머신 러닝

• 머신러닝(Machine Learning) 개요

• 인공지능 (Artificial Intelligence)

사람의 지적 능력을 컴퓨터를 통해 구현하는 기술로 하위 개념으로 딥러닝과 머신러닝이 있다.

• 머신러닝 (Machine Learning)

머신러닝은 인간이 데이터 분석의 힌트를 알려준 후 학습해서 추론할 수 있게 하는 기술이다.

• 딥러닝 (Deep Learning)

딥러닝은 처음부터 기계가 학습하는 구조로 머신러닝이 쉽게 할 수 없는 이미지, 비디오, 비정형 데이터를 분석할 수 있다.

• 머신러닝 구현 예제

예를 들어 중고 스마트폰을 파는 경우 판매자와 구매자간의 적절한 가격을 정해야한다. 가격을 정하는 기준은 기기의 스펙(제조사/모델명/제조연월/화면크기/CPU코어/내장메모리/램(RAM)/색상) 또는 상태(파손여부/스크래치/번인)에 따라 달라진다.

• 사람이 직접 프로그래밍

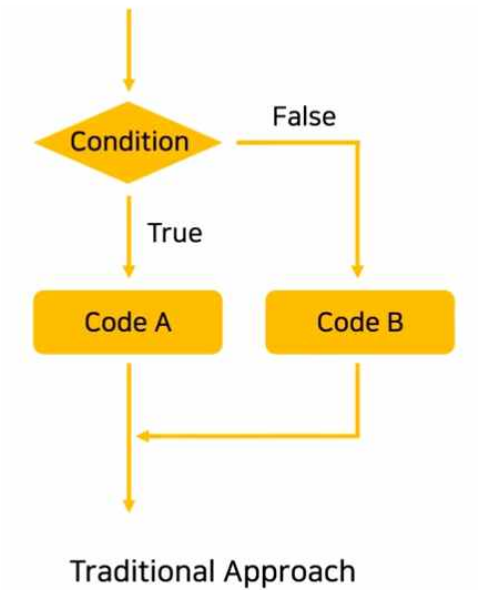
```
def get_price(최대금액, 액정파손, 번인, 찍힘, 스크래치, 색상, ....):
    적정금액 = 최대금액  #300000

    if 액정파손  == True:
        적정금액 -= 150000
    elif 번인 == True:
        적정금액 -= 50000

    if 찍힘 > = 3:  #3군데 이상
        적정금액 -= 30000
    elif 스크래치 == True:
        적정금액 -= 5000

    if 배터리 < 10  #10시간미만
        적정금액 -= 20000

    ...
    return 적정금액
```

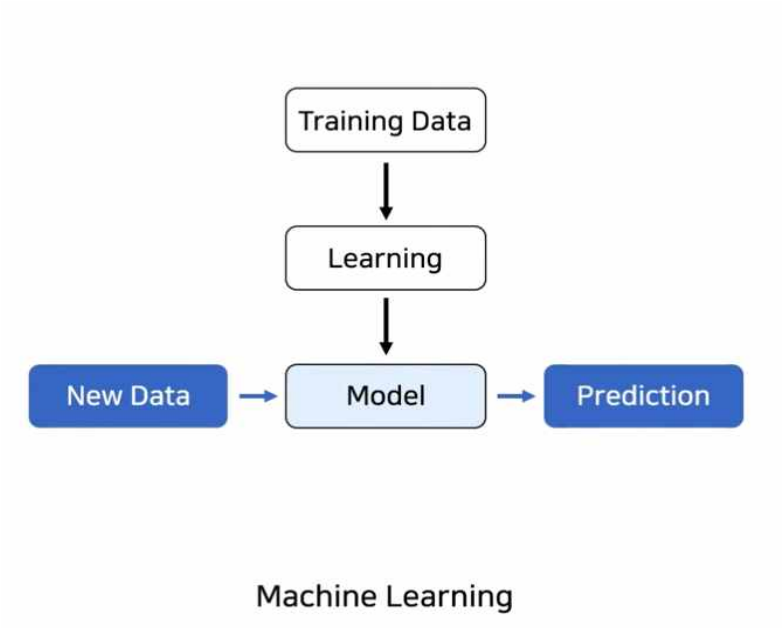


- **기계가 스스로 학습**

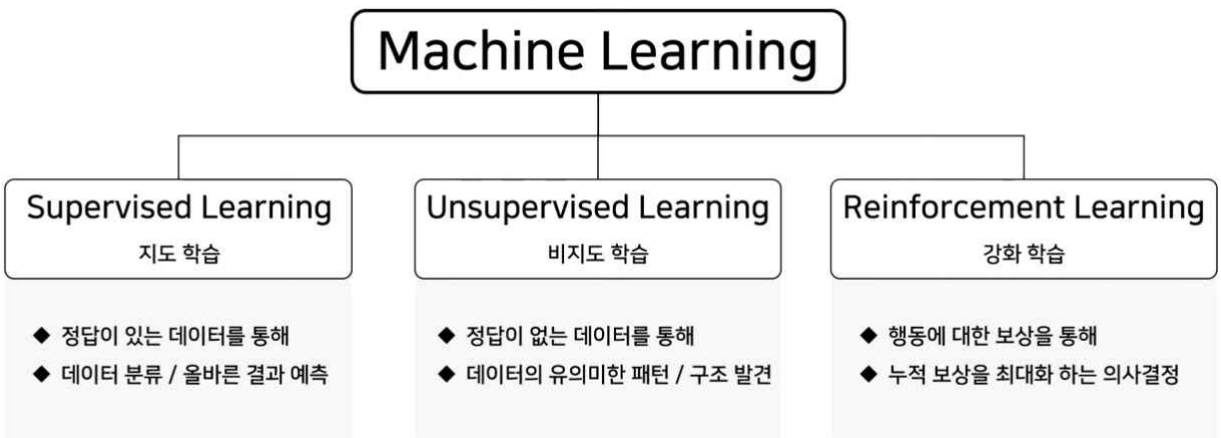
기계가 아래 주어진 거래 Data를 분석하여 상관관계를 찾아서 거래 가격을 스스로 결정한다.

제조사	모델명	제조 연월	내장메모리	램 (RAM)	액정파손	번인	찍힘	생활기스	...	거래가격
A	A1	2021	128	8	X	O	1	O		280,000
A	A1	2021	256	12	X	X	3	O		310,000
A	A1	2021	256	12	O	O	5	O		235,000
B	B1	2020	256	8	X	O	2	O		560,000
B	B2	2022	512	12	X	X	1	X		820,000
...	...	...	...	...	...	...	...	...	...	...

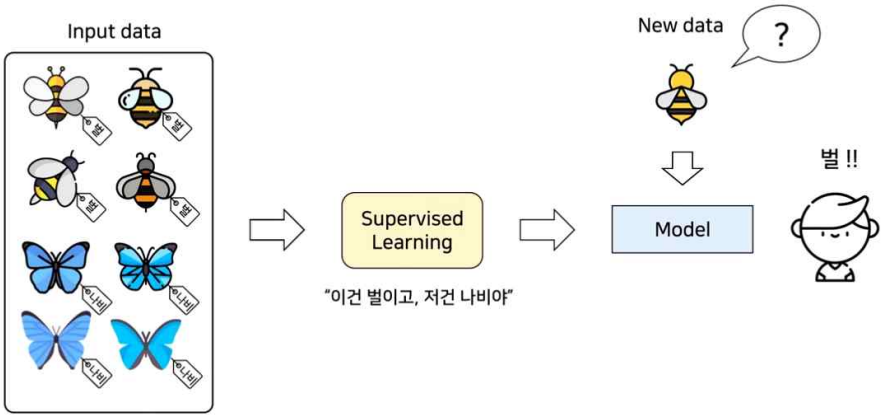
훈련 데이터를 제공하면 기계는 데이터를 통해 학습을 하여 새로운 모델(앞에서 생성한 모델에 해당)을 생성한다. 이 모델을 이용하여 새로운 데이터를 입력하여 결과를 예측한다.



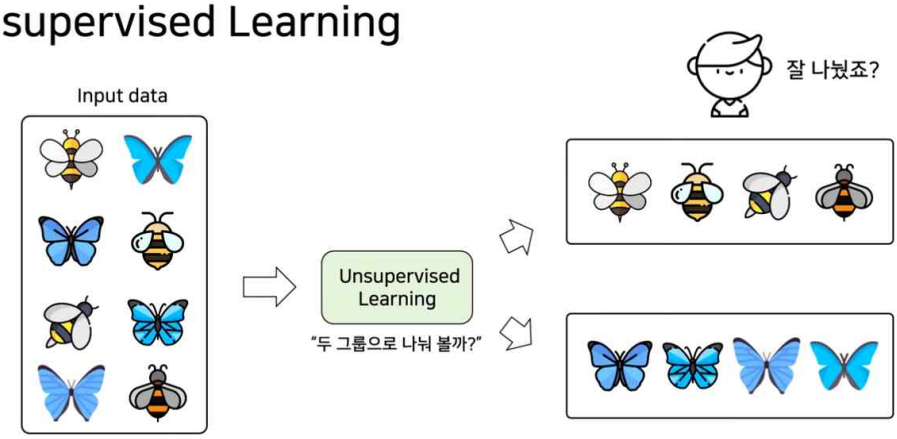
- **머신러닝 분류**



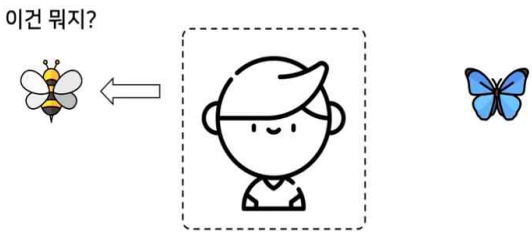
Supervised Learning



Unsupervised Learning



Reinforcement Learning



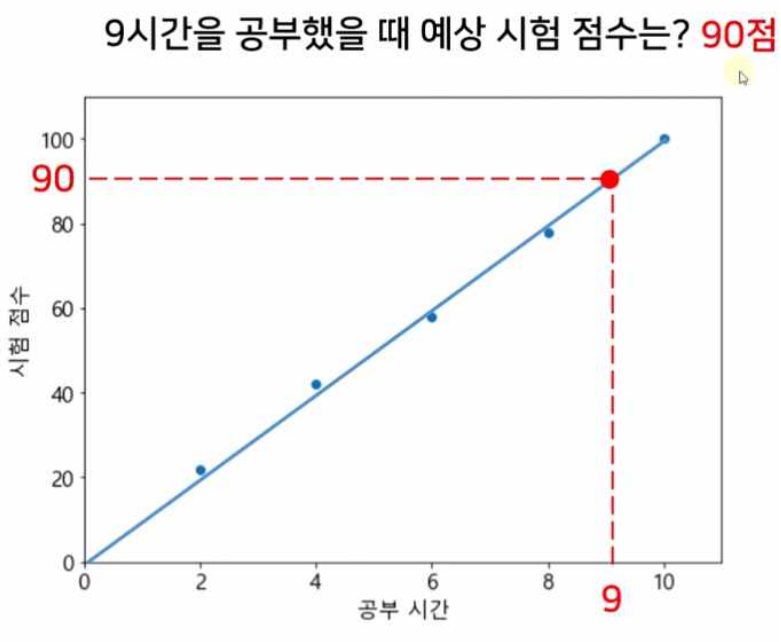
- **지도학습(Supervised Learning) 종류**

- **회귀(Regression)**

회귀란 변수들 간의 상관관계를 찾는 것, 연속적인 데이터로부터 결과를 예측하는 것으로 예측 결과가 숫자일 때 적합하다. 예를 들어 근속연수에 따른 임금, 키에 따른 몸무게, 사용 기간에 따른 스마트폰 가격 등이 있다.

아래 예제는 주어진 데이터를 분석한 후 상관관계를 갖는 직선의 방정식을 구하여 공부시간에 대한 점수를 예측한다.

공부 시간	시험 점수
2 시간	22 점
4 시간	42 점
6 시간	58 점
8 시간	78 점
10 시간	100 점

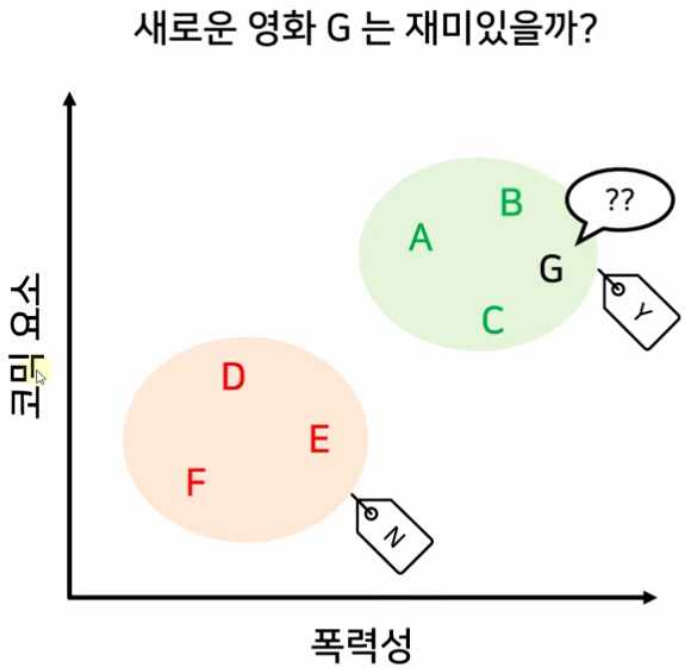


- **분류(Classification)**

주어진 데이터를 정해진 범주(category)에 따라 분류하여 예측하는 것으로 예측 결과가 숫자가 아닐 때 적합하다. 예를 들어 스팸 메일 필터링, 시험 합격 여부, 재활용 분리수거 품목, 악성 종양 여부 등이 있다.

아래는 코믹과 폭력성을 기준(좋아요)으로 그룹으로 나누어 G가 어는 그룹에 속하는지에 따라서 좋아요 결과를 예측한다.

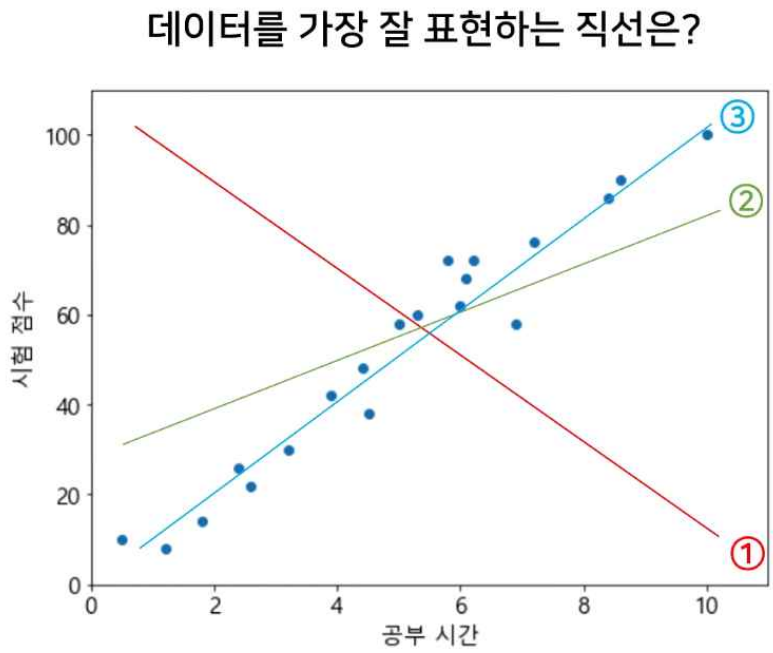
영화	좋아요
A	Y
B	Y
C	Y
D	N
E	N
F	N



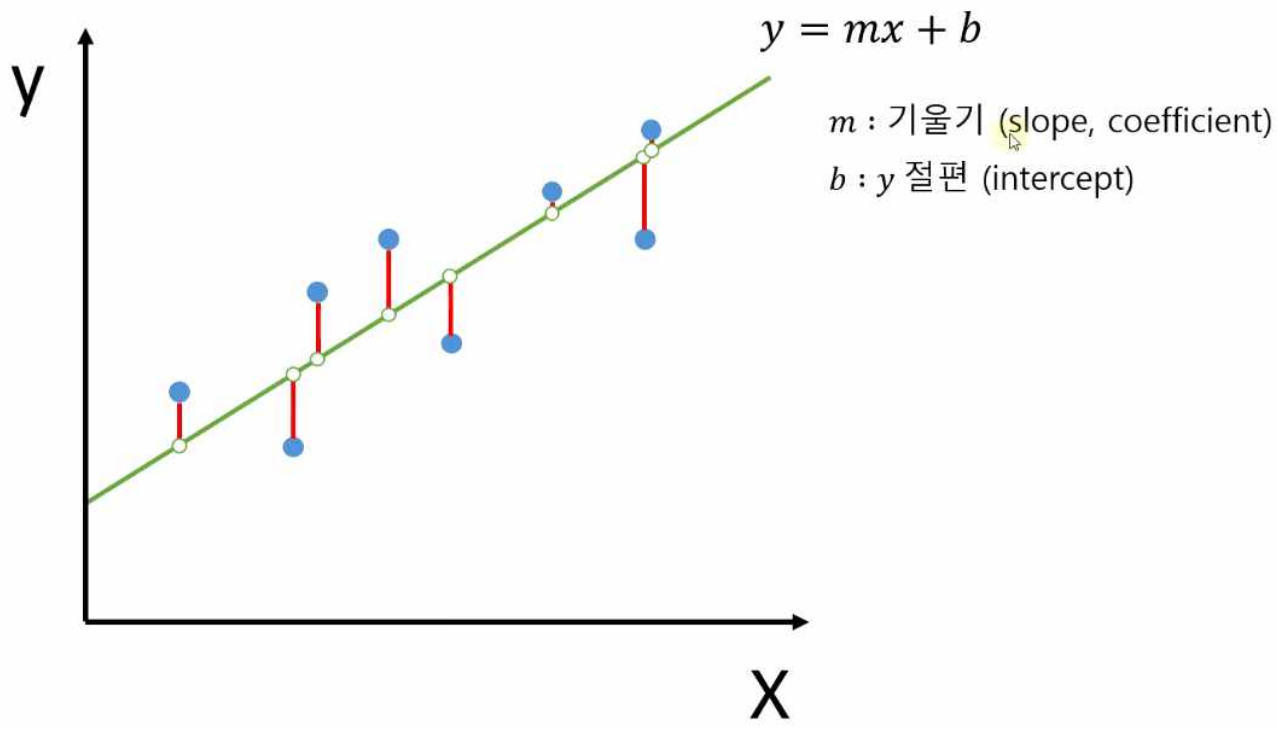
• 선형 회귀 (Linear Regression)

선형 회귀란 종속변수(y)와 한 개 이상의 독립변수(X)와의 선형 상관관계를 모델링하는 회귀분석 기법이다.

공부 시간	시험 점수
0.5	10
1.2	8
1.8	14
2.4	26
2.6	22
3.2	30
3.9	42
4.4	48
4.5	38
5	58
5.3	60
5.8	72
6	62
6.1	68
6.2	72
6.9	58
7.2	76
8.4	86
8.6	90
10	100



- **공부시간: Independent variable(독립변수) 원인(X) = 입력변수, feature**
- **시험점수: Dependent Variable(종속변수) 결과 (y) = 출력변수, target, label**
- **최적의 직선: (실제 값: 파랑 - 예측 값: 초록)의 제곱의 합이 최소인 직선**



• 선형 회귀 실습1

▪ 새로운 01.Linear Regression.ipynb 파일을 생성한 후 터미널에서 아래의 두 개의 라이브러리를 설치한다.

```
pip install matplotlib
pip install pandas
```

▪ 시각화를 위한 matplotlib 과 데이터 분석을 위한 pandas 라이브러리를 import한다.

```
import matplotlib.pyplot as plt
import pandas as pd
```

▪ 공부시간에 따른 시험점수가 저장되어 있는 csv 파일을 읽어온다.

```
dataset = pd.read_csv('LinearRegressionData.csv')
```

▪ head()함수를 이용하여 상위 5개의 데이터가 출력되는지 확인해 본다.

```
dataset.head()
```

	hour	score
0	0.5	10
1	1.2	8
2	1.8	14
3	2.4	26
4	2.6	22

▪ 데이터세트에서 독립변수와 종속변수의 값을 저장한다.

1) 독립변수(원인)에 모든 행의 처음부터 마지막 칼럼 직전까지의 데이터 값을 X에 저장한다.
2) 종속변수(결과)에 모든 행의 마지막 칼럼 데이터 값을 y에 저장한다.

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

▪ X변수와 y변수 값들을 출력하여 확인해 본다. X 값들은 2차원 배열로, y 값들은 1차원 배열로 출력된다.

```
X, y
```

```
(array([[ 0.5],
        [ 1.2],
        [ 1.8],
        [ 2.4],
        [ 2.6],
        ...
        [ 6.9],
        [ 7.2],
        [ 8.4],
        [ 8.6],
        [10. ]]),
 array([ 10,   8,  14,  26,  22,  30,  42,  48,  38,  58,  60,  72,  62,
         68,  72,  58,  76,  86,  90, 100], dtype=int64))
```

- 인공지능 선형 회귀 모델을 생성하기 위하여 아래 라이브러리를 설치한다.

```
pip install scikit-learn
```

- LinearRegression 클래스로 선형 회귀 객체 생성한 후 fit 함수로 학습을 하여 모델을 생성한다.

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X, y)
```

```
▾ LinearRegression
LinearRegression()
```

- 독립변수(X)에 대한 예측 값(y_pred)을 predict()함수를 이용하여 예측한 후 출력해 본다.

```
y_pred = reg.predict(X)
y_pred
```

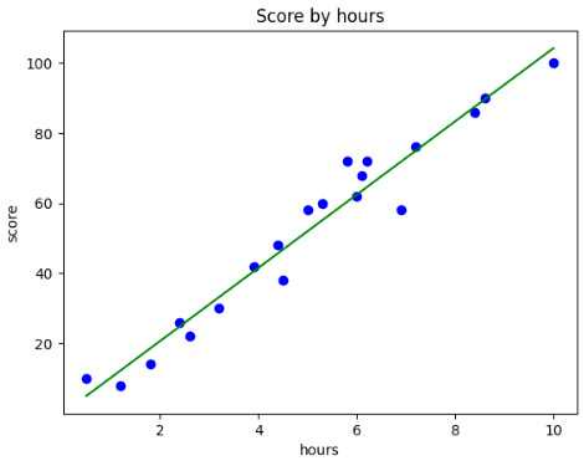
```
array([  5.00336377,  12.31395163,  18.58016979,  24.84638795,
        26.93512734,  33.20134551,  40.51193337,  45.73378184,
        46.77815153,  52.        ,  55.13310908,  60.35495755,
        62.44369694,  63.48806663,  64.53243633,  71.84302419,
        74.97613327,  87.5085696 ,  89.59730899, 104.2184847 ])
```

• 선형 회귀 실습2

matplotlip 라이브러리를 사용하여 그래프 출력으로 시각화 작업을 해본다.

- 실제 값들은 scatter() 함수를 이용해 산점도 그래프로 예측 값들은 plot() 함수를 이용하여 선 그래프로 출력해 본다.

```
plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='green')
plt.title('Score by hours')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- **독립변수(X) 데이터가 이차원 배열 형태이므로 값을 2차원 배열 형태로 입력해 준다.**

```
print('9시간 공부했을 때 예상 점수: ', reg.predict([[9]]))
```

```
9시간 공부했을 때 예상 점수:  [93.77478776]
```

```
print('9, 8, 7시간 공부했을 때 예상 점수: ', reg.predict([[9], [8], [7]]))
```

```
9, 8, 7시간 공부했을 때 예상 점수:  [93.77478776 83.33109082 72.88739388]
```

- **선형 회귀 모델(y=mx + b)의 기울기(coefficient)값 m을 출력해 본다.**

```
reg.coef_
```

```
array([10.44369694])
```

- **선형 회귀 모델(y=mx + b)의 절편(intercepter)값 b을 구한 후 출력해 본다.**

```
reg.intercept_
```

```
-0.218484702867201
```

```
y = 10.4436 * 9 -0.2184
y
```

```
93.774
```

- **데이터 세트 분리**

기존 데이터를 가지고 모델 평가를 위하여 읽어온 데이터 세트를 훈련 세트(Train set) 80%, 테스트 세트(Test set) 20%로 나눈다. 그리고 훈련 세트로 모델을 만들고 테스트 세트로 훈련 모델이 잘 만들었는지 테스트 한다.

```
import matplotlib.pyplot as plt
import pandas as pd
```

```
dataset = pd.read_csv('LinearRegressionData.csv')
dataset
```

	hour	score
0	0.5	10
1	1.2	8
2	1.8	14
3	2.4	26
...		
16	7.2	76
17	8.4	86
18	8.6	90
19	10.0	100

- **독립변수(X)와 종속변수(y)에 데이터 값을 저장한다.**

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

- **train_test_split(80:20) 함수를 이용해 훈련 세트와 테스트 세트를 분리한다.**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

- **전체 데이터 세트 값과 개수를 출력해 본다.**

```
X, len(X)
```

```
(array([[ 0.5],
        [ 1.2],
        [ 1.8],
        [ 2.4],
        [ 2.6],
        [ 3.2],
        [ 3.9],
        [ 4.4],
        [ 4.5],
        [ 5. ],
        [ 5.3],
        [ 5.8],
        [ 6. ],
        [ 6.1],
        [ 6.2],
        [ 6.9],
        [ 7.2],
        [ 8.4],
        [ 8.6],
        [10. ]]),
 20)
```

- **훈련 세트 데이터 값과 개수를 출력해 본다. (원본 데이터 중에서 80%개를 Random하게 선택해 저장한다.)**

```
X_train, len(X_train)
```

```
(array([[5.3],
        [8.4],
        [3.9],
        [6.1],
        [2.6],
        [1.8],
        [3.2],
        [6.2],
        [5. ],
        [4.4],
        [7.2],
        [5.8],
        [2.4],
        [0.5],
        [6.9],
        [6. ]]),
 16)
```

- **테스트 세트 데이터 값과 수를 출력해 본다. (원본 데이터 중에서 20%개를 Random하게 선택해 저장한다.)**

```
X_test, len(X_test)
```

```
(array([[ 8.6],
        [ 1.2],
        [10. ],
        [ 4.5]]),
 4)
```

- **전체 데이터 종속변수(y)값을 출력해 본다.**

```
y, len(y)
```

```
(array([ 10,  8,  14,  26,  22,  30,  42,  48,  38,  58,  60,  72,  62,
        68,  72,  58,  76,  86,  90, 100], dtype=int64),
 20)
```

- **훈련 세트(y_train) 값을 출력해 본다.**

```
y_train, len(y_train)
```

```
(array([60, 86, 42, 68, 22, 14, 30, 72, 58, 48, 76, 72, 26, 10, 58, 62],
       dtype=int64),
 16)
```

- **테스트 세트(y_test) 값을 출력해 본다.**

```
y_test, len(y_test)
```

```
(array([ 90,  8, 100,  38], dtype=int64), 4)
```

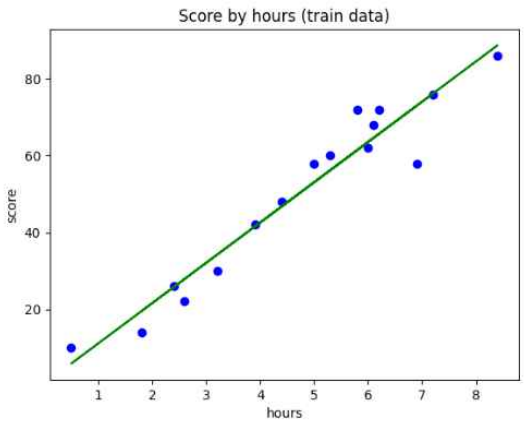
- **분리된 데이터(훈련 세트)를 통해 LinearRegression 객체를 생성하고 학습 후 선형회귀 모델을 생성한다.**

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

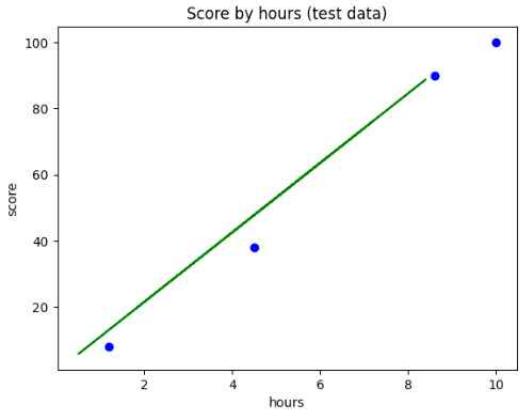
- **훈련 세트의 데이터 시각화**

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, reg.predict(X_train), color='green')
plt.title('Score by hours (train data)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- **테스트 세트의 데이터 시각화**

```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_train, reg.predict(X_train), color='green')
plt.title('Score by hours (test data)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- **훈련 모델의 기울기를 출력해 본다.**

```
reg.coef_
```

```
array([10.49161294])
```

- **훈련 모델의 y절편을 출력해 본다.**

```
reg.intercept_
```

```
0.6115562905169369
```

- **테스트 세트를 통한 모델을 평가해 본다.**

```
reg.score(X_test, y_test)
```

```
0.9727616474310156
```

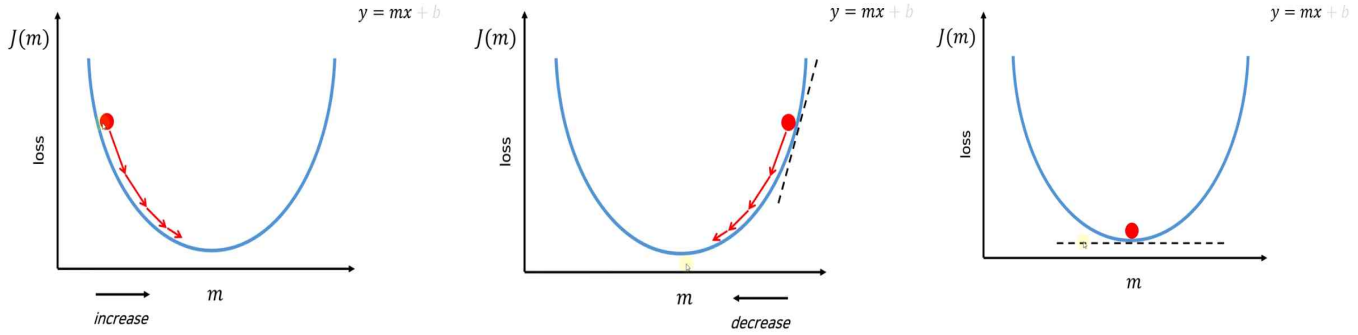
- **훈련 세트를 통한 모델을 평가해 본다.**

```
reg.score(X_train, y_train)
```

```
0.9356663661221668
```

- **경사 하강법 (Gradient Descent)**

경사 하강법은 선형 회귀 최소제곱법(OLS:Ordinary Least Squares)의 노이즈(이상 값)에 취약점을 보완하기위한 모델링 기법으로 실제값과 예측값과의 loss를 최소화 하는 기울기(m)에 값을 찾는 것을 목표로 한다.



- **확률적 경사 하강법(Stochastic Gradient Descent)**

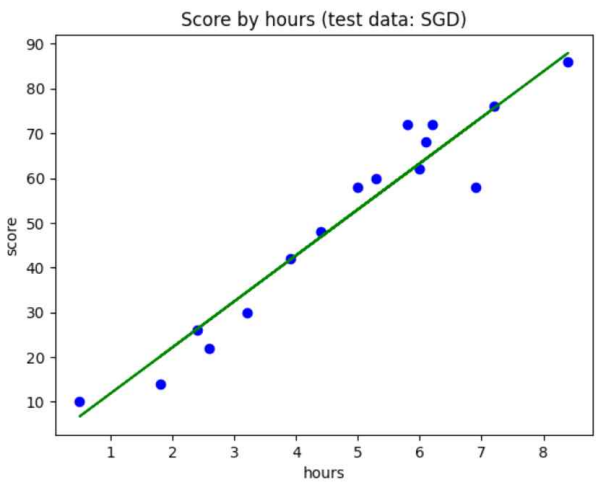
경사 하강법은 전체 데이터에 대하여 가중치를 적용하기 때문에 데이터가 많을수록 시간이 오래 걸리는 단점이 있다. 이런 점을 보강하는 알고리즘이 여러 가지 있는데 그 중 하나가 확률적 경사 하강법(SGD) 이다. SGD는 랜덤적으로 추출한 하나의 데이터에만 가중치를 적용한다. 즉, 속도가 훨씬 빨라지지만 정확도는 전체 데이터에 대하여 가중치를 적용하는 경사 하강법에 비하여 떨어진다.

- **확률적 경사 하강(SGD: Stochastic Gradient Descent) 클래스로 객체를 생성 후 훈련 모델을 생성한다.**

```
from sklearn.linear_model import SGDRegressor
sr = SGDRegressor()
sr.fit(X_train, y_train)
```

```
▾ SGDRegressor
SGDRegressor()
```

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, sr.predict(X_train), color='green')
plt.title('Score by hours (test data: SGD)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()  #LinearRegression이용한 직선과 거의 유사하다.
```



- **선형회귀 모델의 기울기(10.49161294), y절편(0.6115562905169369)과 비교해 본다.**

```
sr.coef_, sr.intercept_
```

```
(array([10.21667615]), array([1.32363048]))
```

- **테스트 세트를 통한 경사하강 모델을 평가한다.**

```
sr.score(X_test, y_test)
```

```
0.9770432600935411
```

- **훈련 세트를 통한 경사하강 모델을 평가한다.**

```
sr.score(X_train, y_train)
```

```
0.9343604100805729
```

- **확률적 경사 하강 옵션 사용**

1) max_iter : 최대 학습 반복 횟수 (Epoch 횟수)- Epoch를 높일수록 다양한 무작이 가중치로 학습을 하므로 손실 값이 작아진다.
2) eta0 : 학습률 (learning rate) 지수 표기법 가능(1e-3 : 0.001)-학습률이 작으면 보폭이 조금씩, 크면 큰 보폭으로 움직인다.
3) verbose : 생략하면 아래 결과 출력이 사라진다.

- **최대 학습 반복회수가 1000이고 학습률이 기본값(0.001)인 경우에는 학습 횟수가 84에서 loss값이 가장 작다.**

```
from sklearn.linear_model import SGDRegressor
sr = SGDRegressor(max_iter=1000, eta0=0.001, random_state=0, verbose=1)
sr.fit(X_train, y_train)
```

```
-- Epoch 1
Norm: 2.40, NNZs: 1, Bias: 0.442470, T: 16, Avg. loss: 1181.034371
Total training time: 0.00 seconds.
...
-- Epoch 84
Norm: 10.28, NNZs: 1, Bias: 1.783802, T: 1344, Avg. loss: 17.015926
Total training time: 0.00 seconds.
Convergence after 84 epochs took 0.00 seconds
```

- **최대 학습 반복회수가 1000이고 학습률이 기본값(0.0001)인 경우에는 학습 횟수가 873에서 loss값이 가장 작다.**

```
from sklearn.linear_model import SGDRegressor
sr = SGDRegressor(max_iter=1000, eta0=1e-4, random_state=0, verbose=1)
sr.fit(X_train, y_train)  #학습률이 적어지만 학습 반복횟수는 늘어난다.
```

```
-- Epoch 1
Norm: 0.27, NNZs: 1, Bias: 0.048869, T: 16, Avg. loss: 1484.241876
Total training time: 0.00 seconds.
...
-- Epoch 873
Norm: 10.19, NNZs: 1, Bias: 1.776030, T: 13968, Avg. loss: 17.042407
Total training time: 0.06 seconds.
Convergence after 873 epochs took 0.06 seconds
```

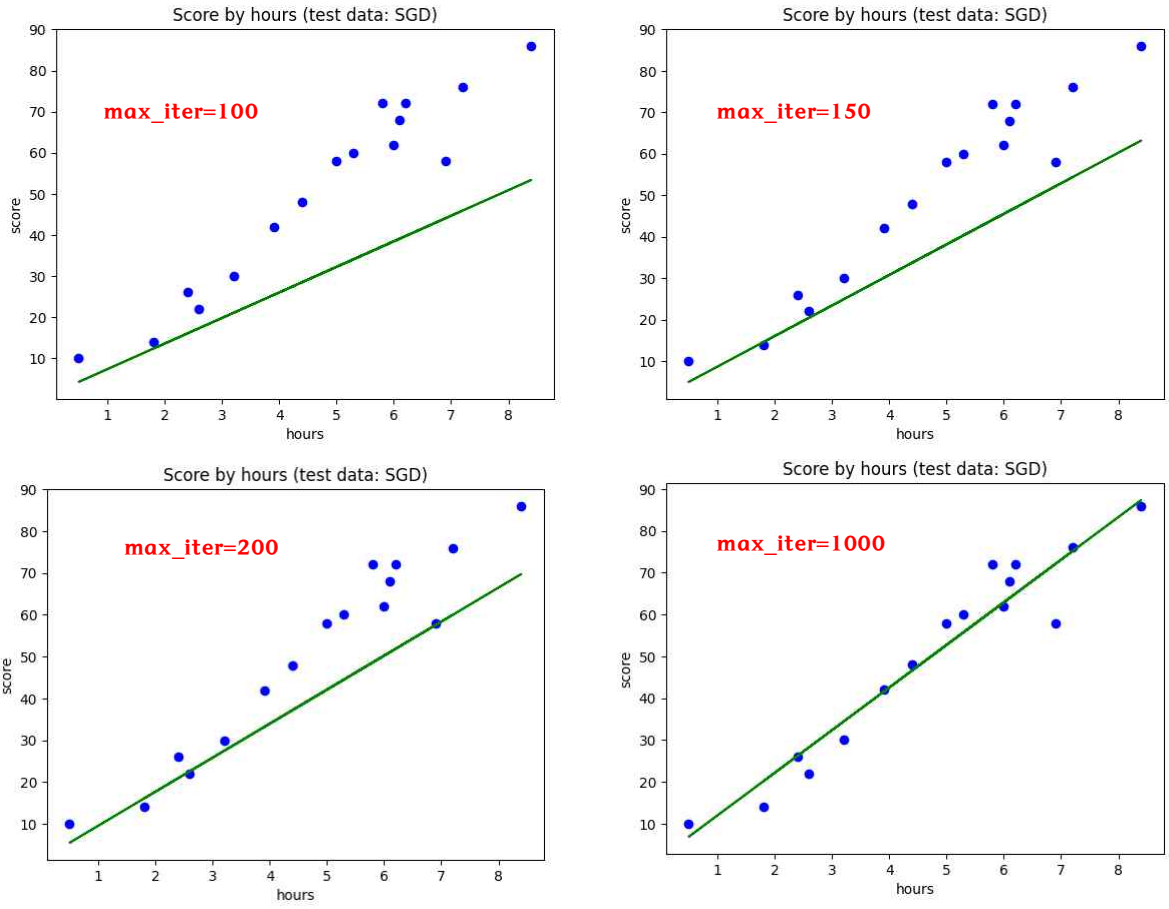
- **최대 학습 반복회수가 100이고 학습률이 기본값(0.0001)인 경우에는 학습 횟수가 873에서 loss값이 굉장히 크다.**

```
from sklearn.linear_model import SGDRegressor
sr = SGDRegressor(max_iter=100, eta0=1e-4, random_state=0, verbose=1)
sr.fit(X_train, y_train)
```

```
-- Epoch 1
Norm: 0.27, NNZs: 1, Bias: 0.048869, T: 16, Avg. loss: 1484.241876
Total training time: 0.00 seconds.
-- Epoch 2
Norm: 0.47, NNZs: 1, Bias: 0.083896, T: 32, Avg. loss: 1419.741822
Total training time: 0.00 seconds.
...
-- Epoch 100
Norm: 6.22, NNZs: 1, Bias: 1.098974, T: 1600, Avg. loss: 253.278959
Total training time: 0.01 seconds.
```

- **학습 반복횟수에 따라 그래프를 그려보면 반복횟수 많을수록 정확한 예측 모델이 생성됨을 확인할 수 있다.**

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, sr.predict(X_train), color='green')
plt.title('Score by hours (test data: SGD)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



• 다중 선형 회귀 (Multiple Linear Regression)

다중 선형 회귀란 종속변수(y)와 여러 개의 독립변수(X)들과의 선형 상관관계를 모델링하는 회귀분석 기법이다.

공부 시간	결석 횟수	공부 장소	시험 점수
0.5	3	Home	10
1.2	4	Library	8
1.8	2	Cafe	14
2.4	0	Cafe	26
2.6	2	Home	22
3.2	0	Home	30
3.9	0	Library	42
4.4	0	Library	48
4.5	5	Home	38
5	1	Cafe	58
5.3	2	Cafe	60
5.8	0	Cafe	72
6	3	Library	62
6.1	1	Cafe	68
6.2	1	Library	72
6.9	4	Home	58
7.2	2	Cafe	76
8.4	1	Home	86
8.6	1	Library	90
10	0	Library	100

$$y = b + m_1x_1 + m_2x_2 + m_3x_3$$

공부 시간 결석 횟수 공부 장소

One-Hot Encoding

공부 장소
Home
Library
Cafe

Home	Library	Cafe
1	0	0
0	1	0
0	0	1

다중공선성

※ Dummy Column 이 n 개면? n-1 개만 사용

공부 장소
Home
Library
Cafe

Home	Library	삭제 Cafe
1	0	0
0	1	0
0	0	1

공부 시간	결석 횟수	Home	Library	Cafe	시험 점수
0.5	3	1	0	0	10
1.2	4	0	1	0	8
1.8	2	0	0	1	14
2.4	0	0	0	1	26
2.6	2	1	0	0	22
3.2	0	1	0	0	30
3.9	0	0	1	0	42
4.4	0	0	1	0	48
4.5	5	1	0	0	38
5	1	0	0	1	58
5.3	2	0	0	1	60
5.8	0	0	0	1	72
6	3	0	1	0	62
6.1	1	0	0	1	68
6.2	1	0	1	0	72
6.9	4	1	0	0	58
7.2	2	0	0	1	76
8.4	1	1	0	0	86
8.6	1	0	1	0	90
10	0	0	1	0	100

• 다중 선형 회귀 실습1 (전처리 작업)

▪ 02.Multiple Linear Regression.ipynb 파일을 생성한 후 데이터 세트를 읽어 독립변수와 종속변수 값을 지정한다.

```
import pandas as pd
dataset = pd.read_csv('MultipleLinearRegressionData.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
array([[0.5, 3, 'Home'],
       [1.2, 4, 'Library'],
       [1.8, 2, 'Cafe'],
       [2.4, 0, 'Cafe'],
       [2.6, 2, 'Home'],
       ...
       [6.9, 4, 'Home'],
       [7.2, 2, 'Cafe'],
       [8.4, 1, 'Home'],
       [8.6, 1, 'Library'],
       [10.0, 0, 'Library']], dtype=object)
```

▪ 원-핫 인코딩(One-Hot Encodeing)을 위해 필요한 라이브러리를 import 한 후 인코딩 작업을 한다.

1) drop='first' : 첫 번째 값은 삭제한다. (10:Home, 01:Library, 00:Caffe)
2) [2] : index 2 칼럼을 원-핫 인코딩한다.
3) remainder = 'passthrough' 나머지는 칼럼들은 그냥 Passing한다.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [2])], remainder='passthrough')
X = ct.fit_transform(X)
X
```

```
array([[1.0, 0.0, 0.5, 3],
       [0.0, 1.0, 1.2, 4],
       [0.0, 0.0, 1.8, 2],
       [0.0, 0.0, 2.4, 0],
       [1.0, 0.0, 2.6, 2],
       [1.0, 0.0, 3.2, 0],
       [0.0, 1.0, 3.9, 0],
       ...
       [0.0, 1.0, 6.2, 1],
       [1.0, 0.0, 6.9, 4],
       [0.0, 0.0, 7.2, 2],
       [1.0, 0.0, 8.4, 1],
       [0.0, 1.0, 8.6, 1],
       [0.0, 1.0, 10.0, 0]], dtype=object)
```

• 다중 선형 회귀 실습2

▪ 데이터 세트를 훈련 데이터 세트와 테스트 세트로 분리한다.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

- **훈련 세트를 입력 받아 학습 후 선형 회귀 객체를 생성 하고 학습 후 선형 회귀 모델을 생성한다.**

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

- **테스트 세트의 예측 값(y_pred)과 실제 값(y_test)을 비교한다.**

```
y_pred = reg.predict(X_test)
y_pred
```

```
array([ 92.15457859,  10.23753043, 108.36245302,  38.14675204])
```

```
y_test
```

```
array([ 90,   8, 100,  38], dtype=int64)
```

- **훈련 세트의 기울기(m1, m2, m3, m4)를 출력한다.**

```
reg.coef_  #Home(-5.82712824), Library(-1.04450647), Caff(0), 공부시간당 점수(10.40419528), 결석1회당 점수(-1.64200104)
```

```
array([-5.82712824, -1.04450647, 10.40419528, -1.64200104])
```

- **훈련 세트의 y절편(b)값을 출력한다.**

```
reg.intercept_
```

```
5.3650067065447615
```

- **훈련 세트로 모델을 평가해 본다.**

```
reg.score(X_train, y_train)
```

```
0.9623352565265527
```

- **테스트 세트로 모델을 평가해 본다.**

```
reg.score(X_test, y_test)
```

```
0.9859956178877446
```

- **카페에서 9시간 공부하고 1일 결석한 경우 예측 점수를 출력하시오.**

```
reg.predict([[0, 0, 9, 1]])
```

```
array([97.36076317])
```

- **회귀 모델 평가**

모델 생성 후 모델의 신뢰성을 평가하여 더 좋은 모델을 선택할 수 있다. 평가방법은 아래 4개의 방법들이 있다.

Evaluation

y	$\hat{y}$	$\lvert y-\hat{y}\rvert$
15	25	10
55	42	13
50	59	9
95	76	19
80	93	13

MAE (Mean Absolute Error)

: 실제 값과 예측 값 차이의 절대값들의 평균

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

ex) $MAE = \frac{10+13+9+19+13}{5} = 12.8$

Evaluation

y	$\hat{y}$	$(y-\hat{y})^2$
15	25	100
55	42	169
50	59	81
95	76	361
80	93	169

MSE (Mean Squared Error)

: 실제 값과 예측 값 차이의 제곱한 값들의 평균

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

ex) $MSE = \frac{100+169+81+361+169}{5} = 176$

Evaluation

y	$\hat{y}$	$(y-\hat{y})^2$
15	25	100
55	42	169
50	59	81
95	76	361
80	93	13

RMSE (Root Mean Squared Error)

: MSE 에 루트를 적용

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

ex) $RMSE = \sqrt{\frac{100+169+81+361+169}{5}}$

$= \sqrt{176} = 13.27$

Evaluation

y	$\hat{y}$	$(y-\hat{y})^2$	$(y-\bar{y})^2$
15	25	100	1936
55	42	169	16
50	59	81	81
95	76	361	1296
80	93	169	441

$\bar{y} = 59$ $SSE = 880$

$SST = 3770$

R^2 (R Square)

: 결정계수 (데이터의 분산을 기반으로 한 평가 지표)

$$R^2 = 1 - \frac{SSE}{SST} = \frac{SSR}{SST}$$

$$R^2 = 1 - \frac{\text{(실제 값 - 예측 값)}^2\text{의 합}}{\text{(실제 값 - 평균 값)}^2\text{의 합}}$$

$$ex)\ R^2 = 1 - \frac{880}{3770} = 1 - 0.233 = 0.767$$

※ MAE, MSE, RMSE 3가지는 평가방법은 0에 가까울수록 우수고 R2방법은 1에 가까울수록 우수하다.

- **다양한 평가 지표로 평가하기위해 회귀 모델을 생성한다.**

```
import pandas as pd
dataset = pd.read_csv('MultipleLinearRegressionData.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct=ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [2])], remainder='passthrough')
X = ct.fit_transform(X)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

- **MAE(Mean Absolute Error) 지표로 회귀 모델을 평가한다.**

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test, y_pred)
```

```
3.225328518828796
```

- **MSE(Mean Squared Error) 지표로 회귀 모델을 평가한다.**

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred)
```

```
19.900226981514965
```

- **RMSE(Root Mean Squared Error) 지표로 회귀 모델을 평가한다.**

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test, y_pred, squared=False)
```

```
4.460967045553572
```

- **R2(결정 계수) 지표로 회귀 모델을 평가한다.**

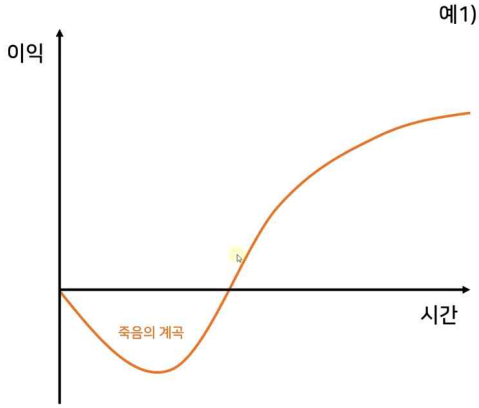
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

```
0.9859956178877446
```

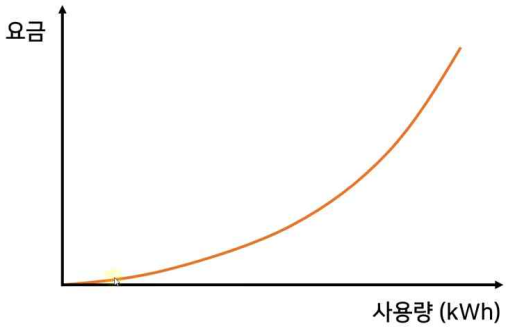
• 다항 회귀 (Polynomial Regression)

기업의 성장 단계나 주택 전기 요금 같은 경우에는 일차 방정식으로 표현하기 힘들다. 이런 경우 다항 회귀 모델을 사용한다.

예1) 기업의 성장 단계



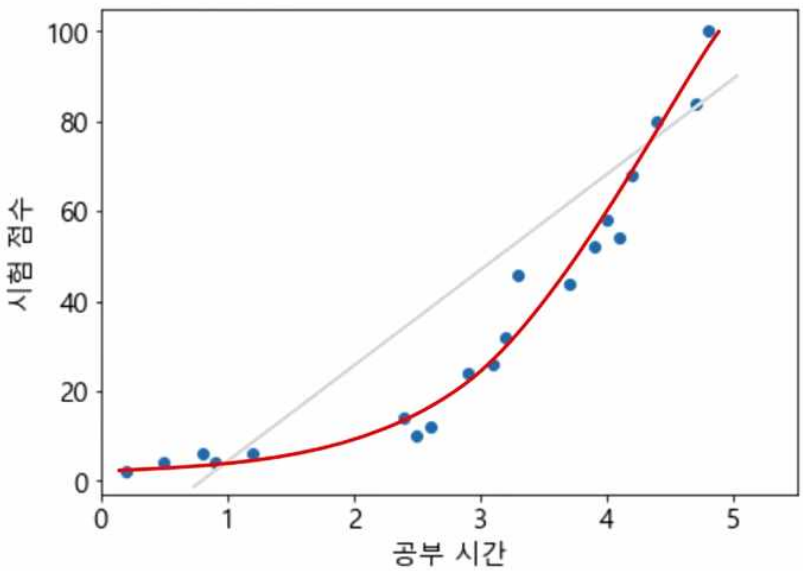
예2) 주택 전기 요금



Simple Linear Regression 단순 선형 회귀	$y = mx + b$
Multiple Linear Regression 다중 선형 회귀	$y = b + m_1x_1 + m_2x_2 + \cdots + m_nx_n$
Polynomial Regression 다항 회귀	$y = b + m_1x + m_2x^2 + \cdots + m_nx^n$

공부 시간	시험 점수
0.2	2
0.5	4
0.8	6
0.9	4
1.2	6
2.4	14
2.5	10
2.6	12
2.9	24
3.1	26
3.2	32
3.3	46
3.7	44
3.9	52
4	58
4.1	54
4.2	68
4.4	80
4.7	84
4.8	100

데이터를 가장 잘 표현하는 선?



• 단순 선형 회귀(Simple Linear Regression) 실습1

- [구글]-[Polynomial regression fit] 검색 후 첫 번째 링크로 이동 후 다양한 옵션으로 다항 회귀를 연습할 수 있다.
- 새로운 파일 '03.Polynomial Regression.ipynb'를 생성한 후 필요한 라이브러리들을 import한다.

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

- csv 파일(우등생들의 시험점수)를 읽어 데이터 세트 생성 후 독립변수(X)와 종속변수(y) 값들을 각각의 변수에 저장한다.

```
dataset = pd.read_csv('PolynomialRegressionData.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

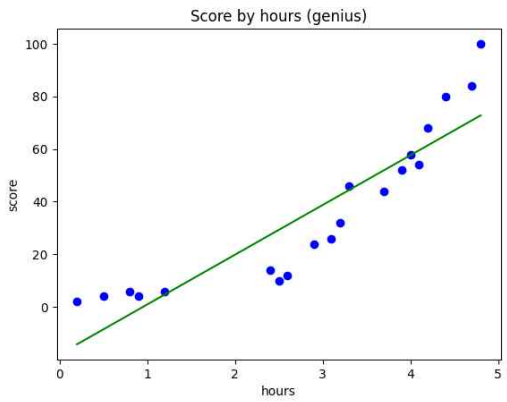
- 전체 데이터로 학습한 후 선형 회귀 모델을 생성한다.

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X, y)
```

▾ LinearRegression
LinearRegression()

- 전체 데이터를 그래프 출력으로 시각화한다.

```
plt.scatter(X, y, color='blue')
plt.plot(X, reg.predict(X), color='green')
plt.title('Score by hours (genius)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- 전체 데이터를 통한 선형 회귀 모델을 평가해 본다.

```
reg.score(X, y)
```

0.8169296513411765

• 다항 회귀(Polynomial Regression) 실습2

▪ **sklearn는 다항 회귀 클래스가 제공되지 않으므로 독립변수(X)를 다항 회귀에 필요한 독립변수로 변환해 주어야 한다.**

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)  #2차 방정식 객체 생성
X_poly = poly_reg.fit_transform(X)
X_poly[:5]  #X의 값이 X의 0승, X의 1승, X의 2승으로 변환
```

```
array([[1.  , 0.2 , 0.04],
       [1.  , 0.5 , 0.25],
       [1.  , 0.8 , 0.64],
       [1.  , 0.9 , 0.81],
       [1.  , 1.2 , 1.44]])
```

▪ **get_feature_names_out() 메서드를 호출하면 X 특성이 각각 어떤 입력의 조합으로 만들어졌는지 알려준다.**

```
poly_reg.get_feature_names_out()
```

```
array(['1', 'x0', 'x0^2'], dtype=object)
```

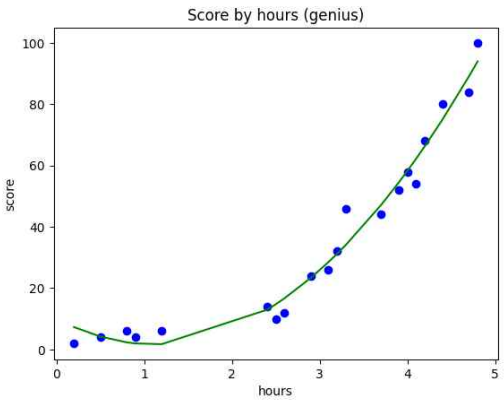
▪ **독립변수(X_poly)를 이용하여 학습한 후 다항 회귀 모델을 생성한다.**

```
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)
```

▾ LinearRegression
LinearRegression()

▪ **변환된 독립변수(X_poly)와 종속변수(y)를 이용하여 다항 회귀 모델을 시각화한다.**

```
plt.scatter(X, y, color='blue')
plt.plot(X, lin_reg.predict(poly_reg.fit_transform(X)), color='green')
plt.title('Score by hours (genius)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- 곡선을 부드럽게 출력하기 위해 독립변수(X)을 일정하게 증가시켜 새로운 값을 생성해 준다. (Numpy 이용)

```
X_range = np.arange(min(X), max(X), 0.1)  #X축을 최솟값~최댓값 0.1단위로 잘라서 데이터를 생성
X_range
```

```
array([0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7,
       2.8, 2.9, 3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7])
```

- 새로운 X_range의 형태를 출력하면 1차원 배열이다.

```
X_range.shape
```

```
(46,)
```

- 원본 X의 형태를 출력하면 을 출력하면 2차원 배열이다.

```
X.shape
```

```
(20, 1)
```

- 독립변수(X_range)를 reshape함수를 이용하여 2차원 배열로 변환한다.

```
X_range=X_range.reshape(-1, 1)  #row은 전체 데이터 수: len(X_range) 또는 -1, column의 1로 지정
X_range.shape
```

```
(46, 1)
```

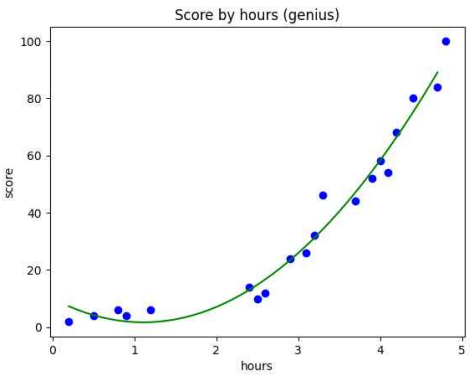
- 독립변수(X_range) 값을 5개 출력해 보면 2차원 배열로 변경되었다.

```
X_range[:5]
```

```
array([[0.2], [0.3], [0.4], [0.5],[0.6]])
```

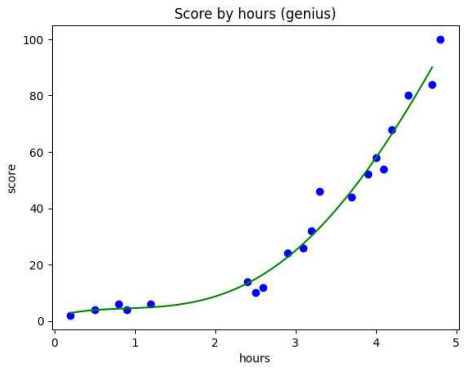
- X값을 X_range값으로 변경한 후 그래프를 출력하면 부드러운 곡선이 출력된다.

```
plt.scatter(X, y, color='blue')
plt.plot(X_range, lin_reg.predict(poly_reg.fit_transform(X_range)), color='green')
plt.title('Score by hours (genius)')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- degree를 4로 변경 후 [Run All]를 실행하면 예측 값이 조금 더 정확한 곡선이 출력된다.

```
...
poly_reg = PolynomialFeatures(degree=4)
```



- 2시간을 공부했을 때 선형 회귀 모델의 예측 값을 출력해 본다.

```
reg.predict([[2]])
```

```
array([19.85348988])
```

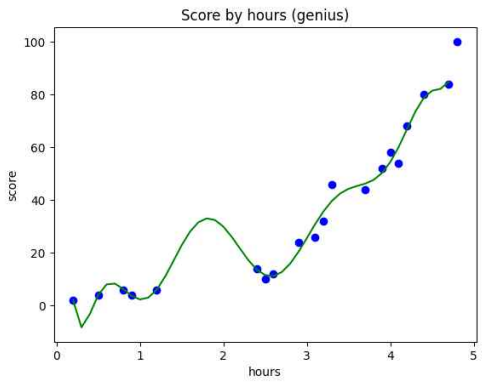
- 2시간을 공부했을 때 다항 회귀 모델의 예측 값을 출력해 본다.

```
lin_reg.predict(poly_reg.fit_transform([[2]]))
```

```
array([8.70559135])
```

- degree를 10로 변경 후 [Run All]를 실행하여 새로운 다항 회귀 모델을 생성해 본다. 훈련에서는 높은 예측률을 가지지만 실제(2시간)의 예측값은 엉터리 값(과대적합, 과소적합)이 예측될 수 있다.

```
...
poly_reg = PolynomialFeatures(degree=10)
```



- 다항 회귀 모델의 예측 점수를 출력해 본다. (선형 회귀 모델점수: 0.8169296513411765)

```
lin_reg.score(X_poly, y)
```

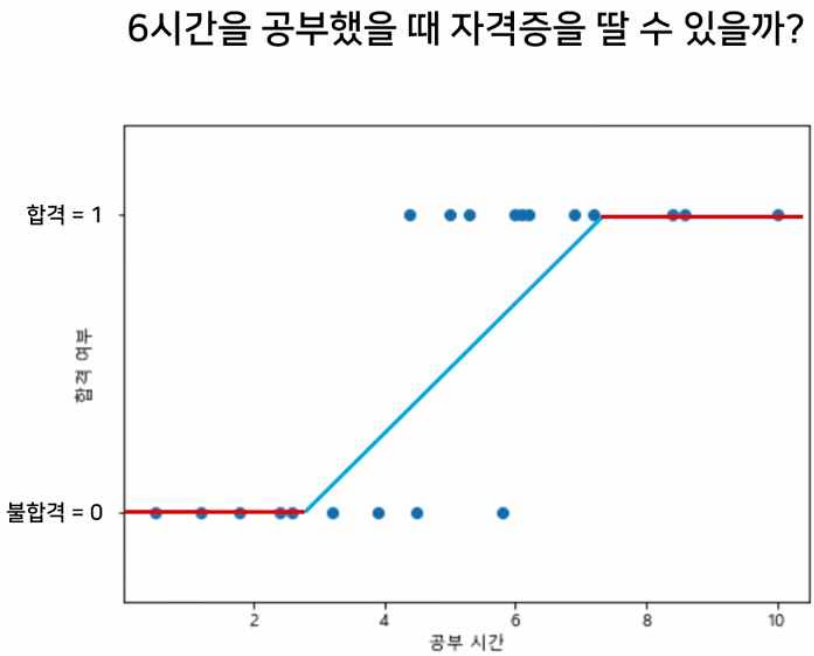
```
0.9782775579000046
```

• 로지스틱 회귀 (Logistic Regression)

선형 회귀 방식을 분류에 적용한 알고리즘으로 데이터가 어떤 범주에 속할 확률을 0~1 사이의 값으로 예측해서 더 높은 범주에 속하는 쪽으로 분류한다. (범주: True/False, Yes/No, 합격/불합격) 적용 예제로는 스팸 메일, 은행 대출, 악성 여부, 고객의 제품 구매 의사 등이 있다. 지도 학습의 분류(Classification)에 속한다.

- **아래 공부시간에 따른 자격증시험 합격 여부를 표현하는 그래프이다. 합격이면 1에 수렴하고 불합격인 경우 0에 수렴한다.**

공부 시간	합격 여부
0.5	불합격
1.2	불합격
1.8	불합격
2.4	불합격
2.6	불합격
3.2	불합격
3.9	불합격
4.4	합격
4.5	불합격
5	합격
5.3	합격
5.8	불합격
6	합격
6.1	합격
6.2	합격
6.9	합격
7.2	합격
8.4	합격
8.6	합격
10	합격



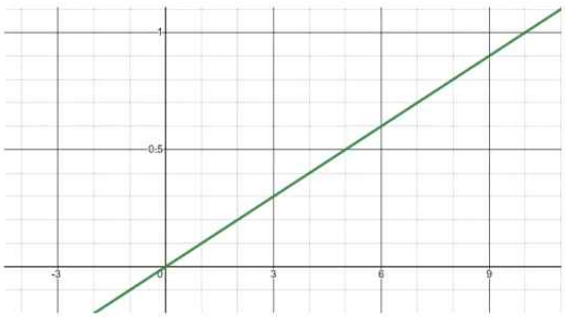
- **첫 번째는 선형 회귀함수, 두 번째는 시그모이드(Sigmoid)함수 또는 로지스틱(Logistic)함수 그래프이다.**

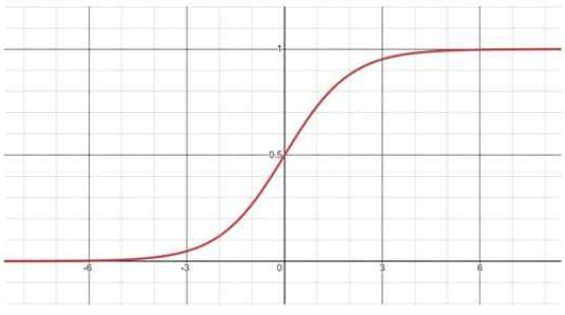
Sigmoid Function

$$y = mx + b$$

$$P = \frac{1}{1 + e^{-y}}$$

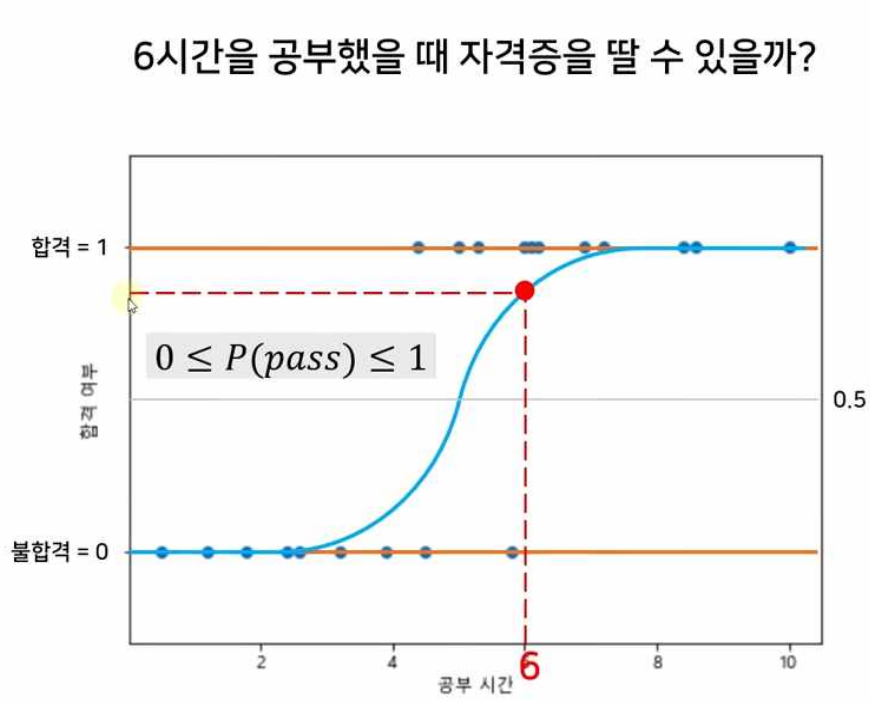
$$ln\left(\frac{P}{1-P}\right) = mx + b$$





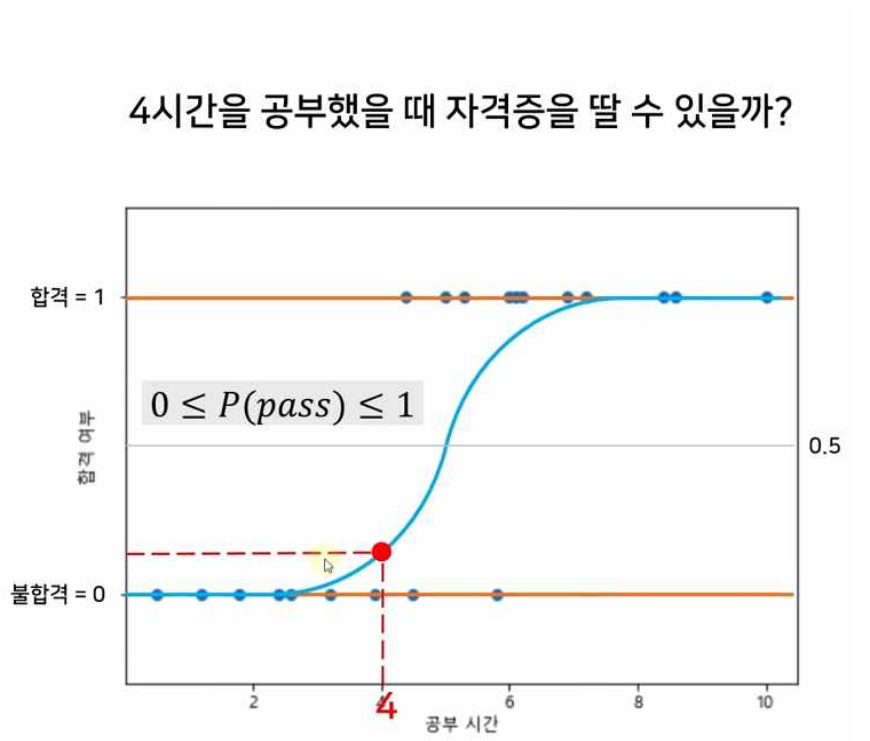
- **6시간 공부했을 때 합격 여부는 0.5보다 크므로 합격으로 예측한다.**

공부 시간	합격 여부
0.5	불합격
1.2	불합격
1.8	불합격
2.4	불합격
2.6	불합격
3.2	불합격
3.9	불합격
4.4	합격
4.5	불합격
5	합격
5.3	합격
5.8	불합격
6	합격
6.1	합격
6.2	합격
6.9	합격
7.2	합격
8.4	합격
8.6	합격
10	합격



- **4시간 공부했을 때 합격 여부는 0.5보다 작으므로 불합격으로 예측한다.**

공부 시간	합격 여부
0.5	불합격
1.2	불합격
1.8	불합격
2.4	불합격
2.6	불합격
3.2	불합격
3.9	불합격
4.4	합격
4.5	불합격
5	합격
5.3	합격
5.8	불합격
6	합격
6.1	합격
6.2	합격
6.9	합격
7.2	합격
8.4	합격
8.6	합격
10	합격



• 로지스틱 회귀 실습

▪ 새로운 파일 '04.Logistic Regression.ipynb'를 생성한 후 필요한 라이브러리를 import한다.

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

▪ csv(공부시간 별 자격증 합격여부)파일을 데이터 세트로 읽어 독립변수(X)와 종속변수(y) 값을 저장한다.

```
dataset = pd.read_csv('LogisticRegressionData.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values
```

▪ 실제 데이터를 훈련 데이터 세트와 테스트 데이터 세트로 분리한다.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

▪ 로지스틱 회귀 객체 생성한 후 훈련 데이터를 이용하여 학습한 로지스틱 회귀 모델을 생성한다.

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

▪ 6시간 공부했을 때 1이 출력되므로 합격이 예측된다.

```
classifier.predict([[6]])
```

```
array([1], dtype=int64)
```

```
classifier.predict_proba([[6]])
```

```
array([[0.14150735, 0.85849265]])  #불합격 확률: 14%, 합격 확률:85%
```

▪ 4시간 공부했을 때 0이 출력되므로 불합격이 예측된다.

```
classifier.predict([[4]])
```

```
array([0], dtype=int64)
```

```
classifier.predict_proba([[4]])
```

```
array([[0.6249966, 0.3750034]])  #불합격 확률: 62%, 합격 확률:37%
```

- 테스트 세트 데이터를 가지고 결과(y_pred)를 예측해 본다.

```
y_pred = classifier.predict(X_test)
y_pred
```

```
array([1, 0, 1, 1], dtype=int64)
```

- 실제 데이터 결과(y_test)를 출력해 본다. (4번째 데이터 값은 예측 결과가 틀리다.)

```
y_test
```

```
array([1, 0, 1, 0], dtype=int64)
```

- 테스트 데이터 세트의 공부 시간(X_test)을 출력해 본다. (4.5시간 예측 결과가 틀리게 출력되었다.)

```
X_test
```

```
array([[ 8.6], [ 1.2], [10. ], [ 4.5]])
```

- 테스트 세트의 모델을 평가해 본다. (전체 테스트 세트 4개 중에서 분류 예측을 맞힌 개수 3개)

```
classifier.score(X_test, y_test)
```

```
0.75
```

• 로지스틱 회귀 데이터 시각화 (훈련 세트)

- 부드러운 곡선을 위해 X 범위를 세분화하여 생성한다. (X의 최솟값~최댓값을 0.1단위로 잘라서 생성)

```
X_range = np.arange(min(X), max(X), 0.1)
X_range
```

```
array([0.5, 0.6, 0.7, 0.8, 0.9, 1. , 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 ,.. 9.6, 9.7, 9.8, 9.9])
```

- 로지스틱 함수 공식에 X_range 값을 대입하여 p의 값을 구한다. p = 1/(1+e^-y), y=mx + b

```
p = 1/(1 + np.exp(-(classifier.coef_ * X_range + classifier.intercept_)))
p
```

```
array([[0.01035705, 0.01161247, 0.01301807, 0.0145913 , 0.01635149,...
        0.99713321, 0.99744558, 0.997724  , 0.99797213, 0.99819325]])
```

- p의 shape를 출력해 보면 2차원 배열이다.

```
p.shape
```

```
(1, 95)
```

- X_range의 shape를 출력해 보면 1차원 배열이다.

```
X_range.shape
```

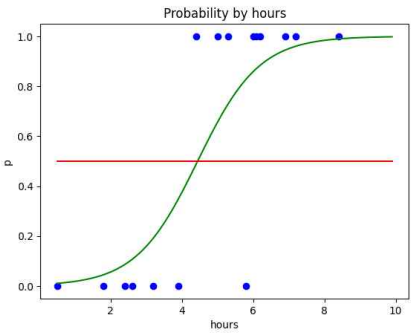
```
(95,)
```

- p를 2차원 배열에서 reshape 함수를 이용하여 1차원 배열로 변경한다.

```
p = p.reshape(-1)  #-1 = len(p) 모든 p의 개수를 리턴 한다.
p.shape
```

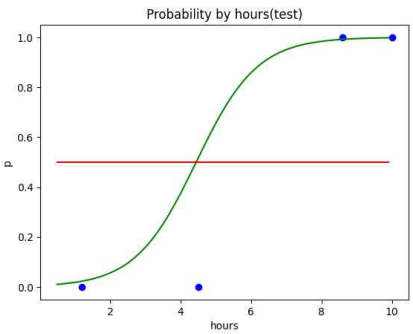
```
(95,)
```

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_range, p, color='green')
plt.plot(X_range, np.full(len(X_range), 0.5), color='red')  #X_range개수만큼 0.5로 가득 찬 배열
plt.title('Probability by hours')
plt.xlabel('hours')
plt.ylabel('p')
plt.show()
```



- 로지스틱 회귀 데이터 시각화 (테스트 세트)

```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_range, p, color='green')
plt.plot(X_range, np.full(len(X_range), 0.5), color='red')  #X_range개수만큼 0.5로 가득 찬 배열
plt.title('Probability by hours(test)')
plt.xlabel('hours')
plt.ylabel('p')
plt.show()
```



- 4.5시간 공부했을 때 확률 (모델에서는 51% 확률로 합격 예측, 실제로는 불합격)

```
classifier.predict_proba([[4.5]])
```

```
array([[0.48310686, 0.51689314]])
```

• 혼동 행렬 (Confusion Matrix)

▪ confusion_matrix 함수를 이용하여 테스트 세트(X_test)와 예측 결과(y_pred)를 혼동 행렬로 변경한다.

```
from sklearn.metrics import confusion_matrix
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1, 1],
       [0, 2]], dtype=int64)
```

True Negative (TN) 불합격을 예측했는데 실제로 불합격한 수 (1)	False Positive (FP) 합격으로 예측했는데 실제로 불합격한 수 (1)
False Negative (FN) 불합격을 예측했는데 실제로 합격한 수 (0)	True Positive (TP) 합격으로 예측했는데 실제로 합격한 수 (2)

▪ confusion_matrix 함수를 이용하여 훈련 세트(X_train)와 예측 결과(y_pred)를 혼동 행렬로 변경한다.

```
from sklearn.metrics import confusion_matrix
y_pred = classifier.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
cm
```

```
array([[6, 1],
       [1, 8]], dtype=int64)
```

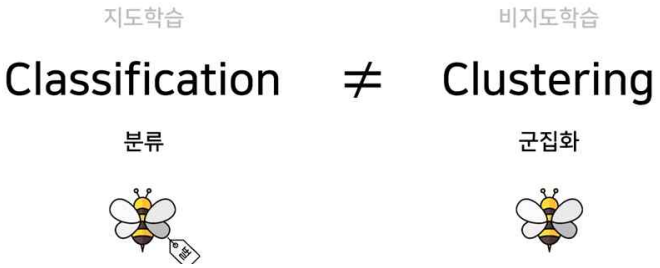
True Negative (TN) 불합격을 예측했는데 실제로 불합격한 수 (6)	False Positive (FP) 합격으로 예측했는데 실제로 불합격한 수 (1)
False Negative (FN) 불합격을 예측했는데 실제로 합격한 수 (1)	True Positive (TP) 합격으로 예측했는데 실제로 합격한 수 (8)

• K-평균 (K-Means)

• 비지도 학습 (Unsupervised Learning)

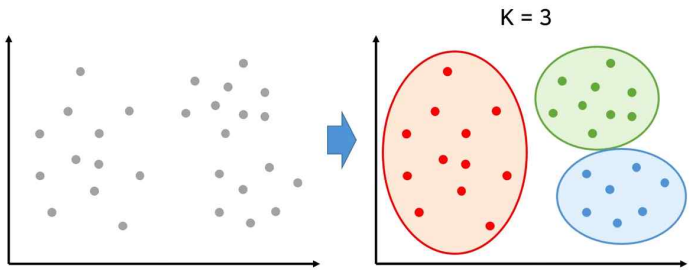
정답이 없는 데이터를 통해 유의미한 패턴/구조를 발견하여 유사한 특징을 가지는 데이터들을 그룹화 하여 학습하는 방법으로 대표적인 학습으로 군집화(Clustering)가 있다. 예제로는 고객 세분화, 소셜 네트워크 분석, 기사 그룹 분류 등이 있다.

Classification은 지도학습으로 정답이 존재하며 Clustering은 비 지도학습으로 정답이 존재하지 않는다.



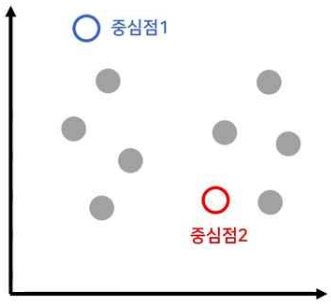
• K-Menas

데이터를 K개의 클러스터(그룹)로 군집화 하는 비지도 학습 방법 중 클러스터의 대표적인 알고리즘이다. 각 데이터로부터 이들이 속한 클러스터의 중심점(Centroid)까지의 평균 거리를 계산하는 방식으로 동작한다.

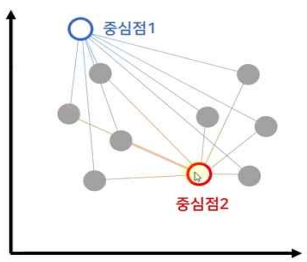


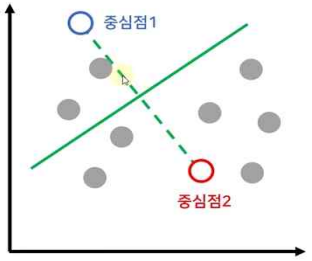
• K-Means 동작순서

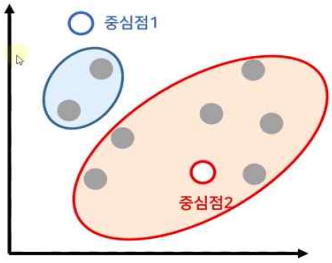
1. K값 설정 (K=2로 설정)
2. 지정한 K개만큼의 랜덤 중심점 좌표 설정



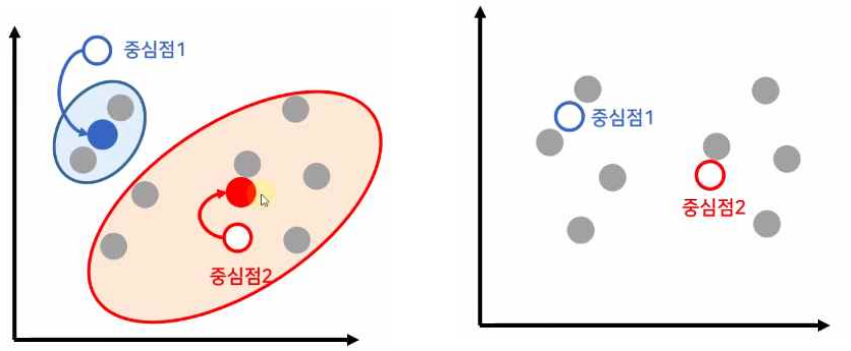
3. 모든 데이터로부터 가장 가까운 중심점 선택



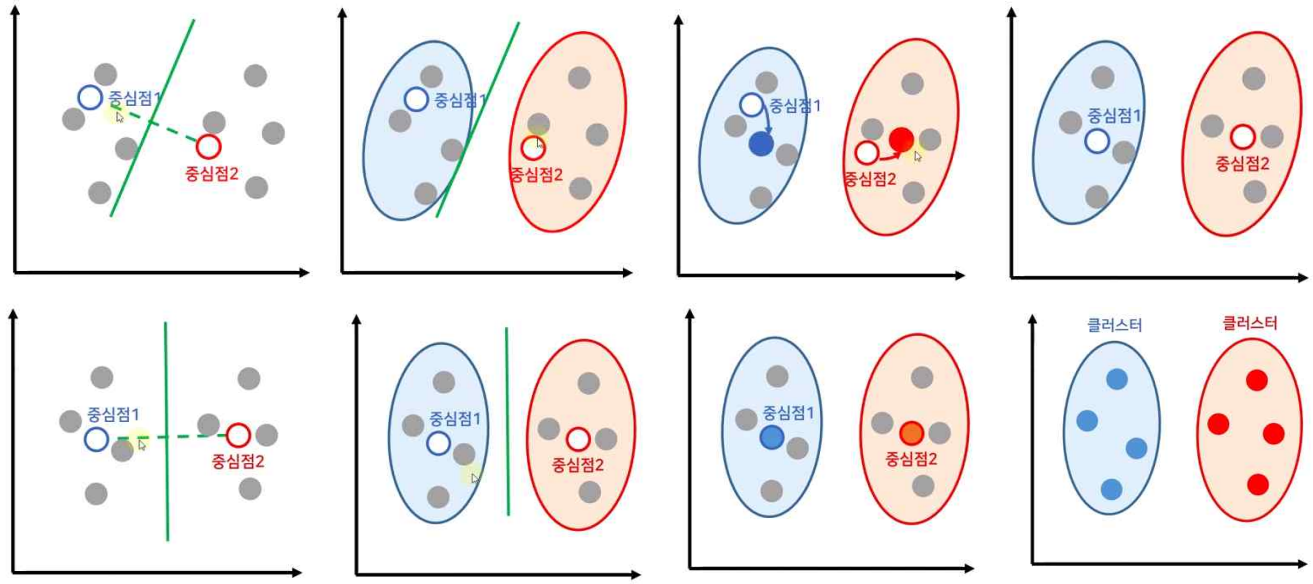




4. 데이터들의 평균 중심으로 중심점 이동

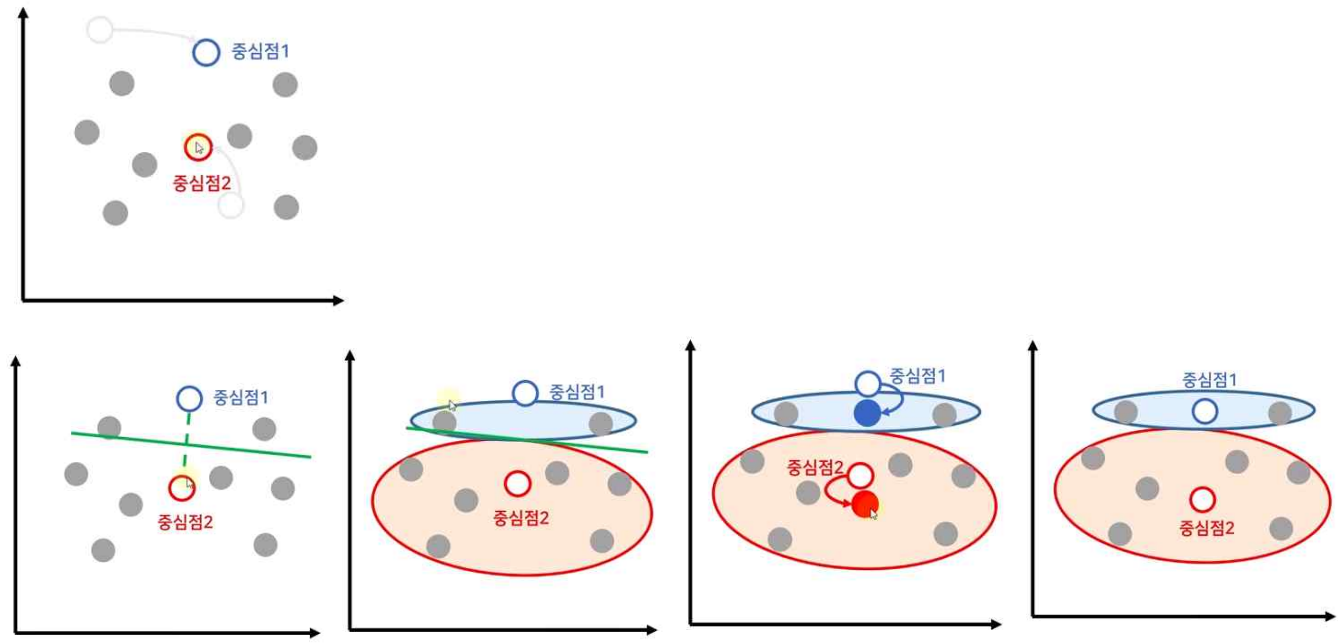


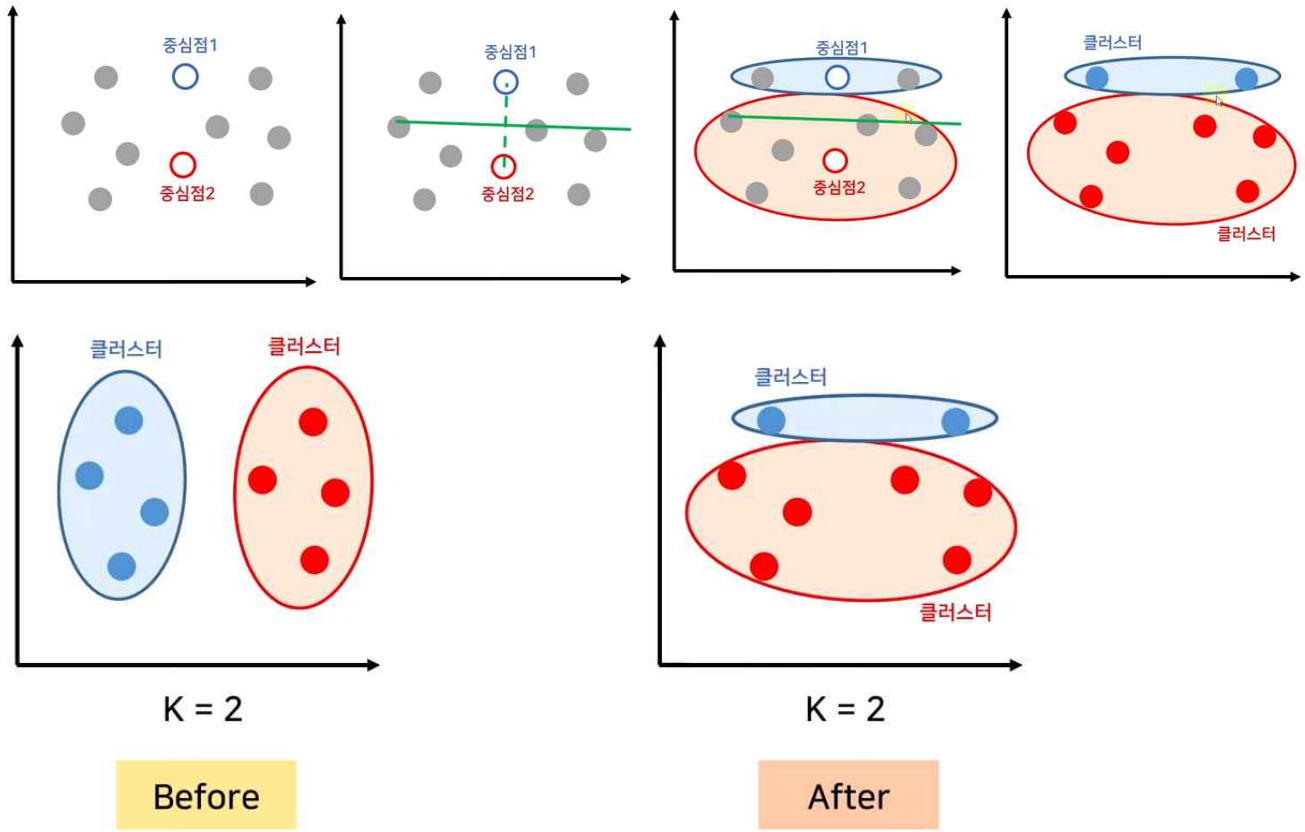
5. 중심점이 더 이상 이동되지 않을 때까지 3번(모든 데이터로부터 가장 가까운 중심점 선택) 반복



- **Random Initialization Trap (중심점 무작위 선정 문제)**

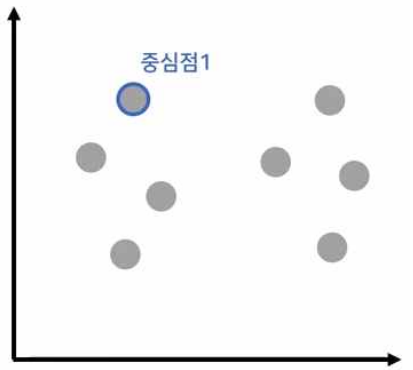
랜덤으로 중심점 좌표 설정을 설정하면 매번 클러스터의 모양이 바뀔 수도 있고 중심점까지의 거리가 짧으면 클러스터 작업이 제대로 이루어지 않는다.



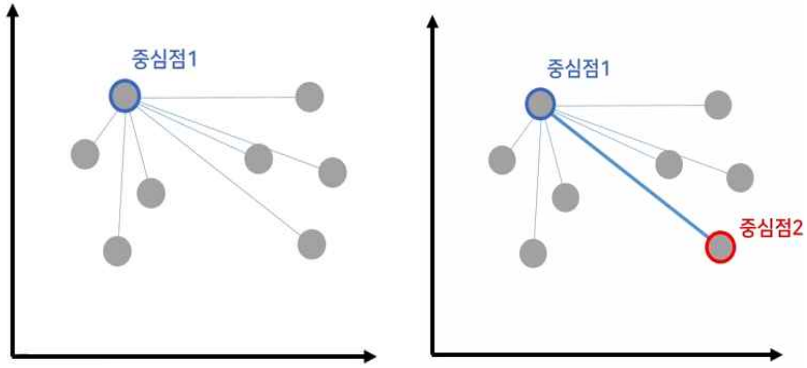


- **K-Means++**

1. 데이터 중에서 랜덤으로 1개를 중심점으로 선택



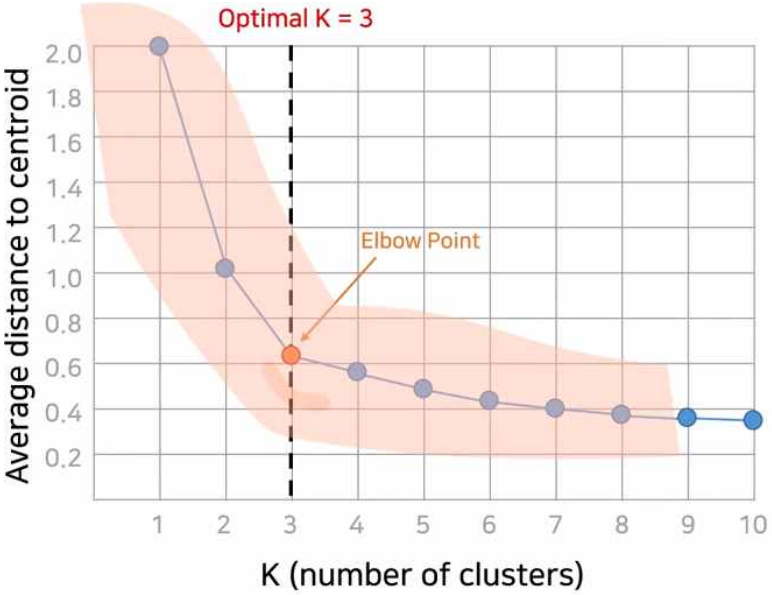
2. 나머지 데이터로부터 중심점까지의 거리 계산
3. 중심점과 가장 먼 지점의 데이터를 다음 중심점으로 선택
4. 중심점이 K개가 될 때까지 반복



5. K-Means 전통적인 방식으로 진행

- **Optimal K (최적의 K)**

최적의 K값을 구할 때는 Elbow Method(엘보우 방법)을 이용하여 구할 수 있다.

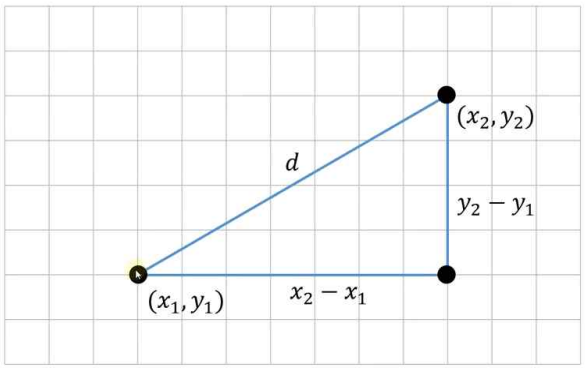


1) K 변화에 따른 중심점까지의 평균 거리 비교
2) 경사가 완만해지는 지점의 K 선정

- **데이터 유사도 구하는 방법**

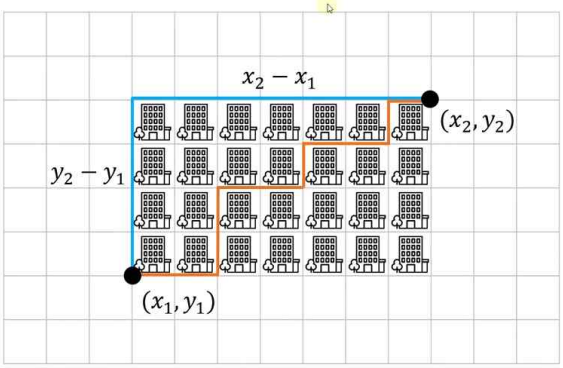
Euclidean Distance

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



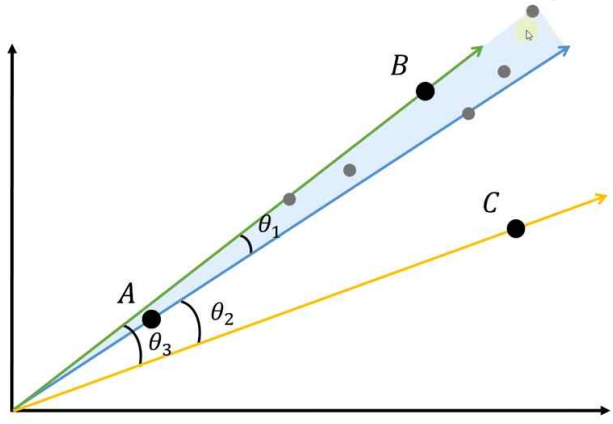
Manhattan Distance

$$d = |x_2 - x_1| + |y_2 - y_1|$$



Cosine Similarity

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



- **K-평균 실습**

  - **새로운 파일 '05.K-Means.ipynb'를 생성한 후 필요한 라이브러리를 import 한다.**

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

  - **KMeanData.csv 파일을 읽어 데이터세트에 저장 후 5개의 데이터(시간과 점수)를 출력한다.**

```
dataset = pd.read_csv('KMeansData.csv')
dataset[:5]
```

	hour	score
0	7.33	73
1	3.71	55
2	3.43	55
3	3.06	89
4	3.33	79

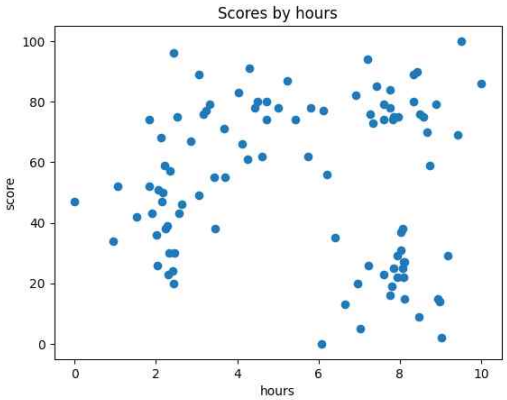
  - **K-Means는 정답이 없는 비지도 학습이므로 종속변수(y)가 없다. 그러므로 독립변수(X)의 모든 값들만 가져온다.**

```
X = dataset.iloc[:, :].values  # X = dataset.values와 결과가 같다.
X[:5]
```

```
array([[ 7.33, 73.  ],
       [ 3.71, 55.  ],
       [ 3.43, 55.  ],
       [ 3.06, 89.  ],
       [ 3.33, 79.  ]])
```

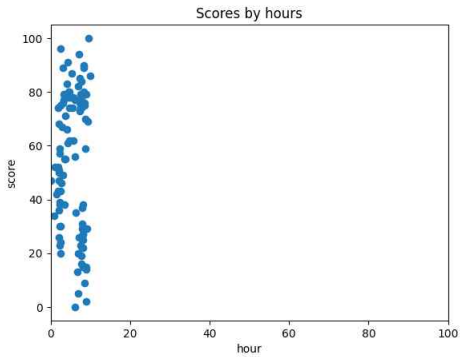
  - **전체 데이터 분포를 시각화로 확인해 본다.**

```
plt.scatter(X[:, 0], X[:, 1])  # x축:hour, y축: score
plt.title('Scores by hours')
plt.xlabel('hours')
plt.ylabel('score')
plt.show()
```



- **중심점과 각 점까지의 거리를 구하기 위해서는 x축과 y축의 범위(limit)값을 같게 한다.**

```
plt.scatter(X[:,0], X[:, 1])
plt.xlim(0, 100)
plt.xlim(0, 100)
plt.title('Scores by hours')
plt.xlabel('hour')
plt.ylabel('score')
plt.show()
```



- **피쳐 스케일링(Feature Scalling) 함수를 이용하여 보기 좋게 x축과 y축 범위를 조정한다.**

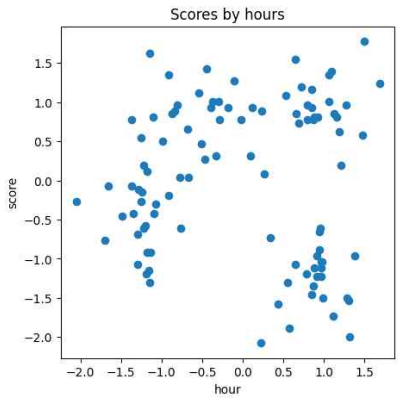
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
X[:5]
```

```
array([[ 0.68729921,  0.73538376],
       [-0.66687438,  0.04198891],
       [-0.77161709,  0.04198891],
       [-0.9100271 ,  1.35173473],
       [-0.8090252 ,  0.96651537]])
```

```
#스케일링 전 데이터
array([[ 7.33, 73.  ],
       [ 3.71, 55.  ],
       [ 3.43, 55.  ],
       [ 3.06, 89.  ],
       [ 3.33, 79.  ]])
```

- **스케일 된 데이터(X)를 시각화(정사각형 그래프)로 확인해 본다.**

```
plt.figure(figsize=(5, 5))
plt.scatter(X[:,0], X[:, 1])
plt.title('Scores by hours')
plt.xlabel('hour')
plt.ylabel('score')
plt.show()
```

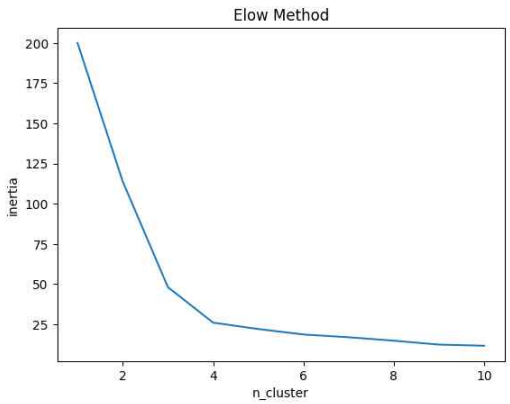


- 몇 개의 cluster로 나눌지 엘보우 방식(Elbow Method)을 이용해 최적의 K값을 찾는다.

```
from sklearn.cluster import KMeans
inertia_list = []  #cluster에 속한 점들이 얼마나 가깝게 모여 있는지를 나타내는 값들

for i in range(1, 11):
    #중심점(centroid)을 K-menas++ 방식으로 계산하여 객체를 생성한다.
    kmenas = KMeans(n_clusters=i, init='k-means++', random_state=0, n_init=10)
    kmenas.fit(X)  #학습 후 모델을 생성한다.
    inertia_list.append(kmenas.inertia_)  #각 지점으로 부터 중심점(centroid)까지의 거리의 제곱의합(inertia)

plt.plot(range(1, 11), inertia_list)
plt.title('Elow Method')
plt.xlabel('n_cluster')
plt.ylabel('inertia')
plt.show()
```



- 위 그래프에서 곡선이 원만해지는 지점의 cluster의 수를 최적의 K값으로 설정한다.

```
K = 4
kmeans = KMeans(n_clusters=4, random_state=0, n_init=10)
y_kmeans = kmeans.fit_predict(X)  #X의 점들이 어느 cluster에 속하는지 예측해 본다.
y_kmeans  #X의 점들의 어느 cluster에 속하는지 출력해 본다.
```

```
array([2, 3, 3, 0, 0, 1, 1, 0, 2, 0, 0, 3, 1, 3, 3, 0, 1, 2, 3, 0, 1, 0,
       3, 1, 2, 2, 3, 3, 3, 3, 1, 1, 3, 0, 2, 2, 3, 0, 0, 0, 3, 1, 2, 3,
       3, 2, 1, 0, 1, 1, 2, 0, 1, 1, 0, 0, 0, 0, 3, 1, 1, 2, 2, 2, 2, 1,
       1, 0, 1, 2, 3, 2, 2, 2, 3, 3, 3, 3, 0, 2, 1, 2, 1, 1, 2, 0, 3, 1,
       2, 3, 0, 1, 0, 2, 3, 2, 2, 0, 1, 3])
```

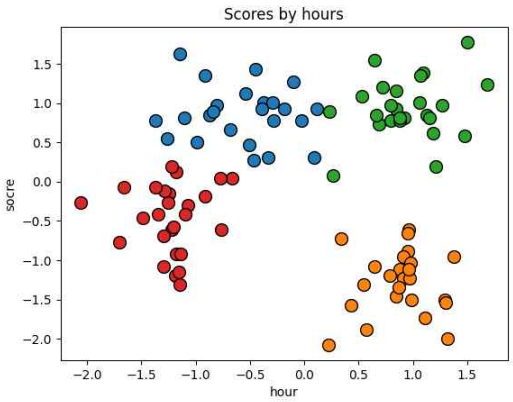
- cluster 0 그룹에 속한 점들의 x, y 좌표들을 출력한다.

```
X[y_kmeans==0, : ]  # X[y_kmeans==0, 0] x좌표, X[y_kmeans==0, 1] y좌표
```

```
array([[-0.9100271 ,  1.35173473],
       [-0.8090252 ,  0.96651537],
       [ 0.09251026,  0.31164246],
       [-0.28531166,  0.77390569],
       ...
       [-0.10201192,  1.27469086]])
```

- **최적의 K를 이용하여 cluster별 점들을 출력해 본다.**

```
for cluster in range(4):
    plt.scatter(X[y_kmeans==cluster, 0], X[y_kmeans==cluster, 1], s=100, edgecolor='black')
plt.title('Scores by hours')
plt.xlabel('hour')
plt.ylabel('socre')
plt.show()
```



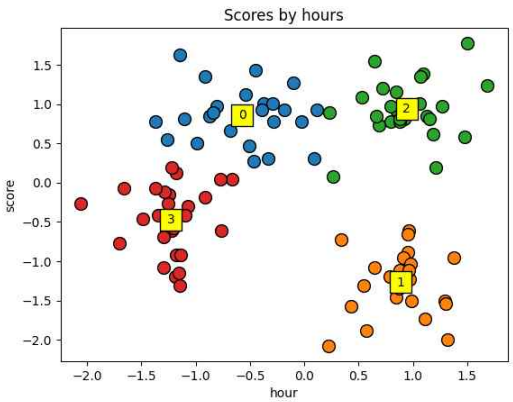
- **각 cluster들(4개) 의 중심점(centroid) 좌표들을 구한다.**

```
centers = kmeans.cluster_centers_
centers
```

```
array([[-0.57163957,  0.85415973],
       [ 0.8837666 , -1.26929779],
       [ 0.94107583,  0.93569782],
       [-1.22698889, -0.46768593]])
```

- **중심점(centroid)를 사각형으로 출력하고 cluster 번호도 텍스트로 출력한다.**

```
for cluster in range(4):
    plt.scatter(X[y_kmeans==cluster, 0], X[y_kmeans==cluster, 1], s=100, edgecolor='black')
    plt.scatter(centers[cluster, 0], centers[cluster, 1],
                s=300, edgecolor='black', color='yellow', marker='s')
    plt.text(centers[cluster, 0], centers[cluster, 1], cluster, va='center', ha='center')
plt.title('Scores by hours')
plt.xlabel('hour')
plt.ylabel('score')
plt.show()
```



- **Feature Scalling 한 독립변수(X)의 값들을 다시 원래 값으로 복원한다.**

```
X_org = sc.inverse_transform(X)
X_org[:5]
```

```
array([
       [ 7.33, 73.  ],
       [ 3.71, 55.  ],
       [ 3.43, 55.  ],
       [ 3.06, 89.  ],
       [ 3.33, 79.  ]
])
```

- **중심점(Centroid)의 값들도 원래 데이터로 복원하다.**

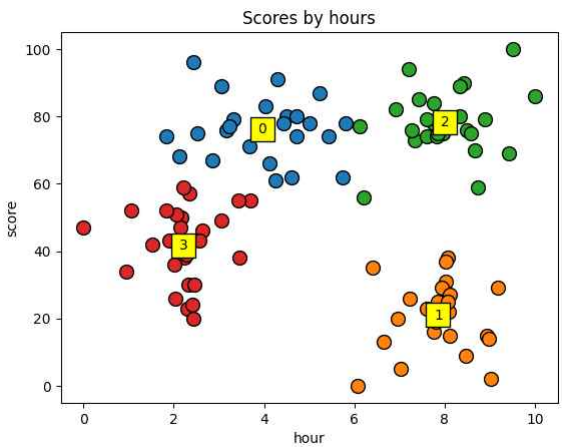
```
centers_org = sc.inverse_transform(centers)
centers_org
```

```
array([
       [ 3.96458333, 76.08333333],
       [ 7.8552    , 20.96      ],
       [ 8.0084    , 78.2       ],
       [ 2.21269231, 41.76923077]
])
```

- **원래대로 복원된 데이터들로 시각화 작업을 한다.**

```
for cluster in range(4):
    plt.scatter(X_org[y_kmeans==cluster, 0], X_org[y_kmeans==cluster, 1],
                s=100, edgecolor='black')
    plt.scatter(centers_org[cluster,0], centers_org[cluster,1],
                s=300, edgecolor='black', color='yellow', marker='s')
    plt.text(centers_org[cluster,0], centers_org[cluster,1], cluster, va='center', ha='center')

plt.title('Scores by hours')
plt.xlabel('hour')
plt.ylabel('score')
plt.show()
```



• 퀴즈

결혼식장에서 피로연의 식수 인원을 올바르게 예측하지 못하여 버려지는 음식으로 고민이 많다고 합니다. 현재까지 진행된 결혼식에 대한 결혼식 참석 인원과 그 중에서 식사를 하는 인원의 데이터가 제공될 때, 아래 각 문항에 대한 코드를 작성하시오.

• 퀴즈 실습

▪ **새로운 파일 '06.Quiz.ipynb'를 생성한 후 필요한 라이브러리를 import 한다.**

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

▪ **데이터 파일로부터 데이터를 읽어 와서 결혼식 참석 인원(total), 식수 인원(reception)을 각각의 변수로 저장한다.**

```
dataset = pd.read_csv('QuizData.csv')
dataset.head()
```

```
X = dataset.iloc[:, :-1].values
y= dataset.iloc[:, -1].values
X[:5], y[:5]
```

```
(array([[118],
        [253],
        [320],
        [ 94],
        [155]], dtype=int64),
 array([ 62, 148, 201,  80,  92], dtype=int64))
```

▪ **전체 데이터를 훈련 세트와 테스트 세트로 분리한다. 이때 비율은 75:25으로 한다.**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
```

▪ **훈련 세트를 이용하여 단순 선형 회귀(Simple Linear Regression) 모델을 생성한다.**

```
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

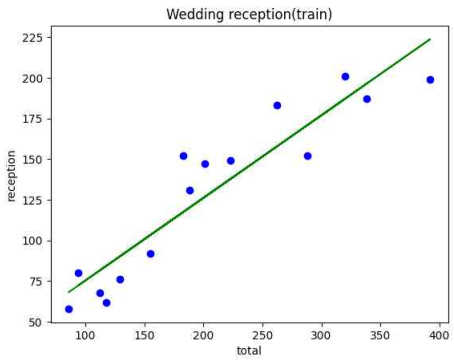
▪ **독립변수(X_train), 종속변수(y_train) 값들을 출력한다.**

```
X_train, y_train
```

```
(array([[262],
        ...
        [ 86]], dtype=int64),
 array([183, 147,  68,  92, 201, 131,  76, 152, 187, 152, 199,  80,  62,
        149,  58], dtype=int64))
```

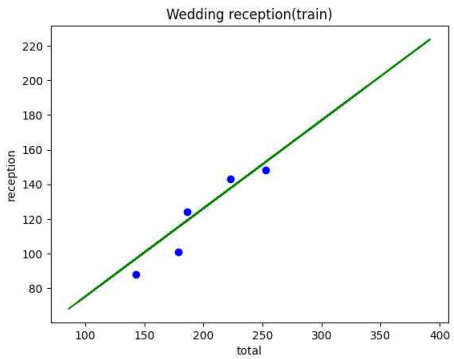
- **데이터 시각화(훈련 세트) 코드를 작성한다.**

```
plt.scatter(X_train, y_train, color='blue')
plt.plot(X_train, reg.predict(X_train), color='green')
plt.title('Wedding reception(train)')
plt.xlabel('total')
plt.ylabel('reception')
plt.show()
```



- **데이터 시각화(테스트 세트) 코드를 작성한다.**

```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_train, reg.predict(X_train), color='green')
plt.title('Wedding reception(test)')
plt.xlabel('total')
plt.ylabel('reception')
plt.show()
```



- **훈련 세트, 테스트 세트에 대해 가각 모델 평가 점수를 구한다.**

```
reg.score(X_train, y_train), reg.score(X_test, y_test)
```

```
(0.8707088403321211, 0.8634953212566615)
```

- **결혼식 참석 인원이 300명일 때 예상되는 식수 인원을 구한다.**

```
total = 300
y_pred = reg.predict([[total]])
print(f'결혼식 인원 {total}명에 대한 예상 식수 인원은 {np.round(y_pred[0]).astype(int)}입니다.')
```

```
결혼식 인원 300명에 대한 예상 식수 인원은 177입니다.
```

• 영화 추천 시스템 프로젝트

• Kaggle이란?

Kaggle은 데이터 분석 및 머신러닝에 대한 학습 플랫폼이자, 경쟁할 수 있는 플랫폼입니다. 기업, 기관 또는 특정 사용자가 데이터를 첨부해서 문제를 제출하면 Kaggle 사용자 누구나 문제에 대한 답을 제출할 수 있다.

• TMDB 5000 Movie Dataset이란?

TMDB 5000 영화 데이터 세트는 유명한 영화 데이터 정보 사이트인 IMDB의 많은 영화 중 주요 5000개 영화에 대한 메타 정보를 새롭게 가공해 Kaggle에서 제공하는 데이터 세트이다.

• Demographic Filtering (인구 통계학적 필터링)

많은 사람들이 일반적으로 좋아하는 영화를 추천하는 방식이다. 예를 들어 영화가 개봉했을 경우 인기가 많아 천만 관객을 돌파하는 등 누가 봐도 좋아할 만한 영화를 추천하는 방식이다. (평점, 평가 수로 추천한다.)

• TMDB 5000 데이터 다운로드

1) [구글]-[TMDB 5000]-[Kaggle 사이트이동]-[상단 Code Tab]-[우측 Hotness 메뉴]-[Most Votes] 선택한다.
2) [Getting Started with a Movie Recommendation System]-[TMDB 5000 Movie Dataset] 선택한다.
3) 'tmdb_5000_credits.csv', 'tmdb_5000_movies.csv' 두 파일을 다운로드 후 프로젝트 폴더로 이동한다.

▪ **새로운 '06.Movie Project.ipynb' 파일을 생성한 후 필요한 라이브러리를 import 한다.**

```
import pandas as pd
import numpy as np
```

▪ **csv 데이터 파일을 읽어 데이터프레임 변수에 각각 저장한다.**

```
df1 = pd.read_csv('tmdb_5000_credits.csv')
df2 = pd.read_csv('tmdb_5000_movies.csv')
```

▪ **인구 통계학적 필터링을 위한 데이터 전처리 작업을 한다.**

1) df1 데이터프레임과 df2 데이터프레임의 병합을 위하여 구조를 확인한다.

```
df1.head()
```

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	John Carter	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

```
df2.head(1)
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	...
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	...

- **df1 데이터프레임의 row(4803)수, column(4)수, df2 데이터프레임의 row(4803), column(20)수를 출력한다.**

```
df1.shape, df2.shape
```

```
((4803, 4), (4803, 20))
```

- **df1 데이터프레임의 'title' 칼럼과 df2 데이터프레임의 'title' 칼럼의 값이 같은지 비교해 본다.**

```
df1['title'].equals(df2['title'])
```

```
True
```

- **df1 데이터프레임의 칼럼 목록을 출력한다.**

```
df1.columns
```

```
Index(['movie_id', 'title', 'cast', 'crew'], dtype='object')
```

2) df1과 df2의 데이터프레임을 병합하기 위하여 df1의 칼럼 중 'movie_id'을 'id'로 변경한다.

```
df1.columns = ['id', 'title', 'cast', 'crew']
```

- **df1 데이터프레임의 칼럼 중 'movie_id'가 'id'로 변경되었는지 데이터프레임을 출력해 본다.**

```
df1.head()
```

	id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...

3) 'title' 칼럼은 df1과 df2에 중복되므로 'title' 칼럼을 제외한 나머지 칼럼으로 데이터프레임(df_temp)을 만든다.

```
df_temp = df1[['id', 'cast', 'crew']]
```

	id	cast	crew
0	19995	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	49529	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...
...	...	...	...

4) df2 데이터프레임과 위에서 만든 df_temp의 데이터프레임을 id가 같은 것 끼리 병합하여 df2 데이터프레임에 저장한다.

```
df2 = df2.merge(df_temp, on='id')
df2.head(1)
```

	spoken_languages	status	tagline	title	vote_average	vote_count	cast	crew
...	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	Avatar	7.2	11800	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

- **평점 가중치 공식**

영화를 평가하기 위해서는 평점도 중요하지만 평가수도 중요하므로 평점에 가중치를 주는 아래 공식이 필요하다.

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m}.R\right) + \left(\frac{m}{v+m}.C\right)$$

where,

- v is the number of votes for the movie;
- m is the minimum votes required to be listed in the chart;
- R is the average rating of the movie; And
- C is the mean vote across the whole report

- **추천 영화 Filtering을 위하여 평점 가중치 공식을 구한다.**

1) 전체영화(4803) 개의 평점의 평균을 구한다.

```
C = df2['vote_average'].mean()
C
```

```
6.092171559442016
```

2) 평가 수(vote_count)가 적은 경우를 제외하기 위하여 하위90%(상위10%) 지점의 값을 구한다.

```
m = df2['vote_count'].quantile(0.9)
m
```

```
1838.4000000000015
```

3) 평가 수(vote_count)가 상위 10% 수(1838.4)보다 크거나 같은 데이터를 필터링해서 새로운 데이터로 copy한다.

```
q_movies = df2.copy().loc[df2['vote_count'] >= m]
q_movies.shape  #데이터 수가 전체(4803)의 10%이다.
```

```
(481, 22)
```

- **가장 적은 평가 수(1840) ~ 가장 많은 평가 수(13752)까지의 데이터만 처리한다.**

```
q_movies['vote_count'].sort_values()
```

```
2585     1840
195      1851
        ...
65      12002
96      13752
Name: vote_count, Length: 481, dtype: int64
```

4) 평점 가중치를 구하는 함수를 작성한다.

```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return(v / (v + m) * R) + (m / (m + v) * C)
```

5) weighted_rating 함수를 적용하여 새로운 'score' 칼럼(axis=1)에 저장한다.

```
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
q_movies.head(2)
```

	spoken_languages	status	tagline	title	vote_average	vote_count	cast	crew	score
...	[{"iso_639_1": "en", "name": "English"}, {"iso...	Released	Enter the World of Pandora.	Avatar	7.2	11800	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...	7.050669
	[{"iso_639_1": "en", "name": "English"}]	Released	At the end of the world, the adventure begins.	Pirates of the Caribbean: At World's End	6.9	4500	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...	6.665696

- **영화 정보 데이터프레임을 'score' 순으로 내림차순 정렬 후 10개의 데이터를 출력한다.**

```
q_movies = q_movies.sort_values('score', ascending=False)
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(10)
```

	title	vote_count	vote_average	score
1881	The Shawshank Redemption	8205	8.5	8.059258
662	Fight Club	9413	8.3	7.939256
65	The Dark Knight	12002	8.2	7.920020
3232	Pulp Fiction	8428	8.3	7.904645
96	Inception	13752	8.1	7.863239
3337	The Godfather	5893	8.4	7.851236
95	Interstellar	10867	8.1	7.809479
809	Forrest Gump	7927	8.2	7.803188
329	The Lord of the Rings: The Return of the King	8064	8.1	7.727243
1990	The Empire Strikes Back	5879	8.2	7.697884

• Content Based Filtering (콘텐츠 기반 필터링)

이 방식은 사용자가 특정 item을 선호하는 경우 그 item과 비슷한 콘텐츠를 가진 다른 item을 추천해 주는 방식이다.

• 줄거리('overview') 기반 추천

영화의 줄거리('overview')분석하여 사용자가 재밌게 관람했던 영화와 비슷한 줄거리 영화를 필터링하여 추천한다.

- **df2 데이터프레임의 5개의 'overview' row를 출력해 본다.**

```
df2['overview'].head(5)
```

```
0    In the 22nd century, a paraplegic Marine is di...
1    Captain Barbossa, long believed to be dead, ha...
2    A cryptic message from Bond’ s past sends him o...
3    Following the death of District Attorney Harve...
4    John Carter is a war-weary, former military ca...
Name: overview, dtype: object
```

- **Bag Of Words(BOW) : 문서의 단어들을 분리하여 몇 번이 나오는지 분리하여 vector로 표현한 것이다.**

문장	I	am	a	boy	girl	결과
I am a body	1	1	1	1	0	(1, 1, 1, 1, 0)
I am a girl	1	1	1	0	1	(1, 1, 1, 0, 1)

문장이 100개이고 모든 문장에서 나온 단어 10,000개 인 경우
100 * 10,000 = 100만개

	단어1	단어2	단어3	단어4	...	단어10000
문장1	1	1	3	0	...	1
문장2	0	4	2	1	...	0
문장3	1	1	3	0	...	1
...	0	0	0	1	...	2
문장100	1	1	3	0	...	1

- sklearn의 BOW 벡터를 만든 클래스

1) CounterVectorizer : 단어의 빈도를 Count하여 BOW 벡터(Vector)로 만든다.
2) TfidfVectorizer : CountVectorizer와 비슷하지만 TF-IDF 방식으로 단어의 가중치를 조정한 BOW 벡터를 만든다.

- **단어의 가중치를 조정하여 BOW 벡터를 생성하는 TfidfVectorizer 클래스를 import 한다.**

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(stop_words='english')  #the, a와 같은 의미 없는 단어를 제외한다.
```

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
ENGLISH_STOP_WORDS
```

```
frozenset({'a',
           'about',
           'above',
...
           'yourself',
           'yourselves'})
```

- **줄거리 분석을 위한 데이터 전처리 작업을 한다.**

1) 'overview'에 null 값이 하나라도 있는지 확인해 본다.

```
df2['overview'].isnull().values.any()
```

```
True
```

2) 'overview'에 null이 있으면 모두 빈값으로 업데이트 해준다. (vector화 작업 시 null이 있는 경우 오류발생)

```
df2['overview'] = df2['overview'].fillna('')
```

3) 'overview'를 BOM vector 데이터(행렬)로 변경하여 저장한다. (문서:4803, 단어:20978)

```
tfidf_matrix = tfidf.fit_transform(df2['overview'])
tfidf_matrix.shape
```

```
(4803, 20978)
```

- **전체 데이터 수를 출력한다.**

```
tfidf_matrix
```

```
<4803x20978 sparse matrix of type '<class 'numpy.float64'>'
        with 125840 stored elements in Compressed Sparse Row format>
```

- **cosine 유사도 값을 구하기 위한 함수를 import 하고 유사도 매트릭스를 구하고 출력해 본다.**

```
from sklearn.metrics.pairwise import linear_kernel  #코사인 유사도를 구하는 함수
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
cosine_sim
```

```
array([[1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 1.        , 0.        , ..., 0.02160533, 0.        ,
        0.        ],
       ...,
       [0.        , 0.02160533, 0.01488159, ..., 1.        , 0.01609091,
        0.00701914],
       [0.        , 0.        , 0.        , ..., 0.01609091, 1.        ,
        0.01171696],
       [0.        , 0.        , 0.        , ..., 0.00701914, 0.01171696,
        1.        ]])
```

```
cosine_sim.shape
```

```
(4803, 4803)
```

	문장1	문장2	문장3	...	문장4801	문장4802	문장4803
문장1	1.	0.	0.	...	0.	0.	0.
문장2	0.	1.	0.	...	0.02160533	0.	0
...	...	...	...	...	...	...	...
문장4803	0.	0.	0.	...	0.00701914	0.01171696	1

- **영화 제목을 입력하면 데이터의 index값을 출력하기 위한 작업을 한다. (series: 일차원배열)**

```
indices = pd.Series(df2.index, index=df2['title']).drop_duplicates()
indices
```

```
title
Avatar                                         0
Pirates of the Caribbean: At World's End       1
Spectre                                        2
The Dark Knight Rises                          3
John Carter                                    4
...
El Mariachi                                 4798
Newlyweds                                   4799
Signed, Sealed, Delivered                   4800
Shanghai Calling                            4801
My Date with Drew                           4802
Length: 4803, dtype: int64
```

- **'Avatar'의 index 값을 출력한다.**

```
indices['Avatar']
```

```
0
```

- **index 0의 데이터 값을 일차원, 이차원 배열로 출력한다.**

```
df2.iloc[0]  # df2.iloc[indices['Avatar']] 결과가 같다.
```

```
budget                                                    237000000
genres                  [{"id": 28, "name": "Action"}, {"id": 12, "nam...
homepage                                    http://www.avatarmovie.com/
id                                                            19995
keywords                [{"id": 1463, "name": "culture clash"}, {"id":...
original_language                                                  en
original_title                                                 Avatar
overview                In the 22nd century, a paraplegic Marine is di...
popularity                                                150.437577
production_companies    [{"name": "Ingenious Film Partners", "id": 289...
production_countries    [{"iso_3166_1": "US", "name": "United States o...
release_date                                              2009-12-10
revenue                                                   2787965087
runtime                                                      162.0
spoken_languages        [{"iso_639_1": "en", "name": "English"}, {"iso...
status                                                     Released
tagline                                    Enter the World of Pandora.
title                                                         Avatar
vote_average                                                     7.2
vote_count                                                     11800
cast                    [{"cast_id": 242, "character": "Jake Sully", "...
crew                    [{"credit_id": "52fe48009251416c750aca23", "de...
Name: 0, dtype: object
```

```
df2.iloc[[0]]
```

	budget	genres	homepage	id	vote_count	cast	crew
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	11800	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...

1 rows × 22 columns

- **영화의 제목을 입력 받으면 코사인 유사도를 통해서 가장 유사도가 높은 상위 10개의 영화 목록 반환함수를 작성한다.**

1) 영화 제목을 통해서 전체 데이터 기준 그 영화의 index 값을 얻어온다. (index: 해당 영화의 위치)

```
test_idx = indices['The Dark Knight Rises']
test_idx
```

```
3
```

2) cosine 유사도 matrix(cosin_sim)에서 idx에 해당하는 데이터를 (index, 유사도) 형태로 얻어온다.

```
cosine_sim[3]
```

```
array([0.02499512, 0.        , 0.        , ..., 0.03386366, 0.04275232,
       0.02269198])
```

```
test_sim_scores=list(enumerate(cosine_sim[3]))
test_sim_scores
```

```
[(0, 0.0249951158376727),
 (1, 0.0),
 (2, 0.0),
 (3, 1.0),
 (4, 0.010433403719159354),
 (5, 0.0051446018158107934),
 (6, 0.01260063243546246),
 (7, 0.026954270578912674),
 (8, 0.02065221688538951),
 (9, 0.1337400906655523),
 (10, 0.0),
 (11, 0.0),
 (12, 0.0),
 (13, 0.0),
 (14, 0.0),
 (15, 0.004071333922512107),
 (16, 0.021121093874993183),
 (17, 0.0),
 (18, 0.006768893195007471),
 (19, 0.010765175685064708),
 (20, 0.007178266390761152),
 (21, 0.033380775071488206),
...
 (996, 0.0),
 (997, 0.0),
 (998, 0.0),
 (999, 0.0),
 ...]
```

3) cosine 유사도가 높은 순으로 정렬한 후 자기 자신(index=0)을 제외한 10개의 추천 영화를 슬라이싱 한다.

```
test_sim_scores = sorted(test_sim_scores, key=lambda x: x[1], reverse=True)
# sort() 함수는 정렬한 결과를 원본데이터에 저장한다.
test_sim_scores = test_sim_scores[1 : 11]
test_sim_scores
```

```
[(65, 0.30151176591665485),
 (299, 0.29857045255396825),
 (428, 0.2878505467001694),
 (1359, 0.264460923827995),
 (3854, 0.18545003006561456),
 (119, 0.16799626199850706),
 (2507, 0.16682891043358278),
 (9, 0.1337400906655523),
 (1181, 0.13219702138476813),
 (210, 0.13045537014449818)]
```

- **lambda(익명) 함수 예제**

```
def get_second(x):
    return x[1]

lst = ['인덱스', '유사도']
print(get_second(lst))
```

```
유사도
```

```
(lambda x: x[1])(lst)
```

```
'유사도'
```

4) 유사도 행렬에서 10개에서 추천영화 인덱스 정보만 추출한다.

```
test_movie_indices = [i[0] for i in test_sim_scores[1: 11]]
test_movie_indices
```

```
[65, 299, 428, 1359, 3854, 119, 2507, 9, 1181, 210]
```

5) 인덱스 정보를 통해 영화 제목을 추출한다.

```
df2['title'].iloc[test_movie_indices]
```

```
65                                  The Dark Knight
299                                  Batman Forever
428                                  Batman Returns
1359                                         Batman
3854    Batman: The Dark Knight Returns, Part 2
119                                  Batman Begins
2507                                     Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                         JFK
210                                  Batman & Robin
Name: title, dtype: object
```

6) 최종적으로 함수를 작성한다.

```
def get_recommendations(title, cosine_sim=cosine_sim):
    # 영화 제목을 통해 영화의 index 값을 얻기
    idx = indices[title]     #df2[df2['title']==title].index[0]

    # cosine 유사도 매트릭스에서 idx에 해당하는 데이터를 [idx, 유사도] 형태로 얻기
    sim_socres = list(enumerate(cosine_sim[idx]))

    # cosine 유사도 기준으로 내림차순 정렬
    sim_socres = sorted(sim_socres, key=lambda x: x[1], reverse=True)

    # 자기 자신을 제외한 10개의 추천 영화를 슬라이싱
    sim_socres = sim_socres[1: 11]

    # 추천 영화 목록 10개의 index 정보 추출
    movie_indices = [i[0] for i in sim_socres]

    # index 정보를 통해 영화 제목 추출
    return df2['title'].iloc[movie_indices]
```

- **작성한 함수(get_recommendation)를 적용하여 추천 영화 목록을 추출해 본다.**

1) 영화 제목 10개를 출력 한다.

```
df2['title'][:20]
```

```
0                                              Avatar
1            Pirates of the Caribbean: At World's End
2                                             Spectre
3                              The Dark Knight Rises
4                                           John Carter
5                                          Spider-Man 3
6                                              Tangled
7                            Avengers: Age of Ultron
8             Harry Potter and the Half-Blood Prince
9               Batman v Superman: Dawn of Justice
10                                  Superman Returns
11                                Quantum of Solace
12        Pirates of the Caribbean: Dead Man's Chest
13                                 The Lone Ranger
14                                      Man of Steel
15          The Chronicles of Narnia: Prince Caspian
16                                     The Avengers
17        Pirates of the Caribbean: On Stranger Tides
18                                     Men in Black 3
19          The Hobbit: The Battle of the Five Armies
Name: title, dtype: object
```

2) 영화 제목 'Avengers: Age of Ultron'와 줄거리가 비슷한 10개의 영화를 출력한다.

```
get_recommendations('Avengers: Age of Ultron')
```

```
16                              The Avengers
79                                Iron Man 2
68                                  Iron Man
26                    Captain America: Civil War
227                             Knight and Day
31                                Iron Man 3
1868                        Cradle 2 the Grave
344                            Unstoppable
1922                             Gettysburg
531                  The Man from U.N.C.L.E.
Name: title, dtype: object
```

3) 영화 제목 'The Avengers'와 줄거리가 비슷한 10개의 영화를 출력한다.

```
get_recommendations('The Avengers')
```

```
7                       Avengers: Age of Ultron
3144                                  Plastic
1715                                  Timecop
4124                       This Thing of Ours
3311                    Thank You for Smoking
3033                            The Corruptor
588          Wall Street: Money Never Sleeps
2136             Team America: World Police
1468                           The Fountain
1286                            Snowpiercer
Name: title, dtype: object
```

- **다양한 요소 기반 추천 (Genre, keywords, cast, crew 기반 추천)**

▪ **데이터를 분석 및 처리하기 위하여 적합한 형태로 만드는 데이터 전처리 과정(1 ~5)을 아래와 같이 실행한다.**

1. genre, keywords, cast, crew 칼럼을 string 타입을 list 타입으로 변환한다.

```
df2.head(2)
```

	budget	genres	homepage	id	keywords	cast	crew
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{'id': 1463, 'name': 'culture clash'}, {'id':...	[{'cast_id': 242, 'character': 'Jake Sully', '...	[{'credit_id': '52fe48009251416c750aca23', 'de...
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...	[{'cast_id': 4, 'character': 'Captain Jack Spa...	[{'credit_id': '52fe4232c3a36847f800b579', 'de...

```
df2.loc[0, 'genres']
```

```
'[{"id":28,"name":"Action"},{"id":12,"name":"Adventure"},{"id":14,"name":"Fantasy"},{"id":878,"name":"Science Fiction"}]'
```

```
type(df2.loc[0, 'genres'])
```

```
str
```

```
from ast import literal_eval
df2['genres'] = df2['genres'].apply(literal_eval)
```

```
([{'id': 28, 'name': 'Action'}], list)
```

```
df2.loc[0, 'genres']
```

```
[{'id': 28, 'name': 'Action'},
 {'id': 12, 'name': 'Adventure'},
 {'id': 14, 'name': 'Fantasy'},
 {'id': 878, 'name': 'Science Fiction'}]
```

```
type(df2.loc[0, 'genres'])
```

```
list
```

- **features 목록의 모든 칼럼들의 타입을 str에서 list로 변경한다. (한번만 실행)**

```
features = ['cast', 'crew', 'keywords']
for feature in features:
    df2[feature] = df2[feature].apply(literal_eval)
```

2. 'crew' 칼럼에서 감독의 이름을 추출한다. (job이 Director를 찾아서 이름을 추출한다.)

```
df2.loc[0, 'crew']
```

```
[
  ...
{'credit_id': '52fe48009251416c750ac9c3',
  'department': 'Directing',
  'gender': 2,
  'id': 2710,
  'job': 'Director',
  'name': 'James Cameron'}
    ...
]
```

- 감독(Director)의 이름 정보(name)를 추출하는 함수를 작성한다.

```
def get_director(x):  # 매개변수 x에는 crew list 값을 받는다.
    for i in x:
        if i['job'] == 'Director':
            return i['name']
    return np.nan
```

- 'director'칼럼을 생성한 후 get_director함수를 적용하여 감독 이름을 저장한다.

```
df2['director'] = df2['crew'].apply(get_director)
df2['director']
```

```
0                 James Cameron
1                 Gore Verbinski
2                     Sam Mendes
3               Christopher Nolan
4                  Andrew Stanton
...
4798             Robert Rodriguez
4799                Edward Burns
4800                 Scott Smith
4801                Daniel Hsia
4802             Brian Herzlinger
Name: director, Length: 4803, dtype: object
```

- 'director' 칼럼에 null이 저장되어 있는 데이터를 출력한다.

```
df2[df2['director'].isnull()]
```

...	runtime	spoken_languages	status	tagline	title	vote_average	vote_count	cast	crew	director
...	95.0	[{"iso_639_1": "en", "name": "English"}]	Released	It's about the music	Flying By	7.0	2	[{'cast_id': 1, 'character': 'George', 'credit...	[]	NaN
...	88.0	[]	Released	NaN	Running Forever	0.0	0	[]	[]	NaN

3. 'cast', 'genres', 'keywords' list에서 3개의 항목만 추출한다.

- 'cast' 칼럼에서 상위 3개의 row만 출력한다.

```
df2.loc[0, 'cast']
```

```
[{'cast_id': 242,
  'character': 'Jake Sully',
  'credit_id': '5602a8a7c3a3685532001c9a',
  'gender': 2,
  'id': 65731,
  'name': 'Sam Worthington',
  'order': 0},
 {'cast_id': 3,
  'character': 'Neytiri',
  'credit_id': '52fe48009251416c750ac9cb',
  'gender': 1,
  'id': 8691,
  'name': 'Zoe Saldana',
  'order': 1},
 {'cast_id': 25,
  'character': 'Dr. Grace Augustine',
  'credit_id': '52fe48009251416c750aca39',
  'gender': 1,
  'id': 10205,
  'name': 'Sigourney Weaver',
  'order': 2},
  ... ]
```

- 'genres' 칼럼의 상위 3개 row만 출력한다.

```
df2.loc[0, 'genres']
```

```
[{'id': 28, 'name': 'Action'},
 {'id': 12, 'name': 'Adventure'},
 {'id': 14, 'name': 'Fantasy'},
 {'id': 878, 'name': 'Science Fiction'}]
```

- 'keywords' 칼럼에서 상위 3개 row만 출력한다.

```
df2.loc[0, 'keywords']
```

```
[{'id': 1463, 'name': 'culture clash'},
 {'id': 2964, 'name': 'future'},
 {'id': 3386, 'name': 'space war'},
 {'id': 3388, 'name': 'space colony'},
 ... ]
```

- list를 받아서 상위 3개만 추출하는 함수를 작성한다. list 타입이 아닌 경우 빈 배열을 리턴한다.

```
def get_list(x):
    if isinstance(x, list):
        names = [i['name'] for i in x]
        if len(names) > 3:
            names = names[:3]
        return names
    return []
```

- 'cast', 'keywords', 'genres' 칼럼에 get_list 함수를 적용하여 각 칼럼을 업데이트한다.

```
features = ['cast', 'keywords', 'genres']
for feature in features:
    df2[feature] = df2[feature].apply(get_list)
```

- df2 데이터세트의 상위 3개의 각 칼럼들을 출력한다.

```
df2[['title', 'director', 'cast', 'keywords', 'genres']].head(3)
```

	title	director	cast	keywords	genres
0	Avatar	James Cameron	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[culture clash, future, space war]	[Action, Adventure, Fantasy]
1	Pirates of the Caribbean: At World's End	Gore Verbinski	[Johnny Depp, Orlando Bloom, Keira Knightley]	[ocean, drug abuse, exotic island]	[Adventure, Fantasy, Action]
2	Spectre	Sam Mendes	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[spy, based on novel, secret agent]	[Action, Adventure, Crime]

4. 값을 입력 받아 빈칸을 없애고 소문자로 치환한다.

- list 또는 string을 받아서 빈칸을 없애고 소문자로 치환하는 함수를 작성한다.

```
def clean_data(x):
    if isinstance(x, list):
        return [str.lower(i.replace(' ', '')) for i in x]
    else:
        if isinstance(x, str):
            return str.lower(x.replace(' ', ''))
        else:
            return ''
```

- 'clean_data' 함수를 'cast', 'keywords', 'genres', 'director' 칼럼에 적용한다.

```
features = ['cast', 'keywords', 'genres', 'director']
for feature in features:
    df2[feature] = df2[feature].apply(clean_data)
```

- 'clean_data' 함수가 잘 적용되었는지 상위 3개의 데이터를 출력한다.

```
df2[['title', 'director', 'cast', 'keywords', 'genres']].head(3)
```

	title	director	cast	keywords	genres
0	Avatar	jamescameron	[samworthington, zoesaldana, sigourneyweaver]	[cultureclash, future, spacewar]	[action, adventure, fantasy]
1	Pirates of the Caribbean: At World's End	goreverbinski	[johnnydepp, orlandobloom, keiraknightley]	[ocean, drugabuse, exoticisland]	[adventure, fantasy, action]
2	Spectre	sammendes	[danielcraig, christophwaltz, léaseydoux]	[spy, basedonnovel, secretagent]	[action, adventure, crime]

5. 'keywords', 'cast', 'director', 'genres' 칼럼들과 각 칼럼 데이터 값들 다음에 빈칸을 연결하여 'soup' 칼럼 생성 후 저장한다.

```
def create_soup(x):
    return ' '.join(x['keywords'])+' '+' '.join(x['cast'])+' '+x['director']+' '+' '.join(x['genres'])
df2['soup'] = df2.apply(create_soup, axis=1)
df2['soup']
```

```
0       cultureclash future spacewar samworthington zo...
1       ocean drugabuse exoticisland johnnydepp orland...
2       spy basedonnovel secretagent danielcraig chris...
3       dccomics crimefighter terrorist christianbale ...
4       basedonnovel mars medallion taylorkitsch lynnc...
                              ...
4798    unitedstates-mexicobarrier legs arms carlosgal...
4799    edwardburns kerrybishé  marshadietlein edwardb...
4800    date loveatfirstsight narration ericmabius kri...
4801    danielhenney elizacoupe billpaxton danielhsia ...
4802    obsession camcorder crush drewbarrymore brianh...
Name: soup, Length: 4803, dtype: object
```

- 'overview' 칼럼과 'soup' 칼럼을 비교해 보면 유사한 형태로 저장되어있다.

```
df2['overview']
```

```
0       In the 22nd century, a paraplegic Marine is di...
1       Captain Barbossa, long believed to be dead, ha...
4       John Carter is a war-weary, former military ca...
                              ...
4798    El Mariachi just wants to play his guitar and ...
4799    A newlywed couple's honeymoon is upended by th...
4800    "Signed, Sealed, Delivered" introduces a dedic...
4801    When ambitious New York attorney Sam is sent t...
4802    Ever since the second grade when he first saw ...
Name: overview, Length: 4803, dtype: object
```

- 각 단어가 중요하므로 단어의 빈도를 Count하여 BOW 벡터(Vector)로 만드는 CountVectorizer 클래스를 사용한다.

```
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer(stop_words='english')
count_matrix = count.fit_transform(df2['soup'])
count_matrix
```

```
<4803x11520 sparse matrix of type '<class 'numpy.int64'>'
        with 42935 stored elements in Compressed Sparse Row format>
```

- CounteVectorizer로 만든 벡터 객체의 cosine 유사도 매트릭스를 구한다.

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_sim2 = cosine_similarity(count_matrix, count_matrix)
cosine_sim2
```

```
array([
        [1. , 0.3, 0.2, ..., 0. , 0. , 0. ],
        [0.3, 1. , 0.2, ..., 0. , 0. , 0. ],
        [0.2, 0.2, 1. , ..., 0. , 0. , 0. ],
        ...,
        [0. , 0. , 0. , ..., 1. , 0. , 0. ],
        [0. , 0. , 0. , ..., 0. , 1. , 0. ],
        [0. , 0. , 0. , ..., 0. , 0. , 1. ]
])
```

- **제목으로 index 값을 추출하는 'indices' 배열(indices)을 생성한다.**

```
df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title'])
indices
```

```
title
Avatar                                        0
Pirates of the Caribbean: At World's End      1
Spectre                                       2
The Dark Knight Rises                         3
John Carter                                   4
...
El Mariachi                                4798
Newlyweds                                  4799
Signed, Sealed, Delivered                  4800
Shanghai Calling                           4801
My Date with Drew                          4802
Length: 4803, dtype: int64
```

- **'The Dark Knight Rises' 제목의 index 값을 출력한다.**

```
indices['The Dark Knight Rises']
```

```
3
```

- **위에서 작성한 함수(get_recommendations)를 적용해 영화를 추천한다.**

1) 'The Dark Knight Rises'과 유사한 영화 10개를 추천한다.

```
get_recommendations('The Dark Knight Rises', cosine_sim2)
```

```
65                  The Dark Knight
119                   Batman Begins
4638          Amidst the Devil's Wings
1196                   The Prestige
3073                Romeo Is Bleeding
3326                Black November
1503                        Takers
1986                        Faster
303                       Catwoman
747                   Gangster Squad
Name: title, dtype: object
```

```
get_recommendations('The Dark Knight', cosine_sim2)
```

```
3                    The Dark Knight Rises
119                         Batman Begins
4638                Amidst the Devil's Wings
2398                                Hitman
1720                              Kick-Ass
1740                            Kick-Ass 2
3326                      Black November
1503                              Takers
1986                              Faster
303                             Catwoman
Name: title, dtype: object
```

2) 'The Martian'과 유사한 영화 10개를 추천한다.

```
get_recommendations('The Martian', cosine_sim2)
```

```
4                              John Carter
95                             Interstellar
365                                Contact
256                              Allegiant
1326                          The 5th Wave
1958                          On the Road
3043                        End of the Spear
3373          The Other Side of Heaven
3392                                 Gerry
3698                           Moby Dick
Name: title, dtype: object
```

```
indices['The Martian']
```

```
270
```

3) 'The Martian'과 'John Carter' 두 영화정보를 비교해 본다.

```
df2.loc[270]
```

```
budget                                                    108000000
genres                    [drama, adventure, sciencefiction]
homepage                                  http://www.foxmovies.com/movies/the-martian
id                                                        286217
keywords                  [basedonnovel, mars, nasa]
original_language                                         en
status                                                    Released
tagline                                                   Bring Him Home
title                                                     The Martian
vote_average                                              7.6
vote_count                                                7268
cast                      [mattdamon, jessicachastain, kristenwiig]
crew                      [{'credit_id': '5607a7e19251413050003e2c', 'de...
director                                                  ridleyscott
soup                      basedonnovel mars nasa mattdamon jessicachasta...
Name: 270, dtype: object
```

```
df2.loc[4]
```

```
budget                                                    260000000
genres                    [action, adventure, sciencefiction]
homepage                                  http://movies.disney.com/john-carter
id                                                        49529
keywords                  [basedonnovel, mars, medallion]
original_language                                         en
original_title                                            John Carter
overview                  John Carter is a war-weary, former military ca...
vote_average                                              6.1
vote_count                                                2124
cast                      [taylorkitsch, lynncollins, samanthamorton]
crew                      [{'credit_id': '52fe479ac3a36847f813eaa3', 'de...
director                                                  andrewstanton
soup                      basedonnovel mars medallion taylorkitsch lynnc...
Name: 4, dtype: object
```

- **영화 추천 웹사이트**

  - **df2에서 'id', 'title' 칼럼들을 복사하여 movies 변수에 저장한다.**

```
movies = df2[['id', 'title']].copy()
```

  - **영화 추천에서 사용할 영화정보('movies') 데이터와 유사도('cosine_sim2') 데이터를 pickle 파일로 저장한다.**

```
import pickle
```

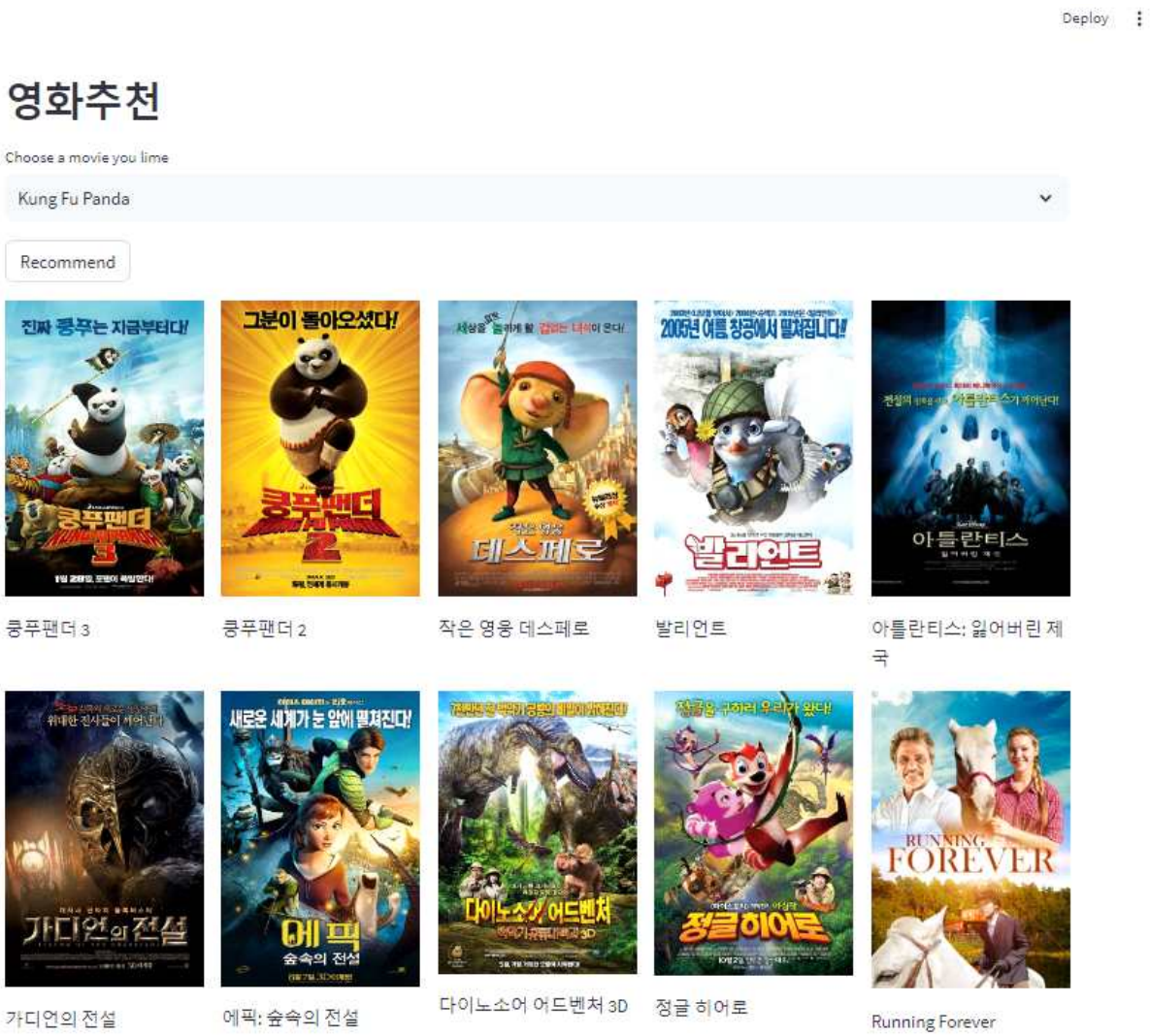
```
pickle.dump(movies, open('movies.pickle', 'wb'))
```

```
pickle.dump(cosine_sim2, open('cosine_sim.pickle', 'wb'))
```

  - **터미널에서 필요한 라이브러리 streamlit(Padas, Matplotlib, sklearn의 결과를 웹에서 확인), TMDB를 설치한다.**

```
pip install streamlit  #브라우저에 출력할 UI를 쉽게 제작할 수 있는 라이브러리
```

```
pip install  tmdbv3api  #TMDB에서 영화 상세정보를 제공해주는 라이브러리
```



- [구글]-[TMDB] 사이트에서 회원가입 후 오른쪽 상단 [설정]에서 [API] 키를 생성한다.
- 웹 프로그램을 작성한다. (실행: [cmd창] streamlit run app.py)

app.py

```
import pickle
import streamlit as st
from tmdbv3api import Movie, TMDb

def get_recommendations(title):
    # 영화 제목을 통해 영화의 index 값을 얻기
    idx = movies[movies['title']==title].index[0]

    # cosine 유사도 매트릭스에서 idx에 해당하는 데이터를 (idx, 유사도) 형태로 얻기
    sim_scores = list(enumerate(cosine_sim[idx]))

    # cosine 유사도 기준으로 내림차순 정렬
    sim_scores = sorted(sim_scores, key=lambda x:x[1], reverse=True)

    # 자기 자신을 제외한 10개의 추천 영화를 슬라이싱
    sim_scores = sim_scores[1:11]

    # 추천 영화 목록 10개의 index 정보 추출
    movie_indices = [i[0] for i in sim_scores]

    # index 정보를 통해 영화 제목 추출
    images = []
    titles = []
    for i in movie_indices:
        id = movies['id'].iloc[i]
        details = movie.details(id)
        image_path = details['poster_path']
        if image_path:
            image_path = 'https://image.tmdb.org/t/p/w500' + details['poster_path']
        else:
            image_path = 'no_image.jpg'
        images.append(image_path)
        titles.append(details['title'])  #details title은 TMDB 한글 설정 시 한글제목을 가져올 수 있다.
    return images, titles

movie = Movie()
tmdb = TMDb()
tmdb.api_key = 'your api key'
tmdb.language ='ko-KR'

movies = pickle.load(open('movies.pickle', 'rb'))
cosine_sim = pickle.load(open('cosine_sim.pickle', 'rb'))

st.set_page_config(layout='wide')
st.header('영화추천')

movie_list = movies['title'].values

title = st.selectbox('Choose a movie you lime', movie_list)

if st.button('Recommend'):
    with st.spinner('Please wait...'):
        images, titles = get_recommendations(title)
        idx = 0
        for i in range(0, 2):
            cols = st.columns(5)
            for col in cols:
                col.image(images[idx])
                col.write(titles[idx])
                idx += 1
```

• Collaborative Filtering(협업 필터링: 사용자 리뷰 기반)

어떤 item(영화)에 대해서 비슷한 취향을 가진 사람들이 다른 item(영화)에 대해서도 비슷한 취향을 가지고 있을 것이라고 가정하고 추천을 하는 알고리즘이다. 추천의 대상이 되는 사람과 취향이 비슷한 사람들을 찾아 이 사람들이 공통적으로 좋아하는 제품 또는 서비스를 추천 대상인에 추천하는 것이다.

• The Movies Dataset 다운로드

1) [구글]-[TMDB 5000]-[Kaggle 사이트이동]-[상단 Code Tab]-[우측 Hotness 메뉴]-[Most Votes] 선택한다.
2) [Getting Started with a Movie Recommendation System]-[The Movie Dataset]
3) 'ratings_small.csv' 파일을 다운로드하여 프로젝트 폴더로 이동한다.

- **협업 필터링에 필요한 라이브러리 'pip install scikit-surprise'를 설치한다.**

```
pip install scikit-surprise
```

- **pip install로 오류가 발생할 경우 conda 명령어로 설치한다.**

```
conda install -c conda-forge scikit-surprise
```

- **'07.Collaborative Filtering.ipynb' 파일을 생성한 후 판다스 라이브러리를 import 한다.**

```
import pandas as pd
```

- **csv 파일을 읽어 데이터프레임(ratins)에 저장한 후 저장된 데이터프레임을 5개 출력한다.**

```
ratings = pd.read_csv('ratings_small.csv')
ratings.head(3)
```

	userId	movieId	rating	timestamp
0	1	31	2.5	1260759144
1	1	1029	3.0	1260759179
2	1	1061	3.0	1260759182
3	1	1129	2.0	1260759185
4	1	1172	4.0	1260759205

- **판다스 DataFrame의 최솟값, 최댓값을 출력한다.**

```
ratings['rating'].min(), ratings['rating'].max()
```

```
(0.5, 5.0)
```

- **Surprise 패키지에서 제공하는 Reader 클래스를 이용하여 평점단위는 0.5~5 설정된 Reader 객체를 생성한다.**

```
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate
reader = Reader(rating_scale=(0.5, 5.0))
```

- **Dataset.load_from-file()은 Reader 객체를 기반으로 데이터를 파싱하여 Surprise Dataset로 로딩 한다.**

```
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader=reader)
data  #Surprise 는 사용자 아이디, item 아이디, 평점 데이터가 로우 레벨로 된 데이터 세트만 적용할 수 있다.
```

```
<surprise.dataset.DatasetAutoFolds at 0x1654899f8d0>
```

- **SVD 알고리즘으로 평점 예측 값을 구하기 위하여 SVD 모델을 생성한다.**

```
svd = SVD(random_state=0)
```

- **알고리즘 평가를 위해 cross_validate() 교차검증 클래스를 이용한다. 이 클래스는 알고리즘 객체, 데이터, 성능 평가 방법, 데이터 세트 개수(cv)를 인자로 받아 fold별 성능평가수치와 전체 fold의 평균 성능평가수치를 함께 보여준다.**

```
cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8999  0.8937  0.8987  0.8958  0.9002  0.8977  0.0025
MAE (testset)     0.6926  0.6898  0.6910  0.6894  0.6944  0.6914  0.0019
Fit time          1.37    1.45    1.42    1.41    1.47    1.42    0.04
Test time         0.19    0.17    0.27    0.18    0.27    0.22    0.05
{'test_rmse': array([0.8999247 , 0.89371236, 0.89873246, 0.89581395, 0.90015218]),
'test_mae': array([0.69263929, 0.68980035, 0.69101543, 0.68937457, 0.69440648]),
'fit_time': (1.3661470413208008,
1.4471731185913086,
1.422184944152832,
1.4082145690917969,
1.4729771614074707),
'test_time': (0.1948843002319336,
0.1679229736328125,
0.2748243808746338,
0.17587780952453613,
0.2651553153991699)}
```

- **교차 검증 (K-Fold 교차 검증)**

100개의 데이터인 경우 4개의 세트로 훈련하고 1개의 세트로 테스트하는 작업을 5번하고 결과의 평균을 구하여 성능을 검증한다.

No	Group	Train Set	Test Set
1	1~20	ABCD	E
2	21~40	ABCE	D
3	41~60	ABDE	C
4	61~80	ACDE	B
5	81~100	BCDE	A

- **build_full_trainset() 메서드를 이용해 학습데이터로 생성하고 SVD를 이용해 학습을 한다.**

```
trainset = data.build_full_trainset()
svd.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x165467859d0>
```

- **userId=9인 특정 사용자가 아직 평점을 매기지 않은 영화를 movieId=42로 선정한 뒤 예측 평점을 계산해 보시오.**

1) 아이디가 9인 사용자가 평점을 매긴 모든 영화들을 movies에 저장 후 영화수와 영화 아이디들을 출력한다.

```
uid=9
mid=42
movies = ratings[ratings['userId']==uid]['movieId']
movies.count()
```

```
(45, array([   1,   17,   26,   36,   47,  318,  497,  515,  527,  534,  593,
         608,  733, 1059, 1177, 1357, 1358, 1411, 1541, 1584, 1680, 1682,
        1704, 1721, 1784, 2028, 2125, 2140, 2249, 2268, 2273, 2278, 2291,
        2294, 2302, 2391, 2396, 2427, 2490, 2501, 2539, 2571, 2628, 2762,
        2857], dtype=int64))
```

2) userId=9인 사용자가 평점을 매긴 영화중 movieId=42 데이터가 있는지 확인한다.

```
if movies[movies==mid].count() == 0:
    print(f'사용자 아이디 {uid}는(은) 영화 아이디 {mid}의 평점 없음')
```

```
사용자 아이디 9는(은) 영화 아이디 42의 평점 없음
```

3) 예측 평점을 출력해 보시오

```
pred = svd.predict(uid, mid)
pred
```

```
Prediction(uid=9, iid=42, r_ui=None, est=2.8882239765624007, details={'was_impossible': False})
```

```
print(f'사용자 아이디 {uid}의 예측 평점은 {pred.est}점 입니다.')
```

```
print(f'사용자 아이디 {uid}의 예측 평점은 {pred.est}점 입니다.')
```

- **특정 사용자에 대한 '평점 매긴 영화수', '전체 영화수', '추천 대상 영화수'를 출력한다.**

1) 특정 사용자가 평점 매긴 영화목록을 추출한다.

```
seen_movies = ratings[ratings['userId']==uid]['movieId'].tolist()
len(seen_movies)
```

```
45
```

2) 전체 사용자가 매긴 평점 영화목록(ratings 데이터세트)에서 중복 영화를 제거하고 전체 영화목록을 추출한다.

```
total_movies = ratings['movieId'].drop_duplicates().sort_values().tolist()
len(total_movies)
```

```
9066
```

3) 추천 대상 영화 목록(사용자가 평점을 매기지 않은 영화 목록)을 추출한다.

```
import numpy as np
unseen_movies = np.setdiff1d(total_movies, seen_movies)
len(unseen_movies)
```

```
9021
```

4) 위 작업(1~3)을 함수로 작성한다.

```
def get_unseen_movies(ratings, uid):
    seen_movies = ratings[ratings['userId']==uid]['movieId'].tolist()
    total_movies = ratings['movieId'].drop_duplicates().sort_values().tolist()
    unseen_movies = np.setdiff1d(total_movies, seen_movies)
    print(f'평점 매긴 영화수:{len(seen_movies)}')
    print(f'추천 대상 영화수:{len(unseen_movies)}')
    print(f'전체 영화수:{len(total_movies)}')
    return unseen_movies
```

5) 위 함수(geet_unseen)를 실행 후 결과를 추천 대상 목록(unseen_movies) 변수에 저장한다.

```
unseen_movies = get_unseen_movies(ratings, uid)
```

```
평점 매긴 영화수:45
추천 대상 영화수:9021
전체 영화수:9066
```

▪ 추천 대상 영화 목록을 학습된 SVD 알고리즘 클래스를 이용해 예측 평점을 구한 후 높은 순으로 Top-10을 출력한다.

1) 추천 대상 영화 목록의 데이터들을 SVD 모델을 이용하여 평점을 예측한다.

```
predictions = [svd.predict(uid, mid) for mid in unseen_movies]
predictions
```

```
[
    Prediction(uid=9, iid=2, r_ui=None, est=3.4776703267798115, details={'was_impossible': False}),
    Prediction(uid=9, iid=3, r_ui=None, est=2.890370484839762, details={'was_impossible': False}),
    ...
]
```

2) 예측 평점이 높은 순으로 정렬하여 예측 목록을 업데이트한다.

```
predictions.sort(key=lambda pred:pred.est, reverse=True)
predictions
```

```
[
    Prediction(uid=9, iid=969, r_ui=None, est=4.5715177935548486, details={'was_impossible': False}),
    Prediction(uid=9, iid=1252, r_ui=None, est=4.562552053514925, details={'was_impossible': False}),
    ...
]
```

3) 내림차순 정렬한 목록 중 상위 10개의 데이터만 추출한다.

```
top_predictions = predictions[:10]
top_predictions
```

```
[
   Prediction(uid=9, iid=969, r_ui=None, est=4.5715177935548486, details={'was_impossible': False}),
   Prediction(uid=9, iid=1252, r_ui=None, est=4.562552053514925, details={'was_impossible': False}),
   Prediction(uid=9, iid=858, r_ui=None, est=4.5531963576285355, details={'was_impossible': False}),
   Prediction(uid=9, iid=953, r_ui=None, est=4.5369347866835295, details={'was_impossible': False}),
   Prediction(uid=9, iid=2064, r_ui=None, est=4.5312957643838905, details={'was_impossible': False}),
   Prediction(uid=9, iid=56782, r_ui=None, est=4.522662680493645, details={'was_impossible': False}),
   Prediction(uid=9, iid=246, r_ui=None, est=4.493473714707143, details={'was_impossible': False}),
   Prediction(uid=9, iid=968, r_ui=None, est=4.4794794549639905, details={'was_impossible': False}),
   Prediction(uid=9, iid=912, r_ui=None, est=4.476629452828997, details={'was_impossible': False}),
   Prediction(uid=9, iid=1276, r_ui=None, est=4.456802536353343, details={'was_impossible': False})
]
```

4) 앞의 평점 예측 목록(top_predictions)에서 영화 아이디와 예측 평점만 추출하여 저장한다.

```
top_movies = [(pred.iid, pred.est) for pred in top_predictions]
top_movies
```

```
[
    (969, 4.5715177935548486),
    (1252, 4.562552053514925),
    (858, 4.5531963576285355),
    (953, 4.5369347866835295),
    (2064, 4.5312957643838905),
    (56782, 4.522662680493645),
    (246, 4.493473714707143),
    (968, 4.4794794549639905),
    (912, 4.476629452828997),
    (1276, 4.456802536353343)
]
```

5) 위 작업(1~4)을 함수로 정의한다.

```
def recomm_movies(predictions):
    predictions = [svd.predict(uid, mid) for mid in unseen_movies]
    predictions.sort(key=lambda pred:pred.est, reverse=True)
    top_predictions = predictions[:10]
    top_movies = [(pred.iid, pred.est) for pred in top_predictions]
    return top_movies
```

6) 위 함수(recomm_movies)를 실행 후 결과를 추천 대상 목록(movies) 변수에 저장하고 movies를 출력한다.

```
movies = recomm_movies(predictions)
movies
print('******* Top-10 추천 영화 리스트 *******')
print('-' * 50)
for movie in movies:
    print(f'영화아이디:{movie[0]} (평점:{movie[1]})')
    print('-' * 50)
```

```
******* Top-10 추천 영화 리스트 *******
--------------------------------------------------
영화아이디:969 (평점:4.5715177935548486)
--------------------------------------------------
영화아이디:1252 (평점:4.562552053514925)
--------------------------------------------------
영화아이디:858 (평점:4.5531963576285355)
--------------------------------------------------
영화아이디:953 (평점:4.5369347866835295)
--------------------------------------------------
영화아이디:2064 (평점:4.5312957643838905)
--------------------------------------------------
영화아이디:56782 (평점:4.522662680493645)
--------------------------------------------------
영화아이디:246 (평점:4.493473714707143)
--------------------------------------------------
영화아이디:968 (평점:4.4794794549639905)
--------------------------------------------------
영화아이디:912 (평점:4.476629452828997)
--------------------------------------------------
영화아이디:1276 (평점:4.456802536353343)
--------------------------------------------------
```

• 기타

• 인공지능 이미지 분류

이미지 분류는 이미지를 정해진 카테고리에 따라 AI가 분류해 주는 기술입니다. 이미지는 0~255 정수 범위의 값을 가지는 Width(너비) x Height(높이) x 3의 크기의 3차원 배열이다. 3은 Red, Green, Blue로 구성된 3개의 채널을 의미한다.

- **'08.ImageClassification.ipynb' 파일을 생성하고 필요한 라이브러리를 import한다.**

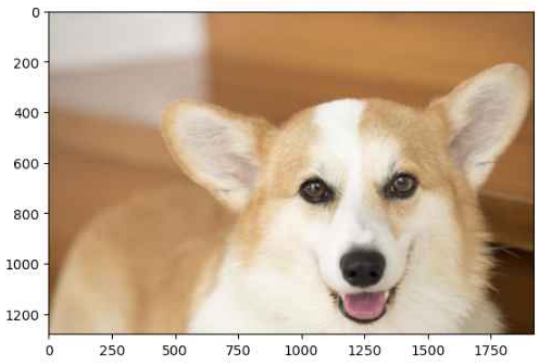
```
import cv2  #이미지관련 라이브러리
import numpy as np
from PIL import Image  PIL(python Image Library) 이미지를 불러오는 라이브러리
import matplotlib.pyplot as plt
```

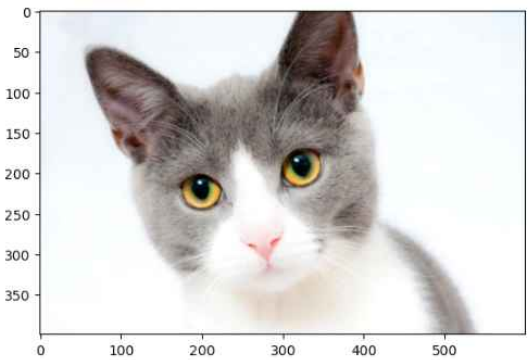
- **프로젝트 폴더에 이미지 파일들을 복사하고 Open한 파일들을 매트릭스로 변경한다.**

```
files  = ['./dog.jpg', './cat.jpg']
images = [np.array(Image.open(file)) for file in files]
```

- **배열(images)에 저장된 이미지를 출력한다.**

```
for image in images:
    plt.imshow(image)
    plt.show()
```





- **인공지능이 예측한 결과를 이미지와 매핑해주는 함수(decode_predictions)를 import 한다.**

```
from tensorflow.keras.applications.imagenet_utils import decode_predictions
```

- **esnet의 ResNet50이라는 인공지능의 imagenet으로 학습된 모델을 생성하여 저장하고 결과를 출력한다.**

```
resnet50_pre=tf.keras.applications.resnet.ResNet50(weights='imagenet', input_shape=(224, 224,3))
resnet50_pre.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            [(None, 224, 224, 3)]  0         []
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           ['input_4[0][0]']
conv1_conv (Conv2D)             (None, 112, 112, 64)  9472       ['conv1_pad[0][0]']
                                                    ...
avg_pool(GlobalAveragePoo       (None, 2048)         0           ['conv5_block3_out[0][0]']
ling2D)
predictions (Dense)             (None, 1000)         2049000     ['avg_pool[0][0]']
==================================================================================================
Total params: 25636712 (97.80 MB)
Trainable params: 25583592 (97.59 MB)
Non-trainable params: 53120 (207.50 KB)
```

- **인공지능이 예측한 결과를 이미지와 매핑해주는 함수(decode_predictions)를 import 하고 예측 함수를 작성한다.**

```
from tensorflow.keras.applications.imagenet_utils import decode_predictions
```

```
def pred_img(img):
    plt.imshow(img)
    plt.axis('off')
    plt.show()

    img_resized = cv2.resize(img, (224, 224))
    pred = resnet50_pre.predict(img_resized.reshape([1, 224, 224, 3]))
    decoded_pred = decode_predictions(pred)

    for i, instance in enumerate(decoded_pred[0]):
        print('{}위: {} ({:.2f}%) '.format(i+1, instance[1], instance[2] * 100))
```

```
pred_img(images[1])
```

```
1/1 [==============================] - 0s 238ms/step
1위: Siamese_cat (42.53%)
2위: Egyptian_cat (28.20%)
3위: tabby (12.51%)
4위: remote_control (7.33%)
5위: tiger_cat (2.23%)
```

- '09.ImageClassification.ipynb' 파일을 생성한다.

```
#셀에서 입력한 내용을 'app.py' 파일로 생성한다.
%%writefile app2.py

#웹 어플리케이션을 쉽게 만드는 라이브러리
import streamlit as st

#인공지능 모델생성 및 호출 라이브러리
import tensorflow as tf

#인공지능이 예측한 결과를 이미지와 매핑해주는 함수
from tensorflow.keras.applications.imagenet_utils import decode_predictions

#PIL(python Image Library) 이미지를 불러오거나 이미지 옵션을 지정하는 라이브러리
from PIL import Image, ImageOps

import numpy as np

#resnet의 ResNet50이라는 인공지능의 imagenet으로 학습된 모델을 호출
resnet50_pre=tf.keras.applications.resnet.ResNet50(weights='imagenet', input_shape=(224, 224,3))

st.title('이미지 분류 인공지능 웹페이지')

file = st.file_uploader('이미지를 올려주세요.', type=['jpg', 'png'])
if file is None:
    st.text('이미지를 먼저 올려주세요')
else:
    image = Image.open(file)  #업로드 한 file를 이미지로 읽어온다.
    st.image(image, use_column_width=True)  #웹 브라우저에 읽어온 이미지를 출력한다.

    #이미지를 인공지능 모델이 인식하도록 변환
    img_resized = ImageOps.fit(image, (224, 224), Image.ANTIALIAS)  #이미지 사이즈를 224x224로 변환
    img_resized = img_resized.convert('RGB')  #이미지를 읽어오면 4개의 채널이므로 'RGB'3개의 채널로 변환
    img_resized = np.asarray(img_resized)  #이미지를 Array형태로 변환

    pred = resnet50_pre.predict(img_resized.reshape([1, 224, 224, 3]))  #이미지 하나를 예측하여 저장
    decoded_pred = decode_predictions(pred)  #이미지의 매핑 결과를 예측하여 저장
    results = ''
    for i, instance in enumerate(decoded_pred[0]):
        results += '{}위: {} ({:.2f}%) '.format(i+1, instance[1], instance[2] * 100)

    st.success(results)
```

```
Overwriting app2.py
```

- 생성된 app2.py 파일을 브라우저에 실행한다.

```
!streamlit run app2.py
```

- 이미지 분류한 결과는 다음과 같다.

1위: Pembroke (53.88%) 2위: Cardigan (22.70%) 3위: Siberian_husky (4.99%) 4위: white_wolf (2.49%)
5위: Pomeranian (1.91%)

• 인공지능 스피커

- 가상머신을 생성한다.

```
python -m venv myenv
```

- 가상머신을 실행한다.

```
.\myenv\Scripts\activate
```

- google에서 제공하는 TTS(Text To Speech) 패키지를 설치한다.

```
(myenv) C:\data\python>pip install gttS
```

- 음성을 출력하는 패키지를 설치한다.

```
(myenv) C:\data\python>pip install playsound==1.2.2
```

- 영어문장을 Play한다.

[AISpeaker] text_to_speech.py

```
from gtts import gTTS
from playsound import playsound
file_name = 'sample.mp3'

text = 'Can I help you?'
tts_en = gTTS(text=text, lang='en')
tts_en.save(file_name)
playsound(file_name)
```

- 한글문장을 Play한다.

[AISpeaker] text_to_speech.py

```
...
text = '파이썬을 배우면 이런 것도 할 수 있어요.'
tts_ko = gTTS(text=text, lang='ko')
tts_ko.save(file_name)
playsound(file_name)
```

- 파일에서 읽어온 문장을 Play한다.

[AISpeaker] text_to_speech.py

```
...
with open('sample.txt', 'r', encoding='utf-8') as f:
    text = f.read()

tts_ko =gTTS(text=text, lang='ko')
tts_ko.save(file_name)
playsound(file_name)
```

- 음성을 텍스트로 변경하는 STT(Speech To Text) 패키지를 설치한다.

```
(myenv) C:\data\python> pip install SpeechRecognition
```

```
(myenv) C:\data\python>pip install PyAudio
```

[AISpeaker] speech_to_text.py

```
import speech_recognition as sr
r = sr.Recognizer()

#마이크로부터 음성 듣기
with sr.Microphone() as source:
    print('듣고 있어요.')
    audio = r.listen(source)  #마이크로부터 음성 듣기

try:
    #구글 API로 인식(하루 50회)
    text = r.recognize_google(audio, language='en-US')
    print(text)
except sr.UnknownValueError:
    print('인식 실패')
except sr.RequestError as e:
    print('요청 실패 : {0}'.format(e))  #API Key 오류, 네트워크 연결실패
```

[AISpeaker] speech_to_text.py

```
import speech_recognition as sr
r = sr.Recognizer()

#마이크로부터 음성 듣기
with sr.Microphone() as source:
    print('듣고 있어요.')
    audio = r.listen(source) #마이크로부터 음성 듣기

try:
    #구글 API로 인식(하루 50회)
    text = r.recognize_google(audio, language='ko')
    print(text)
except sr.UnknownValueError:
    print('인식 실패')
except sr.RequestError as e:
    print('요청 실패 : {0}'.format(e))  #API Key 오류, 네트워크 연결실패
```

[AISpeaker] speech_to_text.py

```
import speech_recognition as sr
r = sr.Recognizer()

#파일로부터 음성 불러오기 (가능파일:wav, aiff/aiff-c, flac 불가능파일: mp3)
r = sr.Recognizer()
with sr.AudioFile('Sample.wav') as source:
     audio = r.record(source)
...
```

- 묻고 답하는 인공지능 스피커를 아래와 같이 만들어 본다.

[AISpeaker] speaker.py

```
import time, os
import speech_recognition as sr
from gtts import gTTS
from playsound import playsound

#2.음성듣기(STT)
def listen(recognizer, audio):
    try:
        text = recognizer.recognize_google(audio, language='ko')
        print('[홍길동]' + text)
        answer(text)
    except sr.UnknownValueError:
        print('인식 실패')
    except sr.RequestError as e:
        print('요청 실패 : {0}'.format(e))  #API Key 오류, 네트워크 연결실패

#3.대답하기
def answer(input_text):
    answer_text = ''
    if '안녕' in input_text:
        answer_text = '안녕하세요? 반갑습니다.'
    elif '날씨' in input_text:
        answer_text = '오늘의 서울 기온은 20도 입니다. 맑은 하늘이 예상됩니다.'
    elif '환율' in input_text:
        answer_text = '원 달러 환율은 1380원입니다.'
    elif '고마워' in input_text:
        answer_text = '별 말씀을요.'
    elif '종료' in input_text:
        answer_text = '다음에 또 만나요'
        stop_listening(wait_for_stop=False)
    else:
        answer_text = '다시 한 번 말씀해 주시겠어요?'
    speak(answer_text)

#1.소리내기(TTS)
def speak(text):
    print('[컴퓨터]' + text)
    file_name = 'voice.mp3'
    tts = gTTS(text=text, lang='ko')
    tts.save(file_name)
    playsound(file_name)
    if os.path.exists(file_name):  #voice.mp3 파일이 있으면 삭제
        os.remove(file_name)

r = sr.Recognizer()
m = sr.Microphone()

speak('무엇을 도와드릴까요?')
stop_listening =r.listen_in_background(m, listen)

while True:
    time.sleep(0.1)
```

- **Flask로 웹페이지 만들기**

- **홈페이지를 작성한다. (실행: Flask run 또는 python app.py)**

[Flask] app.py

```
from flask import Flask, render_template
from aiRoute import ai

app = Flask(__name__)
app.register_blueprint(ai, url_prefix='/ai')

@app.route('/')
def index():
    return render_template('index.html', title='홈페이지')

if __name__ == '__main__':
    app.run(port=5001, debug=True)
```

[Flask]-[templates] base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="viewport" content="width=<device-width>, initial-scale=1.0">
    <link rel="stylesheet" href="/static/css/style.css"/>
    <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
    <title>{{title}}</title>
</head>
<body>
    <div class="container">
        {%include 'header.html'%}
        {% block main_area %}
        {% endblock %}
        {%include 'footer.html'%}
    </div>
</body>
</html>
```

[Flask]-[templates] header.html

```
<div class="row mt-5">
    <div class="col">
        <h1 class="text-center mb-5">인공지능(머신러닝)</h1>
        {%include 'menuAI.html'%}
        <hr>
    </div>
</div>
```

[Flask]-[templates] footer.html

```
<div class="row my-5">
    <div class="col">
        <hr>
        <h4 class="text-center">Copyright 2023 홍길동 All rights reserved.</h4>
    </div>
</div>
```

[Flask]-[templates] memeAI.html

```
<div>
    <a href="/ai/page1" class="me-3">선형회귀</a>
    <a href="/ai/page2" class="me-3">다항회귀</a>
    <a href="/ai/page3" class="me-3">다중선형회귀</a>
    <a href="/ai/page4" class="me-3">로지스틱회귀</a>
    <a href="/ai/page5" class="me-3">K-평균</a>
</div>
```

- 선형회귀 페이지를 작성한다.

[Flask] aiRoute.py

```
from flask import Blueprint, render_template, request, send_file
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
import numpy as np

from io import BytesIO
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic'
matplotlib.rcParams['font.size'] = 15
matplotlib.rcParams['axes.unicode_minus'] = False
plt.switch_backend('agg')
ai = Blueprint('ai', __name__)

@ai.route('/page1')
def ai_page1():
    return render_template('./ai/page1.html')

@ai.route('/linear')
def ai_linear():
    hours = int(request.args['hours'])
    dataset = pd.read_csv('./data/LinearRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
    reg = LinearRegression()
    reg.fit(X, y)
    y_predict = reg.predict([[hours]])
    return str(round(y_predict[0], 2))
```

[Flask] aiRoute.py

```
@ai.route('/linear/graph')
def ai_linear_graph():
    dataset = pd.read_csv('./data/LinearRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
    reg = LinearRegression()
    reg.fit(X, y)

    plt.figure(figsize=(10, 5))
    plt.scatter(X, y, color='blue')
    plt.plot(X, reg.predict(X), color='green')
    plt.title('Score by hours')
    plt.xlabel('hours')
    plt.ylabel('score')
    img = BytesIO()
    plt.savefig(img, format='png', dpi=200)
    img.seek(0)
    return send_file(img, mimetype='image/png')
```

[Flask]-[templates]-[ai] page1.html

```
{%extends 'base.html'%}
{%block main_area%}
    <div>
        <h3 class="text-center mb-5">선형회귀</h3>
        <div class="row justify-content-center">
            <form name="frm" class="col-6">
                <div class="input-group">
                    <input name="hours" class="form-control" placeholder="공부시간">
                    <button class="btn btn-primary">점수예측</button>
                </div>
            </form>
            <div class="mt-3">
                <h5 id="predict" class="text-center"></h5>
            </div>
            <div class="mb-3 text-center">
                <img src="/ai/linear/graph" width="50%">
            </div>
        </div>
    </div>
    <script>
        $(frm).on("submit", function(e){
            e.preventDefault();
            const hours=$(frm.hours).val();
            $.ajax({
                type:"get",
                url:"/ai/linear",
                data: {hours: hours},
                success:function(data){
                    console.log(data);
                    $("#predict").html(`${hours}시간 공부했을 때 예측점수는 ${data}점 입니다.`)
                }
            })
        });
    </script>
{%endblock%}
```

- **다항회귀 페이지를 작성한다.**

[Flask] aiRoute.py

```
...
@ai.route('/page1')
def ai_page1():
    return render_template('./ai/page1.html')

@ai.route('/linear')
def ai_linear():
    ...

@ai.route('/linear/graph')
def ai_linear_graph():
    ...

@ai.route('/page2')
def ai_page2():
    return render_template('./ai/page2.html')

@ai.route('/poly')
def ai_poly():
    hours = float(request.args['hours'])
    dataset = pd.read_csv('./data/PolynomialRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
    poly_reg = PolynomialFeatures(degree=2)
    X_poly = poly_reg.fit_transform(X)
    reg = LinearRegression()
    reg.fit(X_poly, y)
    y_predict= reg.predict(poly_reg.fit_transform([[hours]]))
    return str(round(y_predict[0], 2))

@ai.route('/poly/graph')
def ai_poly_graph():
    dataset = pd.read_csv('./data/PolynomialRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values
    poly_reg = PolynomialFeatures(degree=2)
    X_poly = poly_reg.fit_transform(X)
    reg = LinearRegression()
    reg.fit(X_poly, y)

    plt.figure(figsize=(10, 5))
    plt.scatter(X, y, color='blue')
    X_range = np.arange(min(X), max(X), 0.1)
    X_range=X_range.reshape(-1, 1)
    plt.plot(X_range, reg.predict(poly_reg.fit_transform(X_range)), color='green')
    plt.title('Score by hours (genius)')
    plt.xlabel('hours')
    plt.ylabel('score')
    img = BytesIO()
    plt.savefig(img, format='png', dpi=200)
    img.seek(0)
    return send_file(img, mimetype='image/png')
```

[Flask]-[templates]-[ai] page2.html

```
{%extends 'base.html'%}
{%block main_area%}
    <div>
        <h3 class="text-center mb-5">다항회귀</h3>
        <div class="row justify-content-center">
            <form name="frm" class="col-6">
                <div class="input-group">
                    <input name="hours" class="form-control" placeholder="공부한시간">
                    <button class="btn btn-primary">점수예측</button>
                </div>
            </form>
            <div class="mt-3">
                <h5 id="predict" class="text-center"></h5>
            </div>
            <div class="mb-3 text-center">
                <img src="/ai/poly/graph" width="50%">
            </div>
        </div>
    </div>
    <script>
        $(frm).on("submit", function(e){
            e.preventDefault();
            const hours=$(frm.hours).val();
            $.ajax({
                type:"get",
                url:"/ai/poly",
                data: {hours: hours},
                success:function(data){
                    console.log(data);
                    $("#predict").html(`${hours}시간 공부했을 때 예측점수는 ${data}점 입니다.`)
                }
            })
        });
    </script>
{%endblock%}
```

- **다중선형회귀 페이지를 작성한다.**

[Flask] aiRoute.py

```
from flask import Blueprint, render_template, request, send_file
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
from sklearn.cluster import KMeans
import numpy as np
...
```

[Flask] aiRoute.py

```
...
from io import BytesIO
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams['font.family'] = 'Malgun Gothic'
matplotlib.rcParams['font.size'] = 15
matplotlib.rcParams['axes.unicode_minus'] = False
plt.switch_backend('agg')
ai = Blueprint('ai', __name__)
...
@ai.route('/page1')
def ai_page1():
    return render_template('./ai/page1.html')
...

@ai.route('/page2')
def ai_page2():
    return render_template('./ai/page2.html')
...

@ai.route('/page3')
def ai_page3():
    return render_template('./ai/page3.html')
...

@ai.route('multi')
def ai_multi():
    hours = float(request.args['hours'])
    absence = int(request.args['absence'])
    place= int(request.args['place'])

    if place == 2:
        p1=1
        p2=0
    elif place == 1:
        p1=0
        p2=1
    else:
        p1=0
        p2=0
    dataset = pd.read_csv('./data/MultipleLinearRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    #원-핫인코딩
    from sklearn.compose import ColumnTransformer
    from sklearn.preprocessing import OneHotEncoder
    ct=ColumnTransformer(transformers=[('encoder', OneHotEncoder(drop='first'), [2])], remainder='passthrough')
    X = ct.fit_transform(X)
    reg = LinearRegression()
    reg.fit(X, y)
    y_predict = reg.predict([[p1, p2, hours, absence]])
    return str(round(y_predict[0], 2))
```

[Flask]-[templates]-[ai] page3.html

```
{%extends 'base.html'%}
{%block main_area%}
    <style>
        span {
            width: 150px;
        }
    </style>
    <div class="row justify-content-center">
        <h3 class="text-center mb-5">다중선형회귀</h3>
        <form name="frm" class="col-md-8 col-lg-6 card p-3">
            <div class="input-group mb-3">
                <span class="input-group-text">장소</span>
                <select name="place" class="form-select">
                    <option value="2">집</option>
                    <option value="1">도서관</option>
                    <option value="0">카페</option>
                </select>
            </div>
            <div class="input-group mb-3">
                <span class="input-group-text">공부시간</span>
                <input name="hours" class="form-control" value="0">
            </div>
            <div class="input-group mb-3">
                <span class="input-group-text">결석횟수</span>
                <input name="absence" class="form-control"  value="0" type="number" step="1">
            </div>
            <div>
                <button class="btn btn-primary w-100">점수예측</button>
            </div>
            <div class="text-center my-3">
                <h5 id="predict"></h5>
            </div>
        </form>
    </div>
    <script>
        $(frm).on("submit", function(e){
            e.preventDefault();
            let place=$(frm.place).val();
            let hours=$(frm.hours).val();
            let absence=$(frm.absence).val();
            $.ajax({
                type:"get",
                url:"/ai/multi",
                data:{place:place, hours:hours, absence:absence},
                success:function(data){
                    $("#predict").html(`예측점수:${data}점입니다.`)
                }
            });
        });
    </script>
{%endblock%}
```

- **로지스틱회귀 페이지를 작성한다.**

[Flask] aiRoute.py

```
...
@ai.route('/page3')
def ai_page3():
    return render_template('./ai/page3.html')
...

@ai.route('/page4')
def ai_page4():
    return render_template('./ai/page4.html')

@ai.route('/logistic')
def ai_logistic():
    hours = float(request.args['hours'])
    dataset = pd.read_csv('./data/LogisticRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    classifier = LogisticRegression()
    classifier.fit(X, y)
    y_predict=classifier.predict([[hours]])
    p_predict=classifier.predict_proba([[hours]])
    y=int(y_predict[0])
    p=round(float(p_predict[0,1])*100,2)
    result = {"y":y, "p":p}
    return result

@ai.route('/logistic/graph')
def ai_logistic_graph():
    dataset = pd.read_csv('./data/LogisticRegressionData.csv')
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, -1].values

    classifier = LogisticRegression()
    classifier.fit(X, y)

    X_range = np.arange(min(X), max(X), 0.1)
    p = 1/(1 + np.exp(-(classifier.coef_ * X_range + classifier.intercept_)))
    p = p.reshape(-1)

    plt.figure(figsize=(10, 5))
    plt.scatter(X, y, color='blue')
    plt.plot(X_range, p, color='green')
    plt.plot(X_range, np.full(len(X_range), 0.5), color='red')
    plt.title('Probability by hours')
    plt.xlabel('hours')
    plt.ylabel('p')

    img = BytesIO()
    plt.savefig(img, format='png', dpi=200)
    img.seek(0)
    return send_file(img, mimetype='image/png')
```

[Flask]-[templates]-[ai] page4.html

```
{%extends 'base.html'%}
{%block main_area%}
    <div>
        <h3 class="text-center mb-5">로지스틱회귀</h3>
        <div class="row justify-content-center">
            <form name="frm" class="col-6">
                <div class="input-group">
                    <input name="hours" class="form-control" placeholder="공부한시간">
                    <button class="btn btn-primary">합격예측</button>
                </div>
                <div class="text-center my-3">
                    <h5 id="predict"></h5>
                </div>
            </form>
            <div class="mt-3">
                <h5 id="predict" class="text-center"></h5>
            </div>
            <div class="mb-3 text-center">
               <img src="/ai/logistic/graph" width="50%">
            </div>
        </div>
    </div>
    <script>
        $(frm).on("submit", function(e){
            e.preventDefault();
            hours = $(frm.hours).val();
            if(hours == ''){
                alert("시간을 입력하세요!");
                $(frm.hours).focus();
                return;
            }
            $.ajax({
                type:"get",
                url:"/ai/logistic",
                dataType:'json',
                data:{hours:hours},
                success:function(data){
                    if(data.y==1){
                        $("#predict").html(`합격을 예측합니다. (합격률:${data.p}%)`);
                    }else{
                        $("#predict").html(`불합격을 예측합니다. (합격률:${data.p}%)`);
                    }
                }
            });
        });
    </script>
{%endblock%}
```

[Flask] aiRoute.py

```
...
@ai.route('/page5')
def ai_page5():
    return render_template('./ai/page5.html')

@ai.route('/kmean/graph/<num>')
def kmean(num):
    plt.cla()
    dataset = pd.read_csv('./data/KMeansData.csv')
    X = dataset.iloc[:, :].values

    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X = sc.fit_transform(X)
    K = int(num)
    kmeans = KMeans(n_clusters=K, random_state=0, n_init=10)
    y_kmeans = kmeans.fit_predict(X)
    y_kmeans
    centers = kmeans.cluster_centers_
    X_org = sc.inverse_transform(X)
    centers_org = sc.inverse_transform(centers)
    for cluster in range(K):
        plt.scatter(X_org[y_kmeans==cluster, 0], X_org[y_kmeans==cluster, 1], s=100, edgecolor='black')
        plt.scatter(centers_org[cluster,0], centers_org[cluster,1], s=300, edgecolor='black', color='yellow', marker='s')
        plt.text(centers_org[cluster,0], centers_org[cluster,1], cluster, va='center', ha='center')
    plt.title('Scores by hours')
    plt.xlabel('hours')
    plt.ylabel('score')

    img = BytesIO()
    plt.savefig(img, format='png', dpi=200)
    img.seek(0)
    return send_file(img, mimetype='image/png')

@ai.route('/cluster/graph')
def ai_cluster_route():
    plt.cla()
    dataset = pd.read_csv('./data/KMeansData.csv')
    X = dataset.iloc[:, :].values
    from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    X = sc.fit_transform(X)
    from sklearn.cluster import KMeans
    inertia_list = []
    for i in range(1, 11):
        kmenas = KMeans(n_clusters=i, init='k-means++', random_state=0, n_init=10)
        kmenas.fit(X)
        inertia_list.append(kmenas.inertia_)
    plt.plot(range(1, 11), inertia_list)
    plt.title('Elow Method')
    plt.xlabel('n_cluster')
    plt.ylabel('inertia')

    img = BytesIO()
    plt.savefig(img, format='png', dpi=100)
    img.seek(0)
    return send_file(img, mimetype='image/png')
```

[Flask]-[templates]-[ai] page5.html

```
{%extends 'base.html'%}
{%block main_area%}
    <div>
        <h3 class="text-center mb-5">K-평균</h3>
        <div class="text-center">
            <button class="btn btn-primary" id="btn_cluster">Cluster</button>
            <button class="btn btn-primary" id="btn_kmean">K-Mean</button>
            <select name="cluster" class="p-2" id="num">
                <option>1</option>
                <option>2</option>
                <option>3</option>
                <option selected>4</option>
                <option>5</option>
            </select>
        </div>
        <div class="text-center">
            <img src="/ai/kmean/graph/4" id="graph" width="50%">
        </div>
    </div>
    <script>
        $("#btn_cluster").on("click", function(){
            $("#graph").attr("src", "/ai/cluster/graph")
        });

        $("#btn_kmean").on("click", function(){
            const num = $("#num").val();
            $("#graph").attr("src", "/ai/kmean/graph/" + num)
        });
    </script>
{%endblock%}
```

- K-평균의 실행 결과는 아래와 같다.