6장. 머신러닝(AI)

머신러닝(Machine Learning)

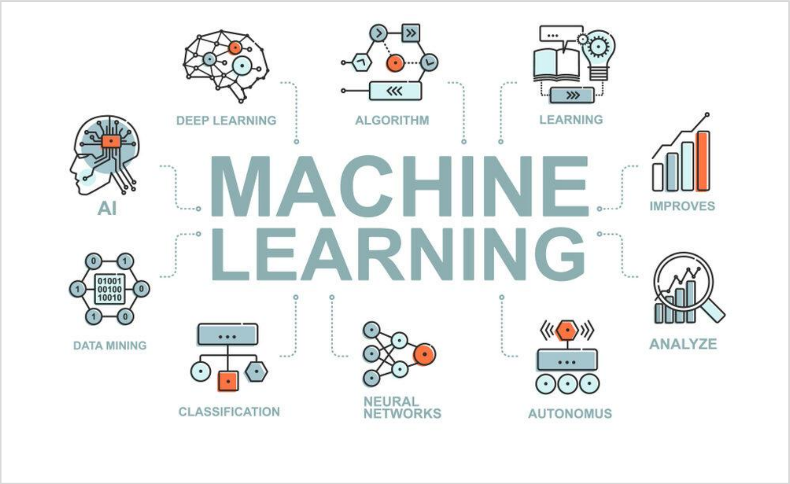
◆ 머신러닝이란?

컴퓨터가 스스로 방대한 양의 데이터를 학습하고 지식을 습득하여 문제를 해결하는 기술이다.

컴퓨터가 학습할때는 데이터와 함께 머신러닝 알고리즘이 필요하다.

머신러닝의 알고리즘의 종류로는 '지도 학습', '비지도학습', '강화 학습'이 있다.

알고리즘(Algorithm)
어떤 문제를 해결하기 위한 절차나 방법



머신러닝의 학습 방법

◆ 지도학습 / 비지도학습

	지도 학습(supervised)	비지도학습(unsupervised)
분석 모형 (알고리즘)	▪ 회귀 ▪ 분류	▪ 군집 ▪ 차원 축소
특징	정답(레이블)을 알고 있는 상태에서 학습	정답(레이블)이 없는 상태에서 서로 비슷한 데이터를 찾아서 그룹화(클러스터링)

지도학습(Supervised Learning)

- 분류(Classification)

  주어진 데이터의 특성을 학습해 새로운 데이터가 어떤 범주(클래스)에 속하는지 예측함

  - 이진분류 : 스팸/일반 메일 구분, 합격/불합격, CT사진에서 폐암이 보이는지 아닌지 예측

  - 다중 분류 : 다양한 새의 종류, 지폐의 종류 등

- 회귀(Regression)

  연속적인 값을 예측하는 것으로 다양한 값을 예측

  - 서울의 집값 예측

    특징 데이터(방의 개수, 대중교통 접근성, 범죄율, 집값 등),

    정답 데이터는 집값

비지도학습(Unsupervised Learning)

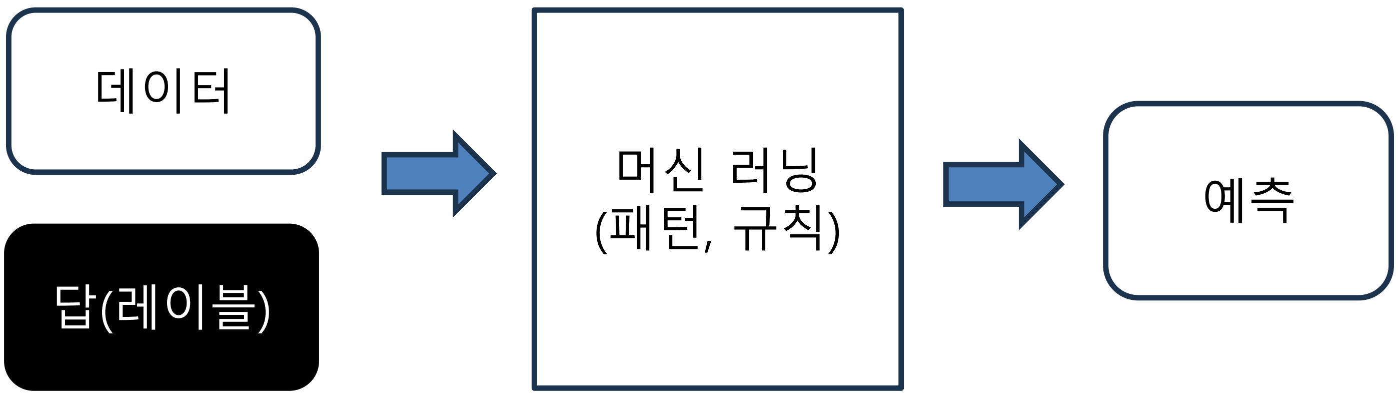
- 군집화(clustering)

  서로 비슷한 데이터를 찾아 그룹화 하는 과정

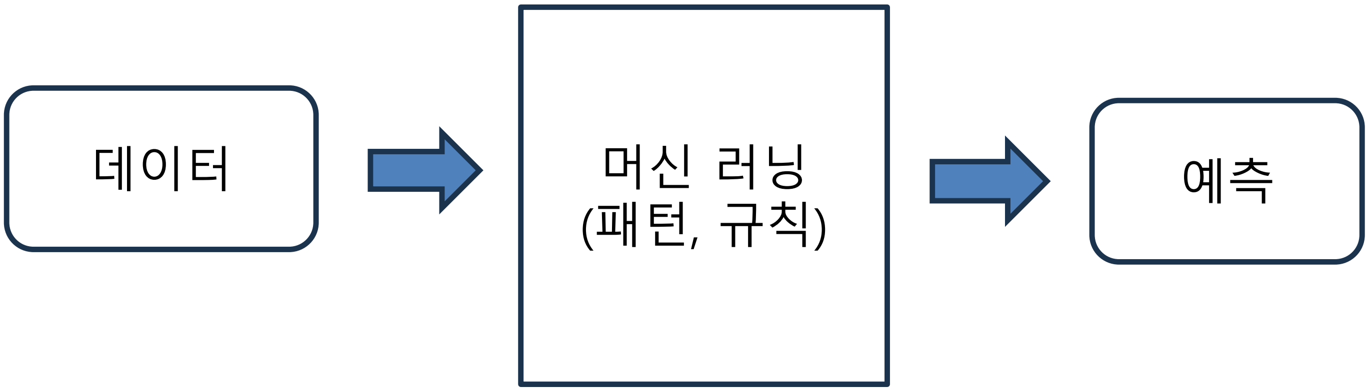
  - 유투브 : 추천 영상

  - 네이버, 카카오 : 사람이 구매한 내역을 보고 그 사람들을 여러 그룹으로 나눔. 알맞은 콘텐츠 제공

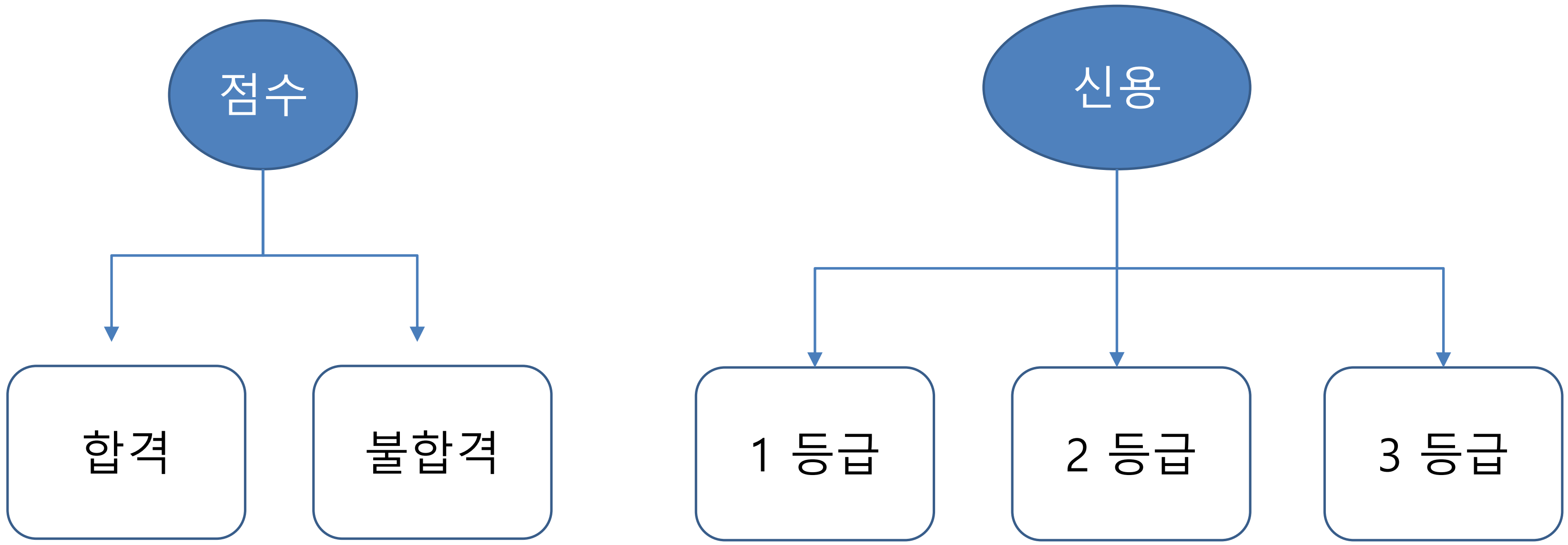
지도학습(Supervised Learning)



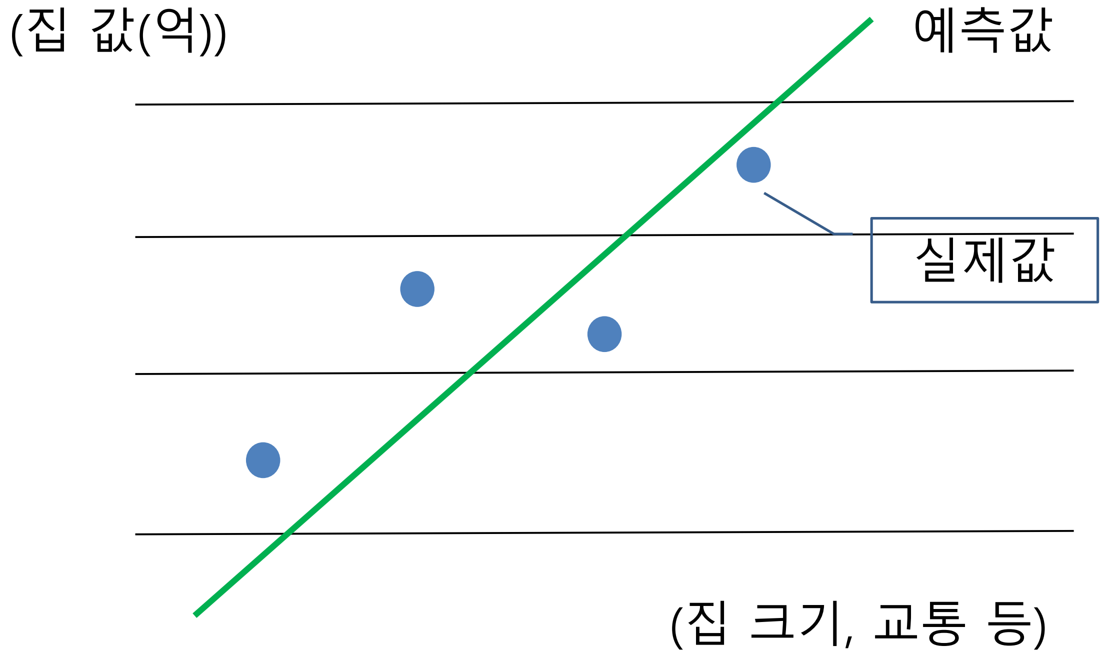
비지도학습(Unsupervised Learning)



분류(Classification)



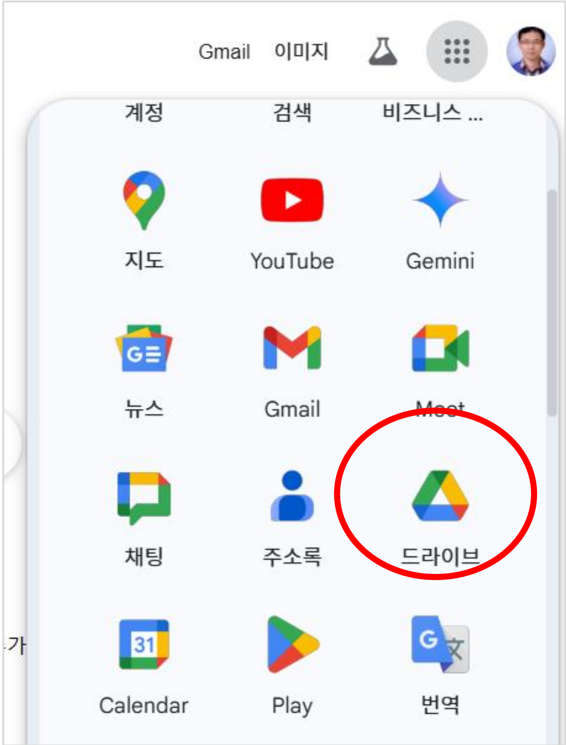
회귀(Regression)



구글 코렙(Colab)

❖ 구글 코랩(Colab)

구글에서 제공하는 클라우드 기반의 Jupyter 노트북 환경이다.



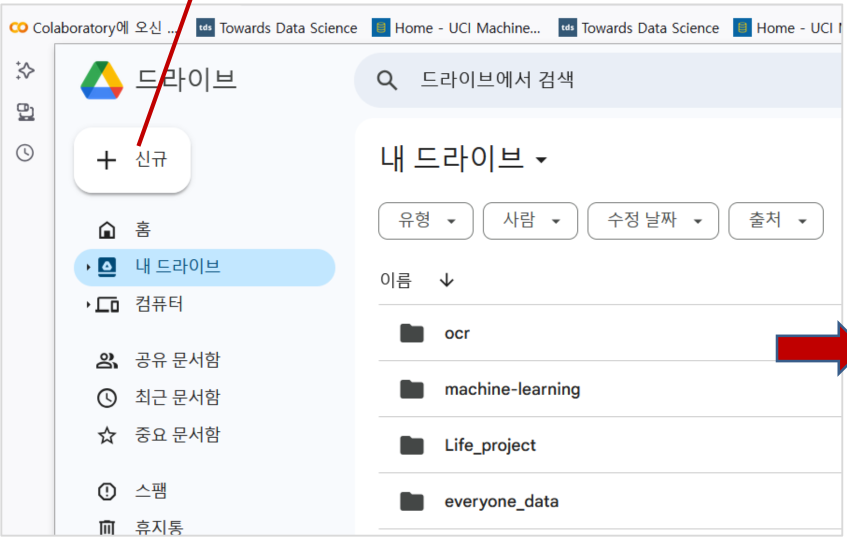
1. 드라이브로 연결하기

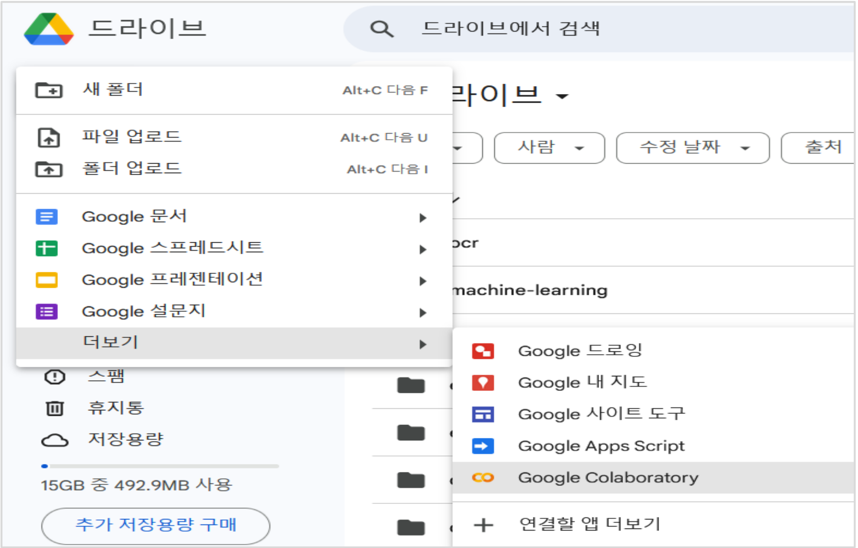
크롬 등 브라우저 > 구글 드라이브

구글 코렙

❖ 구글 코랩(Colaboratory)

신규 > 더보기 > Google Colaboratory





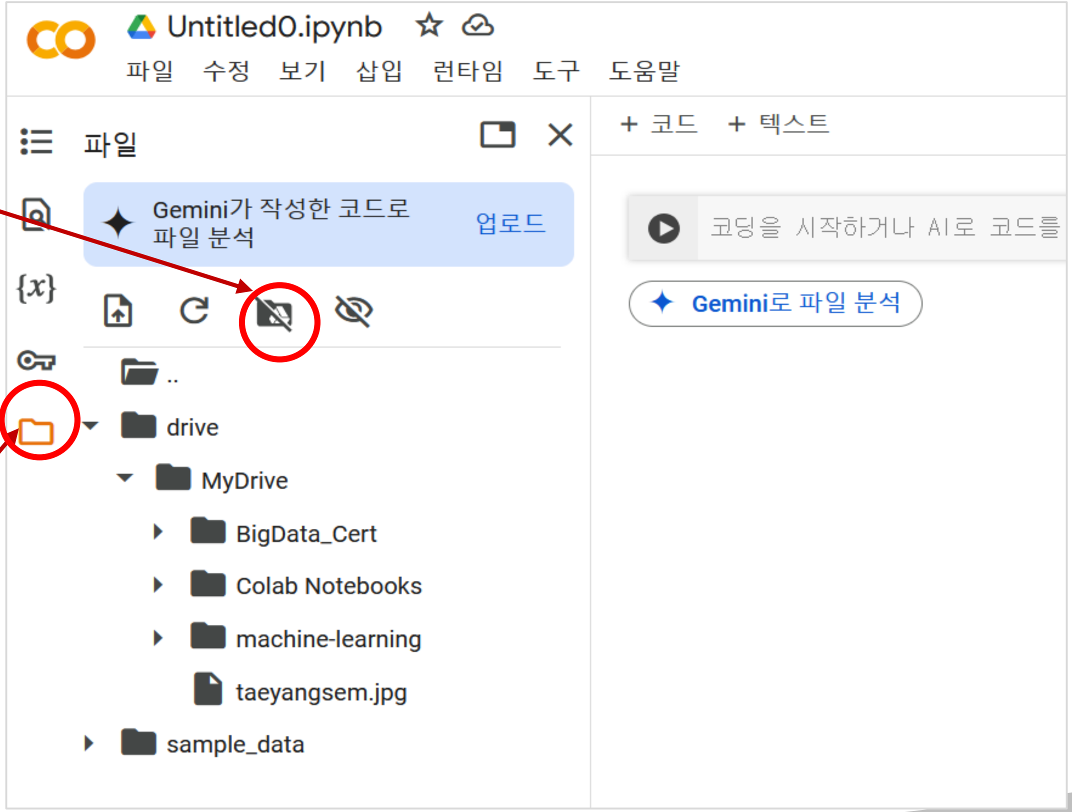
구글 코렙

❖ 구글 코랩(Colaboratory)

➢ Colab 웹으로 접속하기 : 새 노트 클릭

② 드라이브 연결

① 폴더 열기



구글 코렙

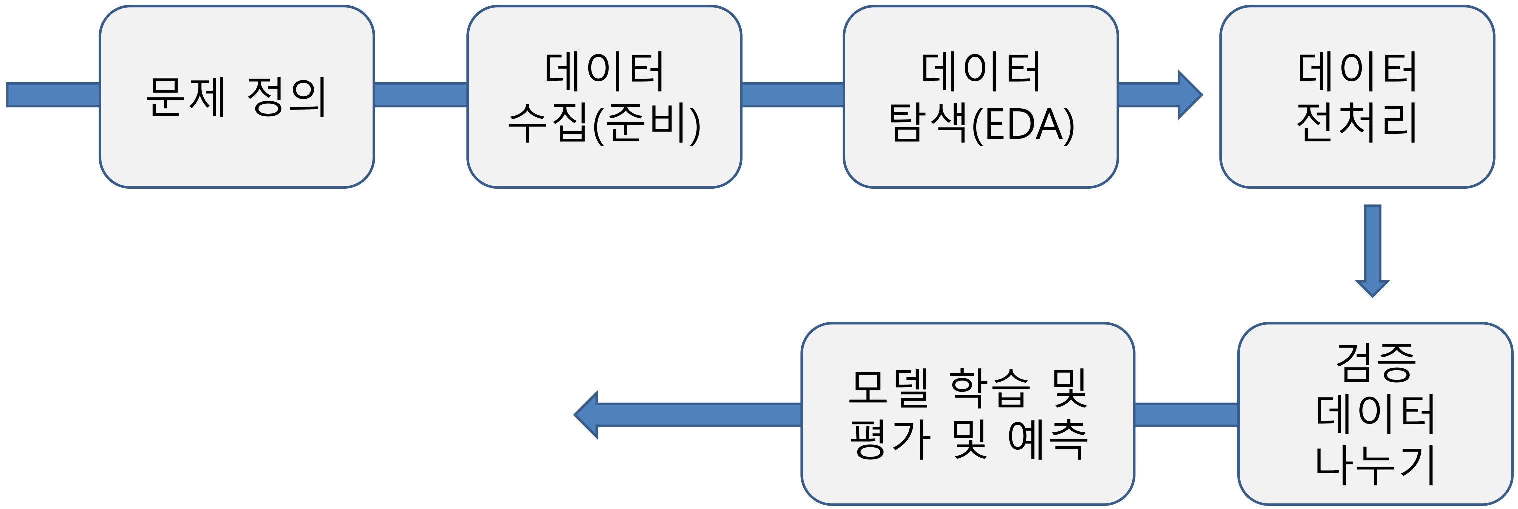
❖ 구글 코랩(Colaboratory)

➢ Colab 웹으로 접속하기

AI(Gemini) 지원 : 설정 > AI 지원 > 동의

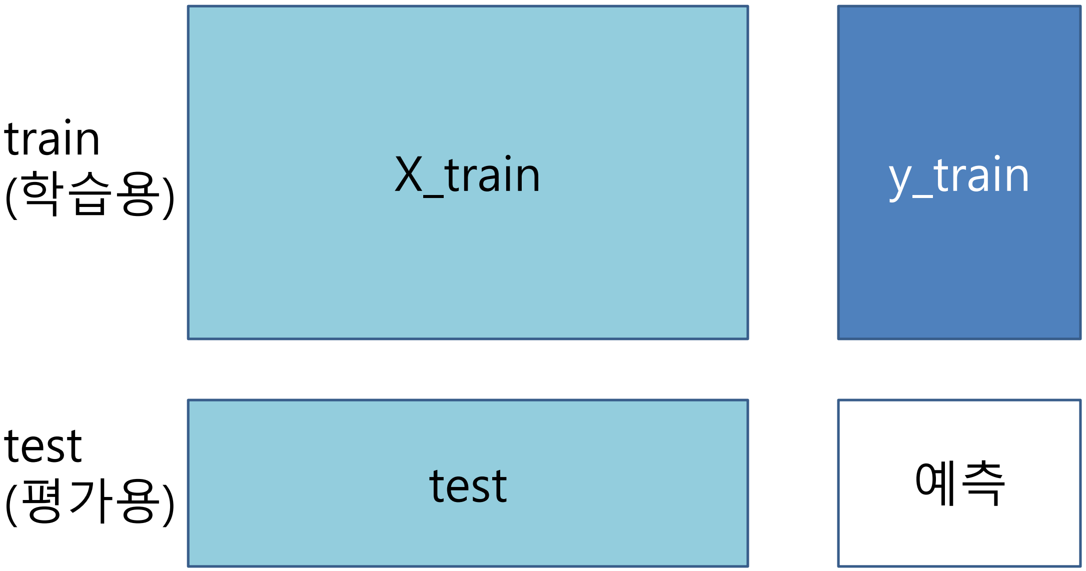
머신러닝 프로세스

- 머신러닝 모델 예측 프로세싱



데이터 수집(준비)

- 데이터는 학습용과 평가용으로 준비한다.



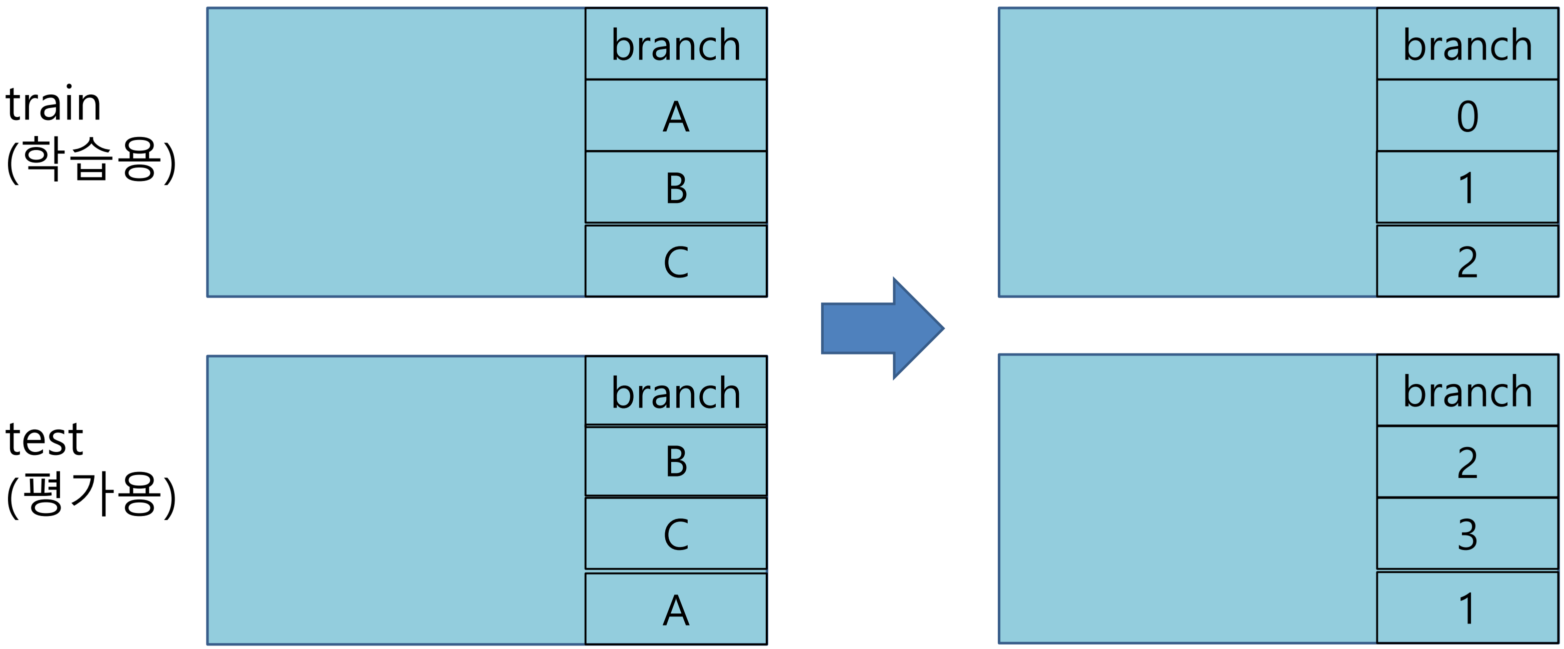
데이터 전처리

- 데이터 처리를 위한 모듈

	알고리즘	필수 여부
레이블 인코딩	sklearn.preprocessing(모듈) LabelEncoder	필수 (원-핫 인코딩중 선택)
스케일링	sklearn. preprocessing(모듈) StandardScaler	선택
데이터 분할	sklearn. model_selection(모듈) train_test_split	필수

데이터 전처리

- 인코딩(Encoding) – 문자형을 숫자형으로 변환
  - ✓ 레이블 인코딩(Label Encoding) : 각 칼럼을 숫자로 매핑하는 인코딩 방법이다.



머신러닝 프로세스

- 인코딩(Encoding) – 문자형을 숫자형으로 변환
  - ✓ 원-핫 인코딩(One-Hot Encoding) : 각 칼럼에 대해 새로운 칼럼을 만들고 새로운 칼럼에 0 또는 1이 저장된다.

train
(학습용)

branch
A
B
C

test
(평가용)

branch
B
C
A

branch_A	branch_B	branch_C
1	0	0
0	1	0
0	0	1

branch_A	branch_B	branch_C
0	1	0
0	0	1
1	0	0

머신러닝 프로세스

- 스케일링(Scaling)

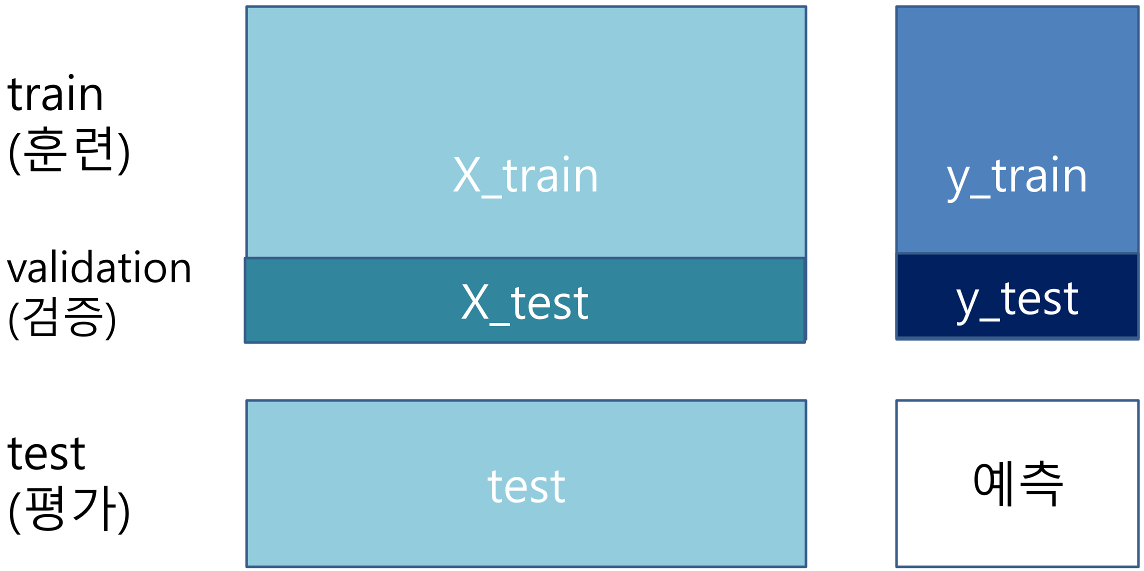
  설명(독립) 변수 열들이 갖는 데이터의 상대적 크기 차이를 없애기 위한 정규화 과정이다.

  - 0 ~ 10000인 경우 이름 0~1 사이의 값으로 변경함
  - A 칼럼 수치데이터가 0 ~ 10000이고, B 칼럼의 데이터가 0~10인 경우 두 칼럼간 데이터 차이가 커서 B 칼럼의 경우 유효하지 않은 데이터가 될 수 있음.
  - 스케일링을 통해 같은 범위(0~1)로 맞춤으로써 모델 성능을 향상 시킬 수 있음

A 칼럼	2000, ,999, 10, 10000	→	0.222, 0.09, 0.001, 1.000
B 칼럼	2, 9, 1, 4	→	0.022, 0.09, 0.001, 0.004

학습용 데이터 분할

- 학습용(훈련/검증) 데이터 분할
  - 훈련용 데이터 중 일부를 검증용 데이터로 분리한다. (8:2 정도비율)



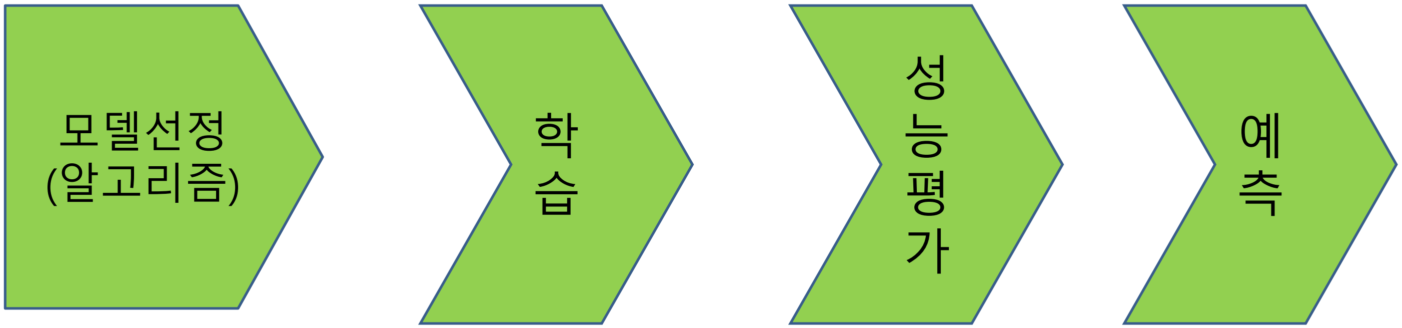
모델 학습 및 평가

- 모델 학습 및 평가

지도 학습	알고리즘	평가 지표
분류	sklearn.ensemble(모듈) RandomForestClassifier	sklearn.metrics(모듈) roc_auc_score, f1_score
회귀	sklearn.linear_model(모듈) LinearRegression sklearn.ensemble(모듈) RandomForestRegressor	sklearn.metrics(모듈) mean_squared_error mean_absolute_error

모델 학습 및 평가

- 모델 학습, 평가 및 예측



선형 회귀 분석

- **단순 선형 회귀**
  - $y=Wx+b$
  - W 를 가중치(Weight), b를 편향(bias)라고 부른다
  - 그래프의 형태는 **직선**

- **다중 선형 회귀**
  - $y=W_1x_1 + W_2x_2 + \dots + W_nx_n + b$
  - 만약 2개의 독립 변수면 그래프는 **곡선**으로 나타남

선형 회귀 분석

- 공부한 시간의 차이에 따른 시험 성적 예측하기
  - 성적을 변하게 하는 요소를 x라 하고 이 x값에 따라 변하는 '성적'을 y라 하자.
  - 독립 변수(x) : 공부한 시간, 사교육비 지출, 학생의 지능지수 등
  - 종속 변수(y) : 성적
  - 어떤 변수가 다른 변수에 영향을 준다면 두 변수 사이에 선형관계가 있다.

선형 회귀 분석

- 공부한 시간의 차이에 따른 시험 성적 예측하기

공부한 시간	2시간	4시간	6시간	8시간
성적	81점	93점	91점	97점

$x = \{2, 4, 6, 8\}$

$y = \{81, 93, 91, 97\}$

직선의 방정식을 구한다.

$y = ax + b$(일차함수)

기울기 a와 절편 b를 알아야 함

최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기
  - 기울기 a 구하기

$$a = \frac{(x - x\text{평균})(y - y\text{평균})\text{의 합}}{(x - x\text{평균})^2\text{의 합}}$$

공부한 시간(x) 평균 : (2 + 4 + 6 + 8) ÷ 4 = 5

성적(y) 평균: (81 + 93 + 91 + 97) ÷ 4 = 90.5

기울기 a = 2.3

X의 편차(각 값과 평균과의 차이)를 제곱해서 합한 값을 분모로 놓고,

x와 y의 편차를 곱해서 합한 값을 분자로 놓으면 기울기가 나옴

최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기
  - 편차(deviation)

    관측값에서 평균을 뺀 값

  - 분산 (variance)

    편차(관측값-평균)를 제곱하고 그것을 모두 더한 후 전체 개수로 나눈것. 즉 편차의 제곱의 평균을 말한다.

  - 표준편차(standard deviation)

    분산의 제곱근, 즉 분산 값에 루트를 씌운것을 말하며 stdev라고 함.

최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기
  - y절편 b값 구하기

b = y의 평균 – (기울기 a x (x의 평균))

b = 90.5 – (2.3 x 5) = 79

최적의 직선(방정식)

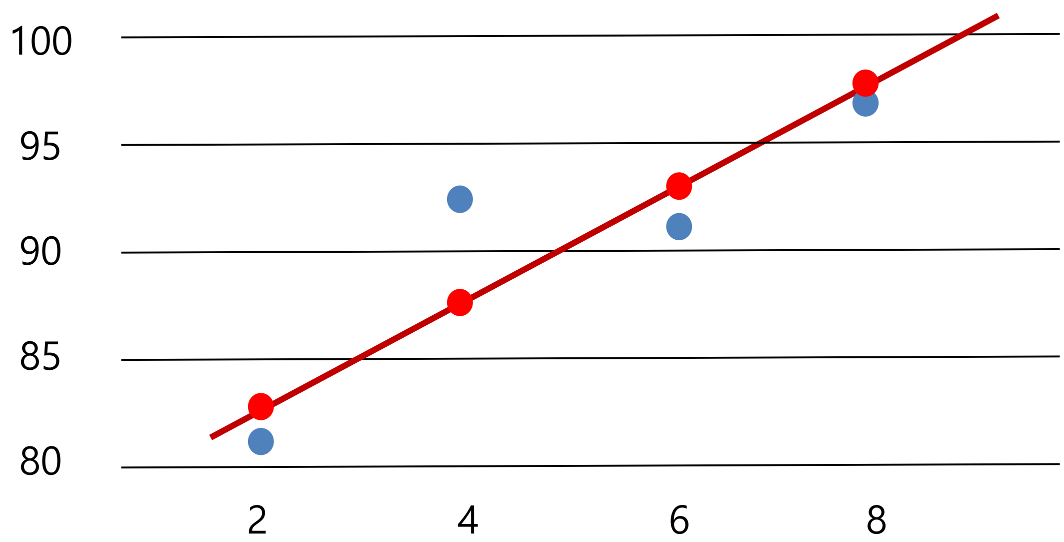
$y = 2.3x + 79$

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4

최소 제곱법

- 공부한 시간의 차이에 따른 시험 성적 예측하기

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	83.6	88.2	92.8	97.4



최소 제곱법

- 코드로 구현하기

```python
import numpy as np

x = np.array([2, 4, 6, 8])  #공부한 시간
y = np.array([81, 93, 91, 97])

print(x, y)

mx = np.mean(x)  # x의 평균값
my = np.mean(y)  # y의 평균값

print(mx, my)
```

```
[2 4 6 8] [81 93 91 97]
5.0 90.5
```

최소 제곱법

- 코드로 구현하기

```python
# 분자 계산 함수
# x와 y의 편차값을 곱하여 누적해서 합산함
def top(x, mx, y, my):
  d = 0
  for i in range(len(x)):
    d  = d + (x[i] - mx) * (y[i] - my)
  return d

#x의 편차를 제곱한 값
for i in x:
  print((i - mx)**2)

# 분모 - x의 편차를 제곱해서 합한 값
dividend = sum([(i - mx)**2 for i in x])

#분자
divisor = top(x, mx, y, my)
# print(divisor, dividend)
```

$$d = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

최소 제곱법

- 코드로 구현하기

```python
# 기울기
a = divisor / dividend

# y 절편
b = my - (mx * a)

print("기울기 a = ", a) #2.3
print("y절편 b = ", b)  #79

# 최적의 직선
# y = 2.3x + 79.0
```

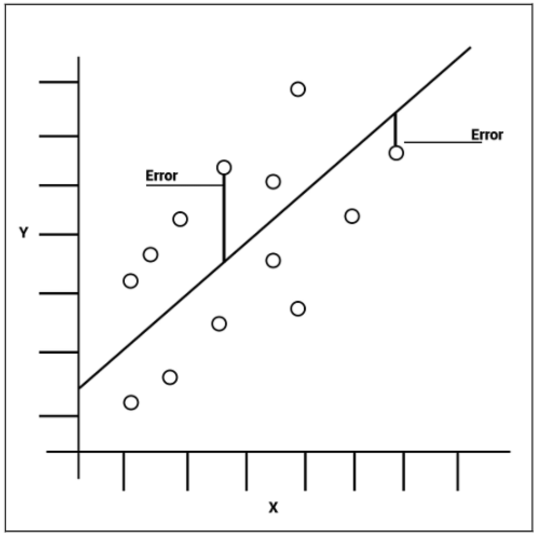
```
9.0
1.0
1.0
9.0
46.0 20.0
기울기 a =  2.3
y절편 b =  79.0
```

평균 제곱 오차

✓ 평균 제곱 오차

가설을 하나 세운 후(먼저 선을 긋고) 이 값이 주어진 요건을 충족하는지 판단해서 조금씩 변화를 주고 이 변화가 긍정적이면 오차가 최소가 될 때까지 이 과정을 계속 반복하는 방법.

선형 회귀는 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고, 이 값을 가장 작게 만들어주는 기울기(a)와 절편(b)을 찾아가는 작업이다.



평균 제곱 오차

✓ 평균 제곱 오차 / 평균 절대값 오차

- MSE(Mean Squared Error)
  - 평균 제곱오차
  - 오차의 제곱을 평균으로 나눈 것

$$MSE = \frac{\sum_{i=1}^{n} (y - \hat{y})^2}{n}$$

- MAE(Mean Absolute Error)
  - 평균 절대 오차
  - 오차의 차이를 절대값으로 변환한 뒤 합산

$$MAE = \frac{\sum |y - \hat{y}|}{n}$$

평균 제곱 오차

✓ 오차 = 실제값 – 예측값

공부한 시간	2	4	6	8
성적	81	93	91	97
예측 값	82	88	94	100
오차	**1**	**-5**	**3**	**3**

분모 : 1 + 25 + 9 + 9 = 44

MSE = 44 / 4 = 11

MAE = 1+5+3+3 = 12

평균 제곱 오차

✓ 평균 제곱 오차 계산

```
import numpy as np

# 가상의 기울기와 y 절편
fake_a = 3
fake_b = 76

# 성적 데이터 생성
x = np.array([2, 4, 6, 8])     #공부한 시간 - 독립 변수
y = np.array([81, 93, 91, 97]) # 점수 - 종속 변수

# 예측 점수
def predict(x):
  return fake_a * x + fake_b
```

평균 제곱 오차

✓ 평균 제곱 오차 계산

```
# 예측점수 리스트
predict_result = []

for i in range(len(x)):
  predict_result.append(predict(x[i]))
  print(f"공부한 시간 = {x[i]}, 실제 점수 = {y[i]}, 예측 점수 = {predict(x[i])}")
# print(predict_result)

# 평균 제곱 오차(Mean Squared Error) 함수
n = len(x)
def mse(y, y_pred):
  return sum((y - y_pred)**2) / n

# 평균 절대값 오차(Mean Absolut Error)
def mae(y, y_pred):
  return sum(abs(y - y_pred)) / n
```

평균 제곱 오차

✓ 평균 제곱 오차 계산

```
# print(mse(y, predict_result))
print(f"평균 제곱 오차: {mse(y, predict_result)}")

print(f"평균 절대값 오차: {mae(y, predict_result)}")
```

```
공부한 시간 = 2, 실제 점수 = 81, 예측 점수 = 82
공부한 시간 = 4, 실제 점수 = 93, 예측 점수 = 88
공부한 시간 = 6, 실제 점수 = 91, 예측 점수 = 94
공부한 시간 = 8, 실제 점수 = 97, 예측 점수 = 100
평균 제곱 오차: 11.0
평균 절대값 오차: 3.0
```

fake_a=3, fake_b=76로 만든 직선은 꽤 괜찮은 예측을 하지만, 여전히 오차가 존재

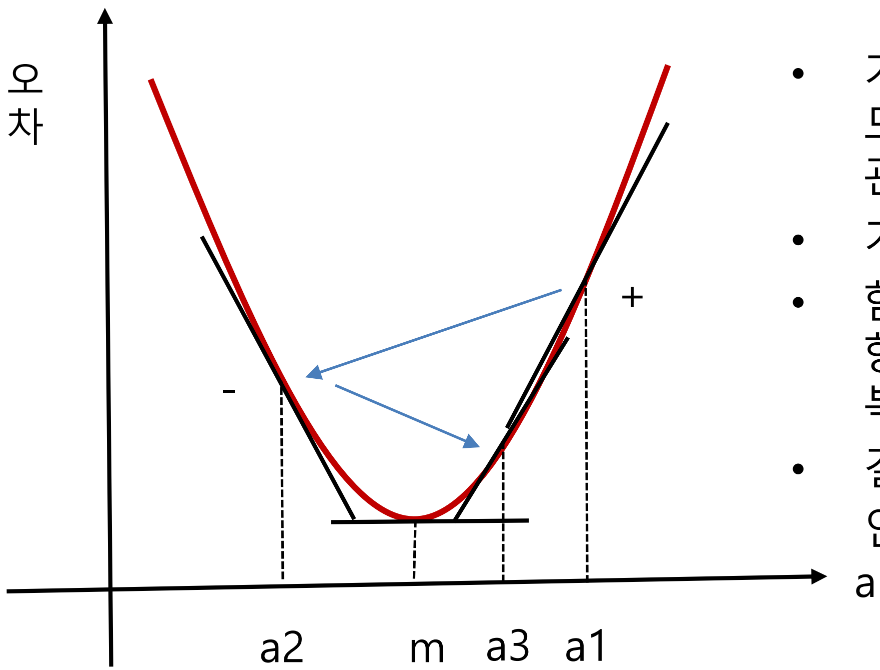
MSE = 11 → 큰 오차에 민감

MAE = 3 → 평균적으로 실제 점수와 예측 점수 차이가 3점 정도

(성능 평가 지표로 사용됨)

경사 하강법

- 경사 하강법(기울기 하강법)



미분값이 0인값을 찾음
이때 가중치는 0이 됨

- 기울기를 크게 잡으면 오차가 커지고, 작게 잡아도 오차가 커진다.기울기와 오차 사이에는 상관관계가 있다.
- 기울기와 오차 사이에는 이차 함수의 관계가 있다.
- 함수의 기울기(경사)를 구하고 기울기의 반대 방향으로 계속 이동시켜서 최소값에 이를 때까지 반복시킨다.
- 결국 m에 이르게 되면 최적의 기울기를 찾는 것인데 이러한 방법을 **경사하강법**이라 한다.

경사 하강법

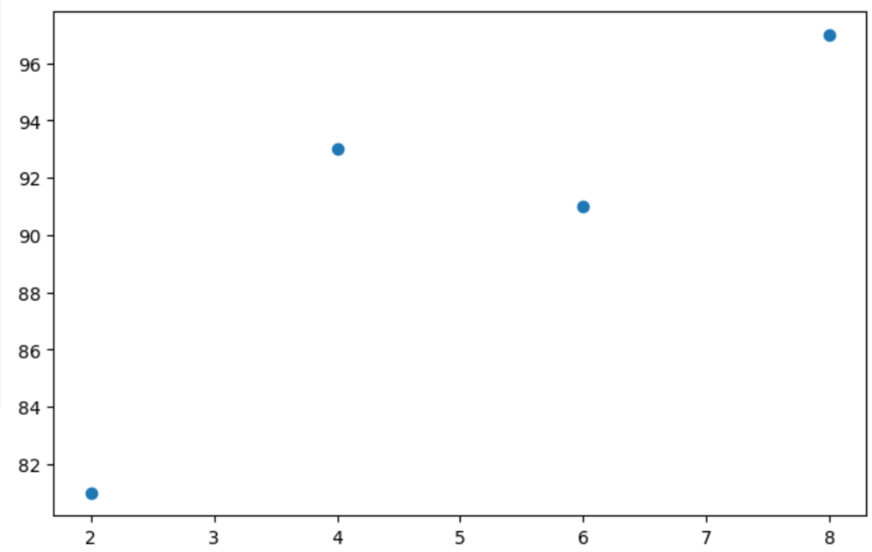
- 경사 하강법

```
import numpy as np
import matplotlib.pyplot as plt

# 공부시간 X와 성적 Y의 리스트 만들기
x = np.array([2, 4, 6, 8])
y = np.array([81, 93, 91, 97])

print(x, y)

# 산점도 그래프로 나타내기
plt.figure(figsize=(8,5))
plt.scatter(x, y)
plt.show()
```



경사 하강법

- 경사 하강법

```python
a = 0 #기울기
b = 0 #절편

lr = 0.03  #학습률(learning rate)

epochs = 2001 #반복 횟수

# 경사하강법 시작
n = len(x)
for i in range(epochs):
  y_pred = a * x + b  #예측값
  error = y - y_pred  #오차 = 실제값 - 예측값

  # 오차 제곱합을 a, b로 각각 편미분한 값
  a_diff = -(1/n) * sum(x * (error))
  b_diff = -(1/n) * sum(error)

  # 기울기 하강법(Gradient Descent) 업데이트
  a = a - lr * a_diff #학습률을 곱해 기존의 a값을 업데이트함
  b = b - lr * b_diff

  if i % 100 == 0: #100번 반복할때마다 출력
    print(f"epoch = {i}, 기울기 a = {a}, y절편 b = {b}")
```

$$\frac{\partial L}{\partial a} = -\frac{2}{n}\sum x_i(y_i - (ax_i + b))$$

$$\frac{\partial L}{\partial b} = -\frac{2}{n}\sum(y_i - (ax_i + b))$$

편미분 공식
여기서는 2를 생략했는데, 어차피 학습률이 조절해주기 때문에 결과에는 큰 문제 없음.

경사 하강법

- 경사 하강법

```
y_pred = a * x + b  #학습이 끝난 예측값
print("실제 점수:", y)
print("예측 점수:", y_pred)
print("기울기:", a)
print("절편:", b)

n = len(x)
# 평균 제곱 오차(Mean Squared Error) 함수
def mse(y, y_pred):
  return sum((y - y_pred)**2) / n

# 평균 절대값 오차(Mean Absolut Error)
def mae(y, y_pred):
  return sum(abs(y - y_pred)) / n

# 예측 성능 평가
print("MSE:", mse(y, y_pred))
print("MAE:", mae(y, y_pred))

# 그래프 그리기
plt.figure(figsize=(8, 5))
plt.scatter(x, y)
plt.plot(x, y_pred, 'r') #학습된 회귀 직선
plt.show()
```

경사하강법의 예측 성능이
평균 제곱 오차법보다 좋다.

경사 하강법

- 경사 하강법

```
epoch = 0, 기울기 a = 13.92, y절편 b = 2.715
epoch = 100, 기울기 a = 10.130651018642078, y절편 b = 32.270023654986495
epoch = 200, 기울기 a = 7.108781631586109, y절편 b = 50.30324779416332
epoch = 300, 기울기 a = 5.253059806295641, y절편 b = 61.377402011009856
epoch = 400, 기울기 a = 4.113466047673803, y절편 b = 68.1780101924428
epoch = 500, 기울기 a = 3.4136445997654805, y절편 b = 72.35424530094595
epoch = 600, 기울기 a = 2.983886139571057, y절편 b = 74.9188599966009
epoch = 700, 기울기 a = 2.719972630402819, y절편 b = 76.49378295143588
epoch = 800, 기울기 a = 2.5579040575351977, y절편 b = 77.46093888244904
epoch = 900, 기울기 a = 2.4583781848577226, y절편 b = 78.0548667263618
epoch = 1000, 기울기 a = 2.3972596153722927, y절편 b = 78.4195962104744
epoch = 1100, 기울기 a = 2.3597268669978946, y절편 b = 78.64357560114365
epoch = 1200, 기울기 a = 2.336678107637268, y절편 b = 78.78112073974577
epoch = 1300, 기울기 a = 2.3225239267932576, y절편 b = 78.86558683770481
epoch = 1400, 기울기 a = 2.313831882582523, y절편 b = 78.91745724022823
epoch = 1500, 기울기 a = 2.308494121719225, y절편 b = 78.94931071426039
epoch = 1600, 기울기 a = 2.305216217196075, y절편 b = 78.96887184659325
epoch = 1700, 기울기 a = 2.303203264885523, y절편 b = 78.98088428510339
epoch = 1800, 기울기 a = 2.3019671163107582, y절편 b = 78.98826109113396
epoch = 1900, 기울기 a = 2.301208000811144, y절편 b = 78.99279116778476
epoch = 2000, 기울기 a = 2.3007418300340166, y절편 b = 78.99557307561541
실제 점수: [81 93 91 97]
예측 점수: [83.59705674 88.1985404  92.80002406 97.40150772]
기울기: 2.3007418300340166
절편: 78.99557307561541
MSE: 8.300003266758804
MAE: 2.400012027909753
```