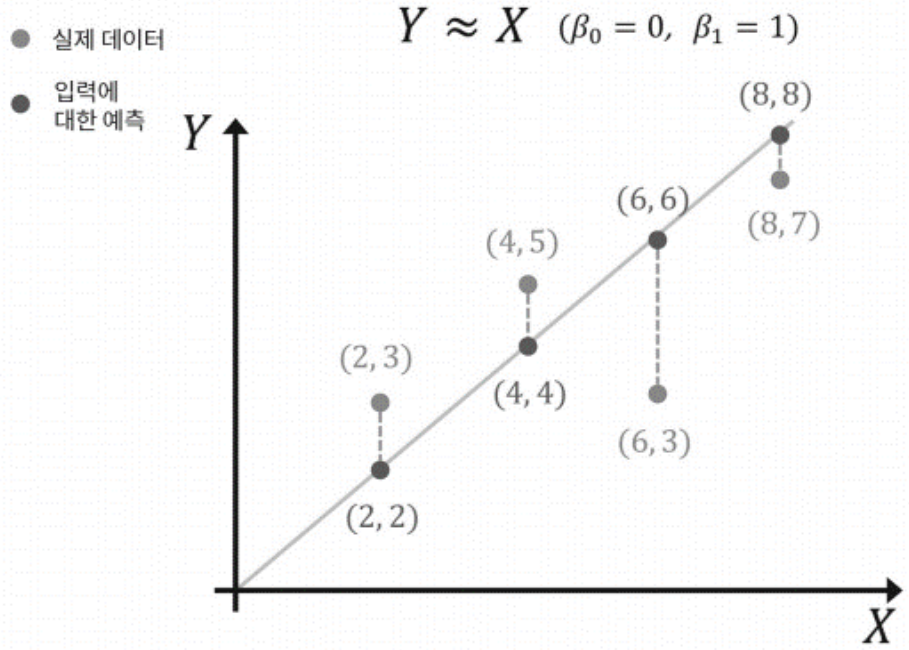


직선을 구성하는 β_0(y절편)과 β_1(기울기)를 구해야 합니다.

- 완벽한 예측을 하는 것은 불가능하기 때문에 최대한 근사하는 선을 찾아야 함.
- 즉, 회귀 분석의 목표는 각 데이터의 실제 값과 모델이 예측하는 값이 차이를 최소화하는 선을 찾는 것

단순 선형 회귀

데이터를 잘 설명한다는 것은



입력값	예측값	실제값	실제값 – 예측값	(실제값 – 예측값)^2
2	2	3	1	1
4	4	5	1	1
6	6	3	-3	9
8	8	7	-1	1
		합계	-2	12

실제 값과 예측 값의 차이를 구해봅시다.

음수와 양수가 서로 상쇄되기 때문에 (실제 값 – 예측 값)의 합은 의미가 없으며,

부호를 없애기 위해 제곱해야 합니다.

단순선형회귀

Loss 함수 이해하기

$$Y \approx \beta_0 + \beta_1 X$$

i번째 데이터 (x^i, y^i) 에 대해,

입력 값 : x^i
실제 값 : y^i
예측 값 : $\beta_0 x^i + \beta_1$

Loss 함수 : $\frac{1}{N}\sum_i^N \left(y^{(i)} - \left(\beta_0 + \beta_1 x^{(i)}\right)\right)^2$

Loss 함수는 실제 값과 예측 값 차이의 제곱의 합을 의미하며,

Loss 함수가 작을 수록 좋은 모델입니다.

단순 선형 회귀

Loss 함수 줄이기

< Loss 함수의 크기를 작게 하는 절편과 기울기를 찾는 방법 >

1. Gradient Descent (경사 하강법)

2. Normal Equation (least squares)

3. Brute force search

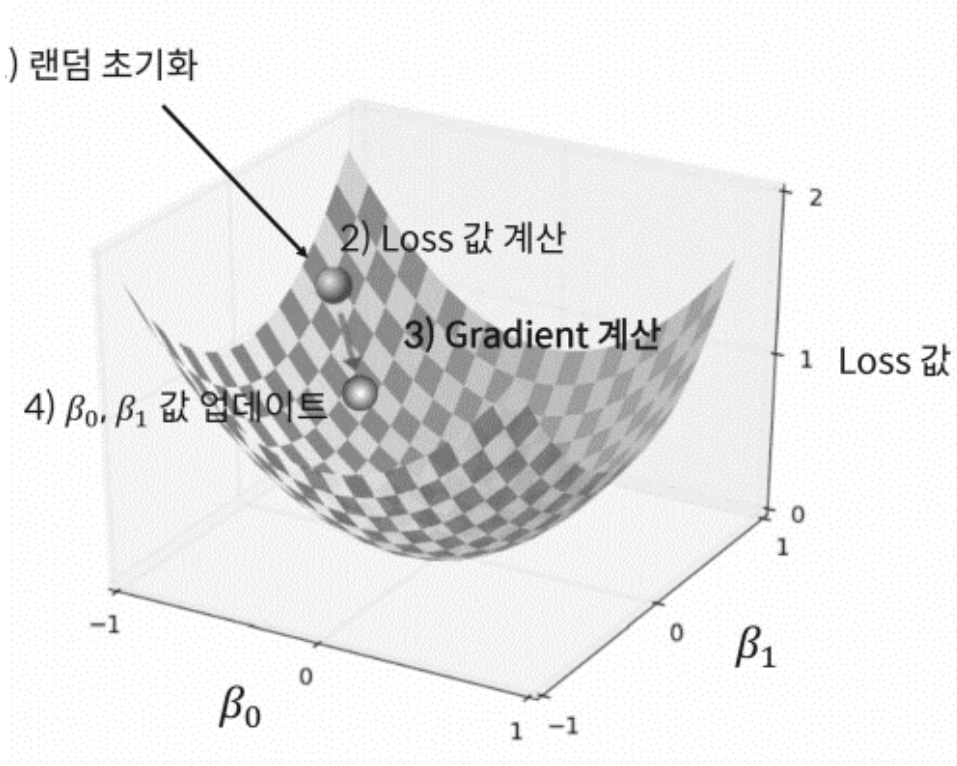
...

Loss 함수의 크기를 작게 하는 절편과 기울기를 찾는 방법은 다양합니다.

그 중, 이번 과목에서는 경사 하강법에 대해 배워봅시다.

단순 선형 회귀

경사 하강법



① 모델의 초기 파라미터(절편과 기울기) 값을 설정합니다.

② 손실 함수를 계산합니다.

③ 손실 함수의 기울기를 계산합니다.

- 기울기는 파라미터가 손실 함수에 미치는 영향을 나타내는 값으로, 파라미터가 이 값의 반대 방향으로 이동할 때 손실 함수가 감소합니다.

④ 절편과 기울기 값을 업데이트 합니다.

⑤ 새로운 절편과 기울기 값에서 손실 함수를 다시 계산합니다.

⑥ 일정한 조건 (예를 들어, 일정한 반복 횟수나 일정한 수준의 손실 함수 값 도달)을 만족할 때까지 3~5 단계를 반복합니다.

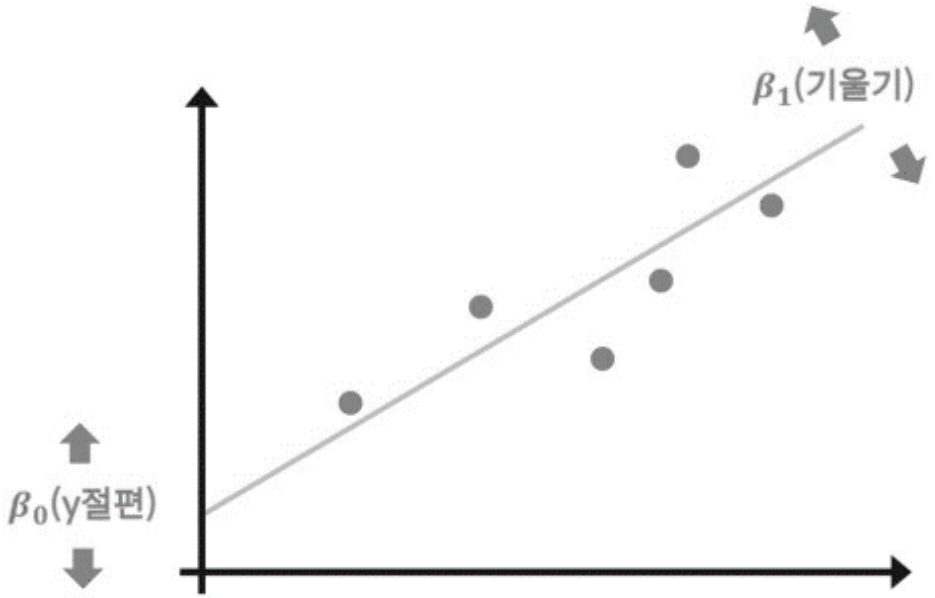
단순 선형 회귀

단순 선형 회귀 특징

- 가장 기초적이나 많이 사용되는 알고리즘입니다.
- 입력 값이 1개인 경우에만 적용 가능합니다.
- 입력 값과 결과 값의 관계를 알아보는 데 용이합니다.
- 입력 값이 결과 값에 얼마나 영향을 미치는지 알 수 있습니다.
- 두 변수 간의 관계를 직관적으로 해석하고자 하는 경우 활용됩니다.

단순 선형 회귀

Loss 함수 이해하기



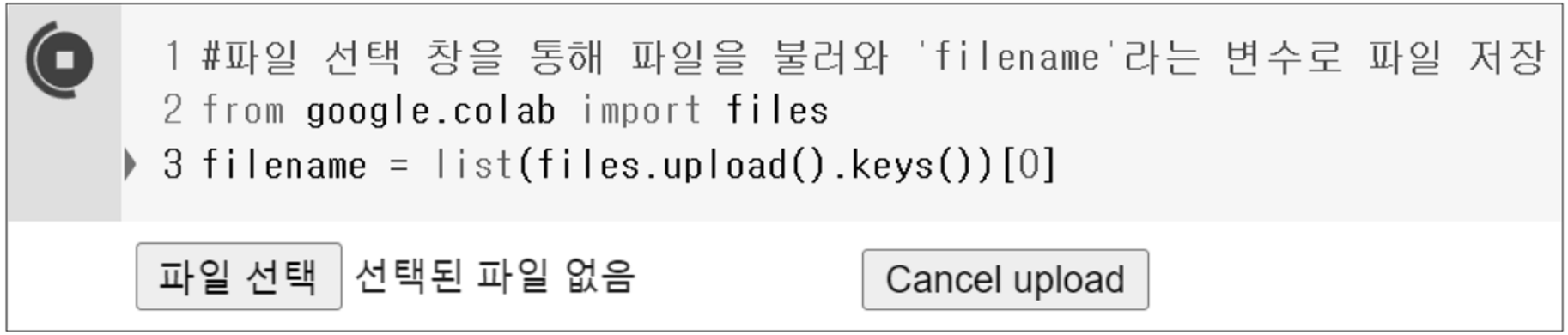
Loss 함수의 크기를 작게 만들기 위해서는 y절편과 기울기 값을 적절히 조절해야 합니다.

데이터 불러오기

데이터 셋 업로드하기

다운로드한 'Student_Marks.csv' 파일을 구글 드라이브에 업로드

```
1 #파일 선택 창을 통해 파일을 불러와 'filename'라는 변수로 파일 저장
2 from google.colab import files
3 filename = list(files.upload().keys())[0]
```



기본 자료형은 이렇게 사용해요!

파일 읽어 들이기

- 판다스 라이브러리를 불러온 다음, 판다스의 read_csv( )로 업로드된 파일을 읽어 들여 데이터프레임 객체 변수에 저장하기

```
데이터프레임 객체 = 판다스 객체.read_csv('파일명.csv')
# '파일명' 대신 파일을 저장한 변수로 파일 제어
```

데이터 불러오기

파일 읽어 들이기

- 판다스 라이브러리를 불러온 다음, 판다스의 read_csv( )로 업로드된 파일을 읽어 들여 데이터프레임 객체 변수에 저장하기

```
데이터프레임 객체 = 판다스 객체.read_csv('파일명.csv')

# '파일명' 대신 파일을 저장한 변수로 파일 제어
```

데이터 불러오기

파일 읽어들이기 • 불러온 학생 점수 데이터 셋을 student_data라는 이름으로 사용하기

```
1 import pandas as pd
2 student_data = pd.read_csv(filename)
3 student_data
```

	number_courses	time_study	Marks
0	3	4.508	19.202
1	4	0.096	7.734
2	4	3.133	13.811
3	6	7.909	53.018
4	8	7.811	55.299
...	...	...	...
95	6	3.561	19.128
96	3	0.301	5.609
97	4	7.163	41.444
98	7	0.309	12.027
99	3	6.335	32.357

100 rows × 3 columns

데이터 불러오기

데이터 셋의 구조

머신러닝에서 다루는 데이터 셋의 구조에서 행, 열은 다양한 이름으로 불린다.

행(row): 데이터 포인트(data point), 오브젝트(object), 레코드(record)라는 이름으로 사용

열(column): 변수(variable), 속성(attribute), 특성(feature)이라는 이름으로 생성,

독립 변수, 종속 변수로 나뉘는데, 종속 변수는 레이블(label), 타깃(target), 클래스(class)라는 이름으로도 사용

데이터 불러오기

데이터 셋의 구조

열, 변수, 속성, 특성

독립 변수

	number_courses	time_study	Marks
0	3	4.508	19.202
1	4	0.096	7.734
2	4	3.133	13.811
3	6	7.909	53.018

행, 데이터 포인트, 오브젝트, 레코드

종속 변수, 레이블, 타깃, 클래스

[학생 점수 데이터 셋 예시]

데이터 불러오기

데이터 살펴보기

- 학생 정보 데이터 셋에는 몇 개의 데이터와 속성이 있는지, 어떤 속성이 있는지 살펴보기
- 판다스 라이브러리의 info( ) 메소드를 사용하기

데이터프레임 객체.info( )

info( ) 메소드를 통해 데이터 개수, 속성 개수, 속성명, 결측치, 속성의 데이터 타입 등 확인

데이터 불러오기

데이터 확인하기

```
1 student_data.info( )
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   number_courses  100 non-null    int64
 1   time_study      100 non-null    float64
 2   Marks           100 non-null    float64
dtypes: float64(2), int64(1)
memory usage: 2.5 KB
```

속성명	타입	설명
number_courses	정수형(int64)	학생이 선택한 과목 수
time_study	실수형(float64)	학생이 하루 평균 공부한 시간
Marks	실수형(float64)	학생의 시험 점수

데이터 탐색 및 전처리하기

판다스 라이브러리의 head( ) 메소드와 tail( ) 메소드

- **head( ) 메소드 : 데이터의 앞부분 확인 가능**
- **tail( ) 메소드 : 데이터의 뒷부분 확인 가능**

소괄호( ) 안에 숫자를 넣지 않으면 앞부분 또는 뒷부분의 5행까지 내용을 볼 수 있고, 숫자를 넣으면 숫자만큼 원하는 행을 볼 수 있음.

	number_courses	time_study	Marks
95	6	3.561	19.128
96	3	0.301	5.609
97	4	7.163	41.444
98	7	0.309	12.027
99	3	6.335	32.357

```
1  데이터프레임 객체.tail( ) # 데이터 하위 5행 출력
```

데이터 불러오기

데이터 통계치 살펴보기

- describe( ) 메소드를 사용하여 데이터들의 통계량(개수, 평균, 표준편차, 최솟값, 최댓값, 4분위수)을 파악하기
- 수칫값을 갖는 속성에 대해서만 통계치 출력, 결측치는 제외

데이터프레임 객체.describe( )

describe( ) 메소드를 통해 데이터 개수, 평균, 표준편차, 최솟값, 최댓값, 4분위수 파악

데이터 불러오기

데이터 통계치 살펴보기

- 3개 속성의 통계량 출력하기

```
1 student_data.describe()
```

	number_courses	time_study	Marks
count	100.000000	100.000000	100.000000
mean	5.290000	4.077140	24.417690
std	1.799523	2.372914	14.326199
min	3.000000	0.096000	5.609000
25%	4.000000	2.058500	12.633000
50%	5.000000	4.022000	20.059500
75%	7.000000	6.179250	36.676250
max	8.000000	7.957000	55.299000

데이터 불러오기

데이터 통계치 살펴보기

- 학생이 선택한 과목 수(number_courses)는 최소 3개, 최대 8개 과목이며 평균 5.29개의 과목을 선택한 것을 알 수 있음.
- 학생이 하루 평균 공부한 시간(time_study)은 최소 0.096시간(5.76분), 최대 7.957시간(약7시간 57분)이며 하루 평균 4시간 정도 공부함.
- 학생의 시험 점수(Marks)는 최소 5.6점, 최대 55.299점.
- 만약 이 점수가 100점 만점에 대한 점수라면 시험 문제가 매우 어려운 것으로 보이지만, 이 데이터에서는 60점 만점이라고도 유추 가능.

데이터 불러오기

데이터 시각화하기

학생의 시험 점수를 예측하는 것이므로 학생의 시험 점수(Marks) 속성에 영향을 미치는 속성을 찾아야 함
이때 어느 속성이 시험 점수와 관련이 깊은지 확인하기 위해 상관관계 분석을 한 후 히트맵과 산점도로 시각화 해 봅니다.

데이터 불러오기

상관관계 분석

corr( ) 메소드를 사용하여 각 속성 간의 상관관계 파악하기

```
1 student_corr = student_data.corr( ) # 상관관계(correlation) 분석
2 student_corr
```

	number_courses	time_study	Marks
number_courses	1.000000	0.204844	0.417335
time_study	0.204844	1.000000	0.942254
Marks	0.417335	0.942254	1.000000

학생 점수와 각 속성 간의 상관 계수이며 1에 가까울수록 상관관계가 높음을 의미합니다.

데이터 탐색 및 전처리하기

상관관계 분석

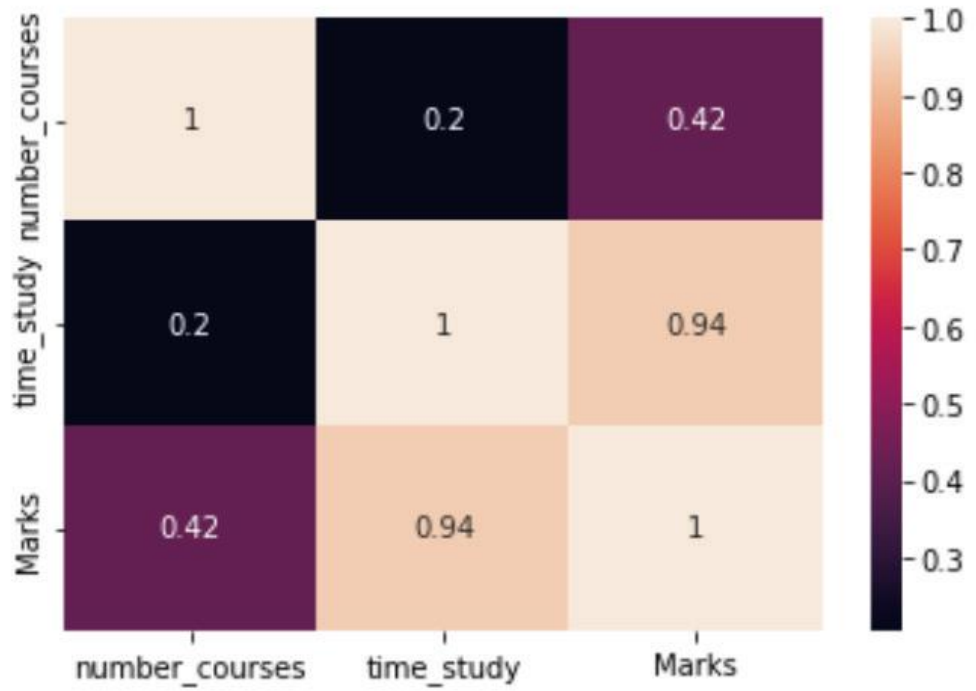
상관관계를 나타낸 위의 결과를 시본(seborn) 라이브러리를 사용해 히트맵으로 시각화하기
히트맵을 사용하면 속성 간의 관계를 색으로 나타내어 상관관계를 쉽게 파악 가능

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 sns.heatmap(student_corr, annot = True) # 히트맵에 값 포함하여 그리기
4 plt.show( ) # 화면에 차트나 플롯(plot)을 출력할 때 사용하며 코랩에서는 생략 가능
```

데이터 불러오기

상관관계 분석

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 sns.heatmap(student_corr, annot = True)
4 plt.show( )
```



데이터 불러오기

상관관계 분석

학생의 시험 점수(Marks)와 학생이 선택한 과목 수(number_courses)의 상관관계는 0.42로 관련이 보통 수준인 반면, 학생의 시험 점수와 학생이 하루 평균 공부한 시간(time_study)의 상관관계는 0.94로 매우 높은 것을 알 수 있다.

데이터 불러오기

산점도로 표현

맷플롯립 라이브러리의 scatterplot( )을 사용하여 산점도로
여러 개의 그래프 표현하기

파이플롯 객체.subplots(nrows = 행의 수, ncols = 열의 수, 그래프 사이즈)

여러 개의 그래프 그리기

시본 객체.scatterplot(데이터프레임 객체, x축명, y축명, 표현할 그래프 인덱스)

데이터 불러오기

산점도로 표현

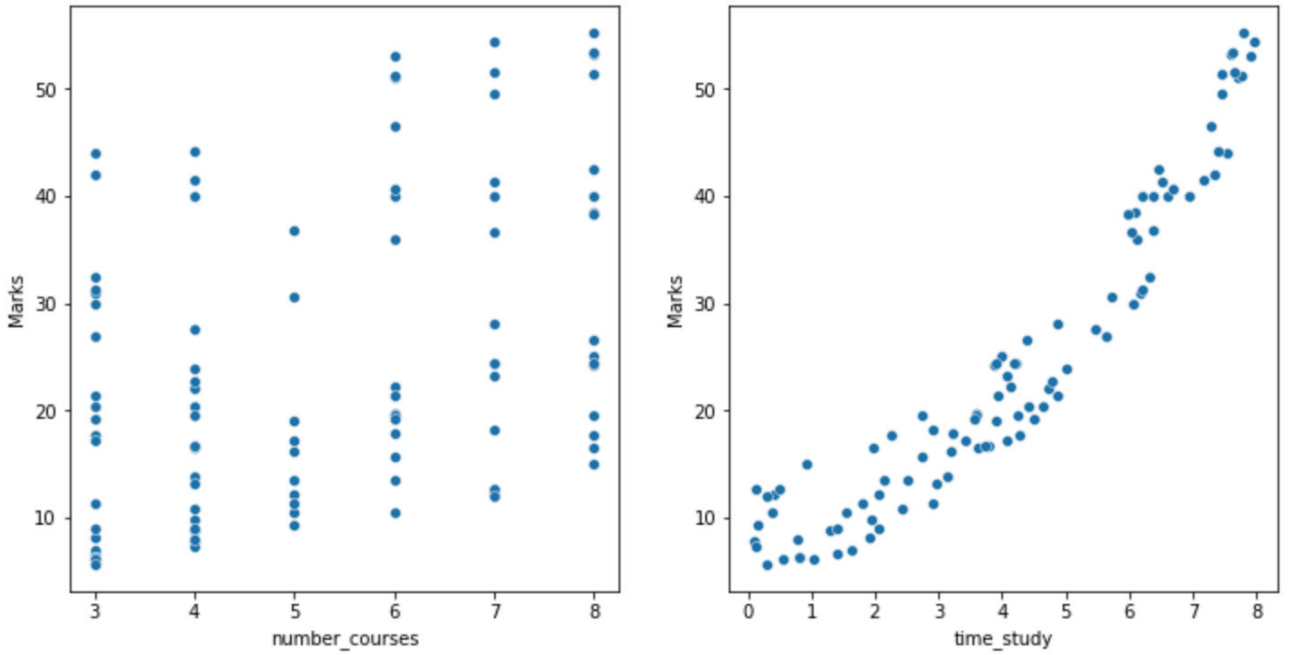
- y축에 학생의 시험 점수 설정하기
- x축에 학생이 선택한 과목 수, 학생이 하루 평균 공부한 시간을 각각 설정하여 2개의 산점도 그려보기

```
1 # 2개 그래프가 양옆으로 들어갈 사이즈
2 fig, ax = plt.subplots(ncols = 2, figsize = (12, 6))
3 sns.scatterplot(data = student_data, x = 'number_courses', y = 'Marks', ax = ax[0])
4 sns.scatterplot(data = student_data, x = 'time_study', y = 'Marks', ax = ax[1])
5 plt.show( ) # fig.show( )와 동일한 결과
```

데이터 불러오기

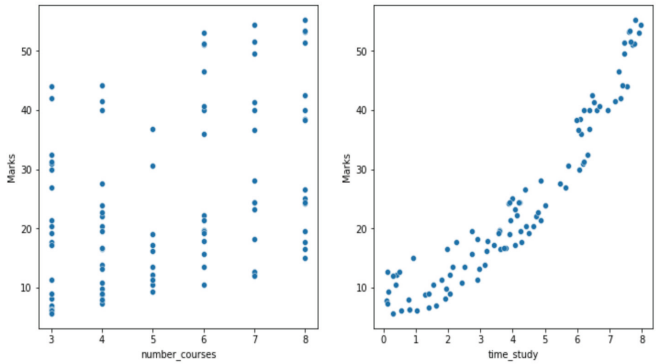
산점도로 표현

```
1 # 2개 그래프가 양옆으로 들어갈 사이즈
2 fig, ax = plt.subplots(ncols = 2, figsize = (12, 6))
3 sns.scatterplot(data = student_data, x = 'number_courses', y = 'Marks', ax = ax[0])
4 sns.scatterplot(data = student_data, x = 'time_study', y = 'Marks', ax = ax[1])
5 plt.show( ) # fig.show( )와 동일한 결과
```



데이터 불러오기

산점도로 표현



- 왼쪽 그래프에서 학생의 시험 점수(Marks)와 학생이 선택한 과목 수(number_courses)는 상관관계가 거의 없다는 것을 알 수 있고.
- 오른쪽 그래프에서 학생의 시험 점수(Marks)와 학생이 하루 평균 공부한 시간(time_study)은 양의 상관관계가 매우 높다는 것을 알 수 있습니다.

데이터 탐색 및 전처리하기

산점관계와 산점도 표현 결과

상관관계 분석 결과를 히트맵과 산점도로 표현해 본 결과 학생이 선택한 과목 수보다 학생이 하루 평균 공부한 시간이 학생의 시험 점수와 상관관계가 높다는 것을 다시 한번 확인

데이터 탐색 및 전처리하기

훈련데이터와 테스트데이터 나누기

인공지능 모델을 학습시키기 위해서는 사이킷런 라이브러리를 사용하여 학생 점수 데이터 셋을 훈련 데이터와 테스트 데이터로 나누어야 한다.

이번 활동에서는 훈련 데이터와 테스트 데이터를 8:2의 비율로 나눈다.

데이터 탐색 및 전처리하기

훈련데이터와 테스트데이터 나누기

```
1 from sklearn.model_selection import train_test_split
2 x = student_data.drop('Marks', axis = 1) # Marks열 삭제한 나머지 열
3 y = student_data['Marks'] # Marks열 선택
4 X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.2,
5                                                     random_state = 42)
6 print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
```

```
(80, 2) (20, 2) (80,) (20,)
```

train_test_split( )은 실행할 때마다 랜덤으로 훈련 데이터와 테스트 데이터를 나누기 때문에 항상 동일한 데이터로 분리하기 위해 random_state를 특정 숫자(예: random_state = 42)로 지정합니다.

데이터 탐색 및 전처리하기

훈련데이터와 테스트데이터 나누기

전체 100개 데이터 중 훈련 데이터는 80개, 테스트 데이터는 20개로 이때의 **독립 변수는 2개**이고, 훈련 데이터 레이블은 80개, 테스트 데이터 레이블은 20개로 이때의 **종속 변수는 1개**로 나누어졌음을 확인할 수 있다.

데이터 탐색 및 전처리하기

훈련데이터와 테스트데이터 나누기

	number_courses	time_study	Marks
0	3	4.508	19.202
1	4	0.096	7.734
2	4	3.133	13.811
3	6	7.909	53.018
4	8	7.811	55.299
...	...	...	...
95	6	3.561	19.128
96	3	0.301	5.609
97	4	7.163	41.444
98	7	0.309	12.027
99	3	6.335	32.357

100 rows × 3 columns

훈련 데이터와 테스트 데이터 정리!

①, ③은 학습할 때 필요

① X_train (80, 2)	③ Y_train (80,)
② X_test (20, 2)	④ Y_test (20,)

②, ④는 테스트할 때 필요

모델 생성 및 학습하기

- 회귀를 위한 머신러닝 모델에는 **선형 회귀(Linear Regression)**, 최근접 이웃 회귀(k-NN Regression), 랜덤 포레스트 회귀(RandomForest Regression) 등 있음.
- 서로 학습하는 방법은 다르지만 프로그램을 작성하는 방법은 거의 동일.
- 이번 활동에서는시험 점수를 예측하는 데 선형 회귀 모델을 사용할 것이므로 선형 회귀 모델에 관해서만 알아보고, 나머지 회귀 모델은 결과를 서로 비교해 보는 실습 정도만 진행함.

모델 생성 및 학습하기

모델 생성하기

- 선형 회귀 모델을 생성하기
- 사이킷런을 이용하여 라이브러리 불러오기
- LinearRegression( ) 함수를 통해 선형 회귀 모델 생성

```
1 from sklearn.linear_model import LinearRegression
2 lr_model = LinearRegression( ) # Linear Regression 모델 생성
```

모델 생성 및 학습하기

모델 학습

- 모델 학습에 X_train(훈련 데이터), Y_train(훈련 데이터 레이블)을 사용하는 것은 분류나 회귀 모델 모두 동일하게 적용
- fit( ) 함수 사용

```
1 from sklearn.linear_model import LinearRegression
2 lr_model = LinearRegression( )
3 lr_model.fit(X_train, Y_train) # Linear Regression 모델 학습
```

```
LinearRegression( )
```

모델 생성 및 학습하기

모델 생성

- 모델 학습에 X_train(훈련 데이터), Y_train(훈련 데이터 레이블)을 사용하는 것은 분류나 회귀 모델 모두 동일하게 적용
- fit( ) 함수 사용

```
1 from sklearn.linear_model import LinearRegression
2 lr_model = LinearRegression( )
3 lr_model.fit(X_train, Y_train) # Linear Regression 모델 학습
```

```
LinearRegression( )
```

모델 생성 및 학습하기

선형 회귀(Linear Regression) 모델

- 선형 회귀 모델은 수치형 데이터 Y에 대하여 Y에 영향을 미치는 데이터 X와의 관계를 평균값으로 예측하기 위해 만드는 모델
- X가 입력되었을 때 Y도출

$$Y = aX + b$$

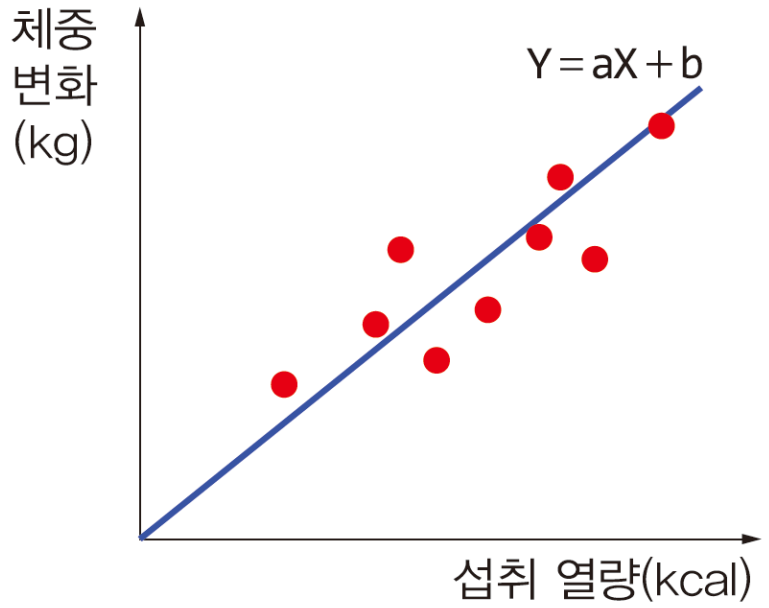
모델 생성 및 학습하기

선형 회귀(Linear Regression) 모델

섭취 열량에 따른 체중 변화를 예측할 때 선형 회귀 모델을 적용하면 영향을 미치는 섭취 열량 데이터 X를 독립 변수라고 하고, 영향을 받는 체중 변화 데이터 Y를 종속 변수라고 함.

- **단순 선형 회귀 : 독립 변수의 개수가 1개**
- **다중 선형 회귀 : 독립 변수의 개수가 2개 이상**

섭취 열량, 운동 시간에 따른 체중 변화는 다중 선형 회귀 모델로 예측할 수 있다.



회귀에서 성능평가

MSE (Mean Squared Error)	예측값과 실젯값 차이를 제곱하여 평균 낸 값으로, 0에 가까울수록 예측값과 실제값의 차이가 없으므로 성능이 우수함.	$(MSE) = \frac{1}{n}\sum_{i=1}^{n}(H(x_i) - y_i)^2$
RMSE (Root Mean Squared Error)	MSE를 Root처리한 값으로 0에 가까울수록 성능이 우수함.	$(\mathrm{SE}) = \sqrt{\mathrm{MSE}}$
MAE (Mean Absolute Error)	예측값과 실젯값 차이의 절댓값 평균 낸 값으로, 0에 가까울수록 성능이 우수함.	$(MAE) = \frac{1}{n}\sum_{i=1}^{n} \mid H(x_i) - y_i \mid$
R2 (R Squared, R2, 결정계수)	1에 가까울수록 성능이 우수함.	$(R^2) = \frac{\sum_{i=1}^{n}(H(x_i) - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$

0에 가까울 수록 정확도 높음

모델 생성 및 학습하기

모델 평가

- 이번 활동에서는 회귀 모델의 평가 지표로 **결정계수(R2)**와 **평균제곱오차(MSE: Mean Squared Error)**를 사용해 봅니다.

- R2는 0부터 1 사이의 값을 가지며 1에 가까울수록 오류가 없음을 의미한다.
- MSE는 머신러닝 모델에서 예측한 값이 실젯값과 얼마나 차이(오류)가 있는지를 평가하는 것으로 0에 가까울수록 좋은 모델이다.
- 분류 모델의 성능은 전체 데이터를 실젯값과 비교해서 몇 개의 데이터를 맞췄는지(이산적)를 평가한다.
- 반면, 회귀 모델의 성능은 전체 데이터를 실제 데이터와 비교해서 얼마나 차이가 있는지(연속적)를 평가한다.
- 결국 **머신러닝의 성능은 오차를 작게 만드는 것을 목표**로 한다.

모델 생성 및 학습하기

성능 평가하기

- 회귀 모델의 평가 지표로 결정계수와 평균제곱오차를 구하기
- 필요한 라이브러리는 사이킷런의 r2_score와 mean_squared_error 함수

```
1 from sklearn.metrics import r2_score
2 from sklearn.metrics import mean_squared_error
```

모델 생성 및 학습하기

성능 평가하기

- 회귀 모델을 평가하기 위해 predict( ) 메소드를 사용하여 학습을 완료한 lr_model에 X_test값을 넣으면 모델 예측
- 이때 모델이 예측한 결과는 lr_pred에 저장
- 모델이 예측한 값은 실젯값(Y_test)과 비교하여 성능 평가 지표인 R2와 MSE를 구하기
- 출력하는 형식으로 문자열 포맷을 위해 Python의 % 연산자나 format( ) 함수 대신 문자열 앞에 f 또는 F를 붙이는 형식(f-string)알고 사용하기

```
f '문자열'

# ' ' 안에는 출력할 문자열과 {변수:출력 형태} 형식 사용
```

모델 생성 및 학습하기

성능 평가하기

```
1 lr_pred = lr_model.predict(X_test)
2 print(f'r2_score: {r2_score(lr_pred, Y_test):.2f}')
3 print(f'Mean Squared Error: {mean_squared_error(lr_pred, Y_test):.2f}')
```

```
r2_score: 0.92
Mean Squared Error: 14.20
```

모델 생성 및 학습하기

성능 평가하기

Mean Squared Error의 14.20의 값만 보면 해당 모델이 얼마나 좋은지 또는 나쁜지 구분하기 힘들기 때문에 0~1 사이의 값을 가지는 R2를 사용하는 것이 좋다.

모델 생성 및 학습하기

시험 점수 예측

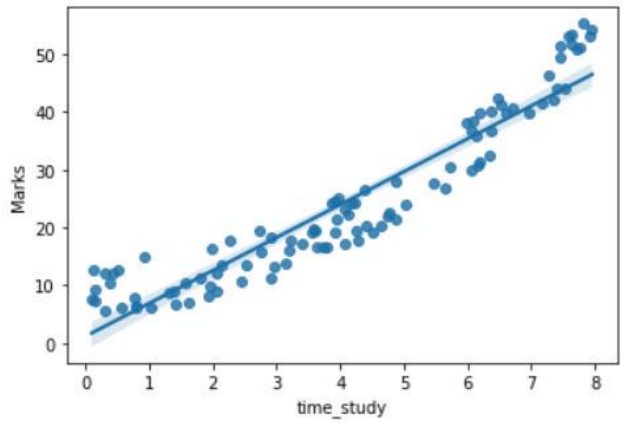
- 시본(seaborn)을 사용하면 **regplot( )** 함수로 간단하게 회귀선을 그리기
- 선형회귀선이란 데이터를 잘 설명하는 직선으로 즉, MSE값이 작은 선 의미
- MSE는 직선과 데이터(원 모양의 데이터)의 거리를 측정하여 구하기

모델 생성 및 학습하기

선형 회귀선 그리기

- 학생의 시험 점수에 영향을 미친 속성인 time_study와 Marks와의 관계를 회귀선으로 표현하기

```
1 # 회귀선 그리기
2 sns.regplot(x = student_data['time_study'],
3 y = student_data['Marks'])
4 plt.show( )
```



모델 생성 및 학습하기

선형 회귀선 그리기

선형 회귀식 구하기

학생의 시험 점수와 선택한 과목 수, 하루 평균 공부한 시간의 관계를 설명하는 선형 회귀식을 구하기

```
1 print(f'Number of Course: {lr_model.coef_[0]:.2f}') # 가중치(x1 계수)
2 print(f'Hours Studying per Day: {lr_model.coef_[1]:.2f}') # 가중치(x2 계수)
3 print(f'Intercept: {lr_model.intercept_:.2f}', end = '\n\n') # 절편
4 print(f'회귀식 y = {lr_model.coef_[0]:.2f} x1 + {lr_model.coef_[1]:.2f} x2 \
5                  { lr_model.intercept_:.2f}')
```

```
Number of Course: 1.87
Hours Studying per Day: 5.17
Intercept: -6.61
회귀식 y = 1.87 x1 + 5.17 x2 -6.61
```

모델 생성 및 학습하기

선형 회귀선 그리기

$y = 1.87x1 + 5.17x2 - 6.61$의 회귀식의 결과를 얻을 수 있다.

해석하면 y(학생의 시험 점수) = 1.87×number_courses + 5.17×time_study -6.61이다.

이를 통해 학생이 높은 시험 점수를 얻기 위해 5.17의 가중치가 있는 time_study가 매우 중요함을 알 수 있다.

모델 생성 및 학습하기

k-NN회귀

k-NN 회귀 모델은 KNeighborsRegressor( ) 함수를 이용하여 학습한다. 선형 회귀 모델과 동일하게 X_train(훈련 데이터)과 Y_train(훈련 데이터 레이블)이 필요하다.

```
1 from sklearn.neighbors import KNeighborsRegressor
2 knn_model = KNeighborsRegressor( )
3 knn_model.fit(X_train, Y_train)
```

```
KNeighborsRegressor( )
```

모델 생성 및 학습하기

k-NN회귀

학습된 knn_model을 이용해 X_test(테스트 데이터)를 넣은 값으로 예측한다. knn_pred(예측값)와 Y_test(실젯값)를 비교하여 평가 지표를 출력해 봅니다.

```
1 knn_pred = knn_model.predict(X_test)
2 print(f'r_score: {r2_score(knn_pred, Y_test):.2f}')
3 print(f'Mean Squared Error: {mean_squared_error(knn_pred, Y_test):.2f}')
```

```
r_score: 0.99
Mean Squared Error: 2.58
```

실행 결과 r_score의 값이 0.99로 선형 회귀 모델의 0.92보다 높기 때문에 k-NN 회귀 모델 성능이 더 좋다고 할 수 있어요.

모델 생성 및 학습하기

Random Forest 회귀

RandomForest 회귀 모델도 동일한 방법으로 학습과 평가를 합니다.

```
1 from sklearn.ensemble import RandomForestRegressor
2 rf_model = RandomForestRegressor(random_state = 42)
3 rf_model.fit(X_train, Y_train)
```

```
RandomForestRegressor(random_state = 42)
```

```
1 rf_pred = rf_model.predict(X_test)
2 print(f'r_score: {r2_score(rf_pred, Y_test):.2f}')
3 print(f'Mean Squared Error: {mean_squared_error(rf_pred, Y_test):.2f}')
```

```
r_score: 0.99
Mean Squared Error: 1.99
```

배운 내용 정리하기

python™

문제 정의하기

문제 상황 이해하기
해결할 문제: 학생의 시험 접수 예측

데이터 불러오기

캐글(kaggle)에서 학생 접수 데이터 셋 불러오기

데이터 탐색 및 전처리하기

데이터 속성 살펴보기(데이터 출력, 기초 정보, 통계치 확인)
속성 간 상관관계 분석하기(성적에 영향을 미치는 속성 선정)
학생 시험 속성 간의 관계 시각화하기(히트맵, 산점도로 시각화)
훈련 데이터와 테스트 데이터 나누기

배운 내용 정리하기

모델 생성하기	선형 회귀 모델 생성하기
모델 학습하기	모델 학습하기(훈련 데이터 사용)
모델 평가 및 예측하기	성능 평가하기(테스트 데이터 사용) 선형 회귀선 그리기/선형 회귀식 구하기
다른 회귀 모델에 적용하기(응용)	선형 회귀와 동일한 방법으로 다른 모델에 적용하기 (k-NN 회귀, RandomForest 회귀)

배운 내용 정리하기

1. 여러 개의 그래프 표현

맷플롯립의 subplots( ) 함수를 사용하면 설정한 행과 열의 수만큼 여러 개의 그래프를 표현할 수 있습니다. 이때 그래프가 겹칠 수 있으므로 그래프가 들어갈 크기를 설정하는 과정이 필요합니다. 속성 간의 관계를 나타내는 그래프를 통해 종속 변수에 영향을 미치는 속성이 무엇인지 찾을 수 있습니다.

배운 내용 정리하기

2. 선형 회귀 모델

수치형 데이터 Y에 대하여 Y에 영향을 미치는 데이터 X와의 관계를 평균값으로 예측하기 위해 만드는 모델로, X가 입력되었을 때 연속형 값인 Y를 도출할 수 있도록 합니다.

배운 내용 정리하기

3. 회귀 모델 평가 지표인 결정계수(R^2)와 평균 제곱 오차(MSE)

R^2는 0~1 사이의 값으로 출력하며 1에 가까울수록 오차가 적다는 것을 의미합니다.
MSE는 예측값과 실젯값을 비교하여 얼마나 차이가 있는지 평가하는 지표로 0에 가까울수록 오차가 적습니다.

배운 내용 정리하기

4. 회귀선과 회귀식

회귀선은 독립 변수와 종속 변수와의 관계를 직선 형태로 표현한 것입니다. 가중치와 절편으로 산출한 회귀식에서 2개 이상의 독립 변수가 존재한다면 가중치값이 클수록 종속 변수에 미치는 영향이 크다고 볼 수 있습니다.

배운 내용 정리하기

활동 정리하기

'머신러닝 문제 해결'의 '시험 점수를 예측하다' 활동에서 회귀 모델인 선형 회귀 모델을 사용하여 학생의 시험 점수를 예측해 보았다.

우선 우리는 문제 해결을 위해 학생 시험 점수 데이터(Student_Marks.csv)의 속성을 살펴보고, 속성 간 상관관계를 살펴보면서 학생의 시험 점수에 영향을 미치는 속성(독립 변수)과 영향을 받는 속성(종속 변수)이 무엇인지 확인할 수 있었다.