#### 딥러닝/클라우드

Chapter 4

Regression

Sejong Oh

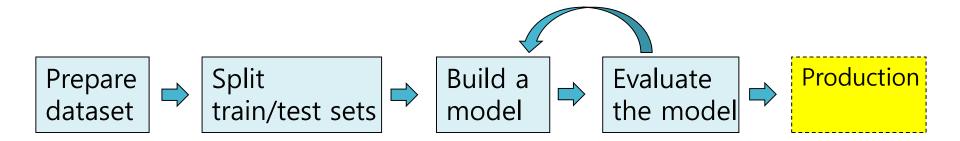
DANKOOK UNIVERSITY

#### **Contents**

- 1. Simple linear regression
- 2. Multiple linear regression
- 3. Logistic regression

#### **Summary**

Building machine learning model



- Fitting
- Learning
- Training

## **Summary**

- Machine learning 분류
  - 지도학습 (supervised learning)
    - 회귀(regression)
    - 분류(classification) 등

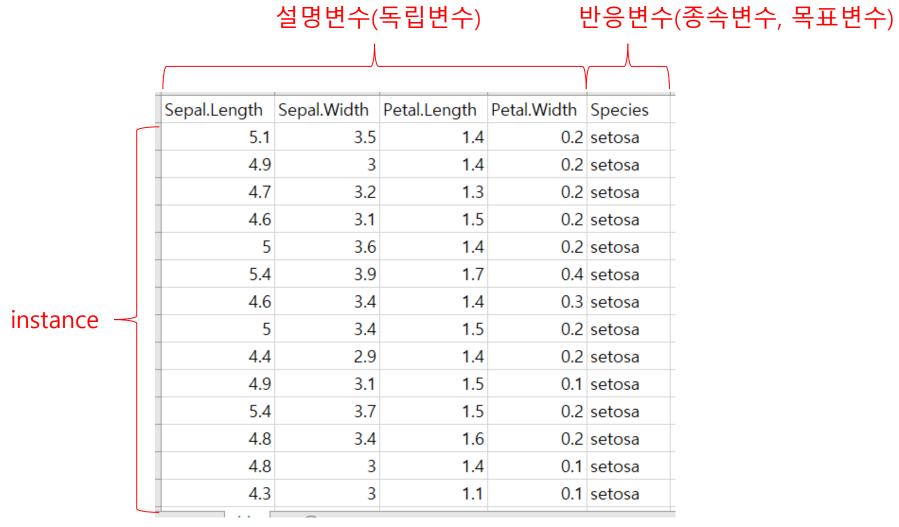
- 비지도학습(unsupervised learning)
  - 군집화(clustering)

○ 강화학습(Reinforcement learning)

#### **Summary**

dataset

Feature, variable



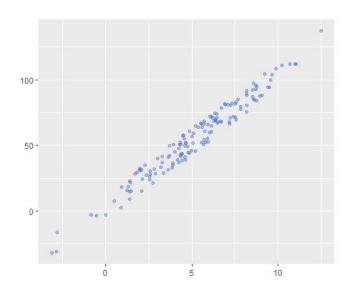


#### • 정의

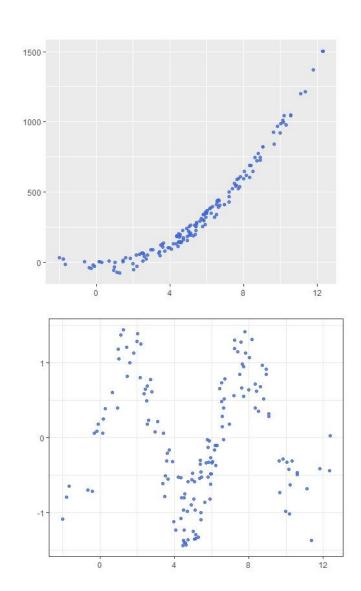
- 종속 변수(y) 와 독립변수(x) 사이의 선형 관계를 파악하고 이를 예측 에 활용하는 방법
- 예) 기온(x) 과 아이스 크림 판매량(y) 사이의 관계식을 찾아낸다. 이를 이용하여 내일의 예상 기온으로 부터 예상 아이스크림 판매량을 예측한다. => 필요한 아이스크림 재료의 양을 예측할 수 있다.
- 기온(x) 과 아이스 크림 판매량(y) 사이의 관계식을 모델(model) 이라고 한다. (회귀 모델, 예측 모델)
- 단순 선형 회귀식은 다음과 같은 형태

$$y=Wx+b$$

상수인 W와 b 를 찾는 것이 학습(훈련) 목표



$$y=Wx+b$$





#### • 정의

○ 기온(x) 과 아이스 크림 판매량(y) 사이의 관계식이 다음과 같다고 하자

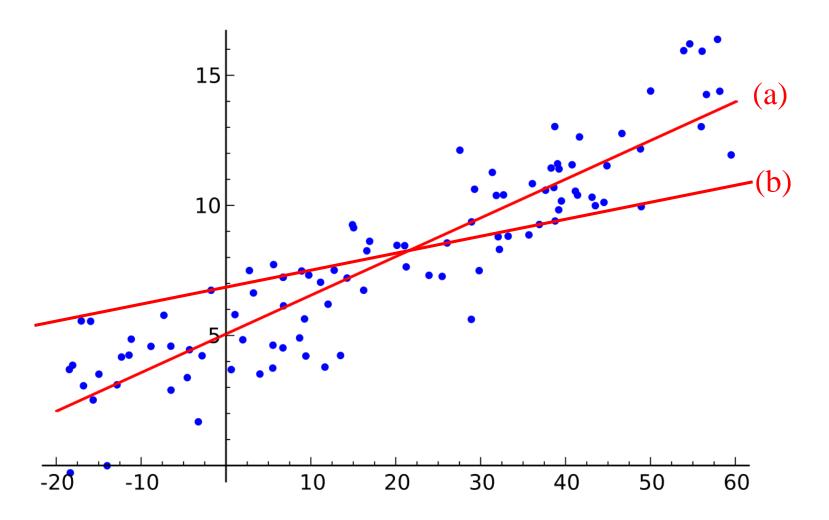
$$y = 2500 x + 45$$

내일의 예상 기온이 32 도 이면 내일의 아이스크림 예상 판매량은 다음과 같이 예측될 수 있다

$$y = 2500 \times 32 + 45 = 80045$$

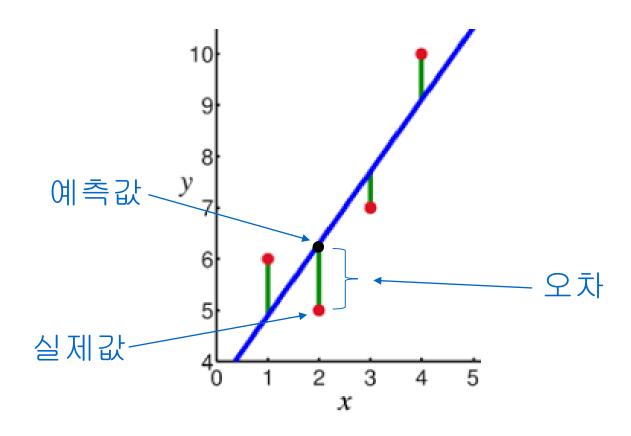
- 현실세계에서는 두 변수가 선형 관계에 있는 경우가 많아서 선형회귀 분석이 유용하다
- 두변수가 선형 관계에 있는지 알아보는 방법 : 산점도, 상관계수

● 회귀식에서 W와 b를 찾는 방법



7

● 회귀식에서 W와 b를 찾는 방법



Example problem: car speed vs stopping distance

4	Α	В	
1	speed	dist	
2	4	2	
3	4	10	
4	7	4	
5	7	22	
6	8	16	
7	9	10	
8	10	18	
9	10	26	
10	10	34	
11	11	17	
12	11	28	
12	10	1/	

cars.csv

- speed: Speed (mph)
- dist: Stopping distance (ft)



https://grimmermotors.co.nz/car-brakes-take-longer-to-stop-repairs-hamilton/

#### Python code

```
# load required modules

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

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

```
# prepare dataset

cars = pd.read_csv('D:/data/cars.csv')

print(cars)

speed = cars['speed'].to_frame()

dist = cars['dist']

to_frame(): 1차원 데이터
(Series)를 2차원 데이터
(dataframe)로 변경
```

```
In [49]: print(cars)
                                         In [55]: speed
                                                              In [57]: dist
    speed dist
                                         Out[55]:
                                                              Out[57]:
0
         4
                                                                       2
1
              10
                                                              1
                                                                      10
                                         2
2
             4
                                                                       4
                                                                      22
3
              22
                                                                      16
              16
4
                                                                      10
5
              10
                                                10
                                                                      18
              18
6
        10
                                                10
                                                                      26
7
       10
              26
                                                10
                                                                      34
8
       10
              34
                                                11
                                                              9
                                                                      17
9
       11
              17
                                         10
                                                11
                                                              10
                                                                      28
10
              28
       11
                                         11
                                                12
                                                              11
                                                                      14
11
        12
              14
                                         12
                                                12
                                                              12
                                                                      20
12
        12
               20
```

50x2 50x1 50x1



```
# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train_test_split(speed, dist, test_size=0.2, random_state=123)
```

\ 하나의 명령문을 여러줄에 걸쳐서 작성할 때 연결기호

	In [5	<b>5</b> ]: sp	eed	In [5	<b>7</b> ]: di	st
	Out[ <b>55]</b> :			Out[5	7]:	1
	0	4		0	2	
	1	4		1	10	
	2	7		2	4	
train_X	3	7		3	22	train v
trairi_X	4	8		4	16	trairi_y
	5	9		5	10	train_y  train_y
	6	10		6	18	
	7	10		7	26	
	8	10		8	34	
	9	11		9	17	
test_X	10	11		10	28	test v
	11	12		11	14	
	12	12		12	20	
	1			I		

random\_state=123

Train, test 를 나눌때 임의로 나누기 때문에 실행 할 때 마다 결과가 달라진다.
실행시 같은 결과가 나오도록 하는 것이 random\_state.

```
# Define learning method
model = LinearRegression()

# Train the model using the training sets
model.fit(train_X, train_y)

# Make predictions using the testing set
pred_y = model.predict(test_X)
print(pred_y)
```

```
In [67]: test X
Out[67]:
                                     In [66]: print(pred_y)
array([[11],
                                     [[25.221513]
       [12],
                                      [29.18014184]
       [17],
                                      [48.97328605]
       [24],
                                      [76.68368794]
       [13],
                                      [33.13877069]
       [4],
                                      [-2.48888889]
       [20],
                                      [60.84917258]
       [12],
                                      [29.18014184]
       [17],
                                      [48.97328605]
       [10]], dtype=int64)
                                      [21.26288416]]
```

```
# prediction test
print(model.predict([[13]]))  # when speed=13
print(model.predict([[20]]))  # when speed=20
```

```
In [68]: print(model.predict([[13]]))  # when speed=13
[[33.13877069]]
In [69]: print(model.predict([[20]]))  # when speed=20
[[60.84917258]]

In [70]: shape([13])
Out[70]: (1,)
In [71]: shape([[13]])
Out[71]: (1, 1)
```

```
>>> print('Coefficients: \{0:.2f\}, Intercept \{1:.3f\}'\
... .format(model.coef_[0], model.intercept_))
Coefficients: 3.96, Intercept -18.323
y = Wx + b
y = 3.96x - 18.323
dist = 3.96 \times speed - 18.323
```

Model evaluation 1

```
# The mean squared error
print('Mean squared error: {0:.2f}'.\
    format(mean_squared_error(test_y, pred_y)))
```

$$MSE = \left[\sum_{k=0}^{n} (test_y_k - pred_y_k)^2\right] / n$$
 실제값 모델에 의한 예측값

이 값을 작을 수록 정확한 모델임

Model evaluation 2

```
R<sup>2</sup> score : 최대값은 1 (모델의 예측 결과와 실제값이 완전히 일치할 때)이 값이 1 에 가까울수록 좋은 모델
완전히 잘못된 모델은 (-)값이 나올수도 있음
```



• <a href="https://scikit-learn.org/stable/modules/model\_evaluation.html">https://scikit-learn.org/stable/modules/model\_evaluation.html</a>

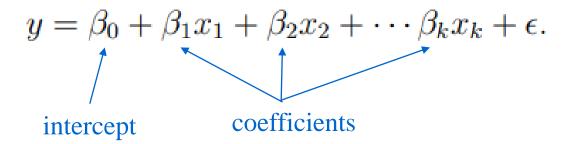
Regression	
'explained_variance'	metrics.explained_variance_score
'max_error'	metrics.max_error
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_root_mean_squared_error'	metrics.mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
' <u>r2'</u>	metrics.r2_score
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance
'neg_mean_absolute_percentage_error'	metrics.mean_absolute_percentage_error
4	



#### • 모델 시각화

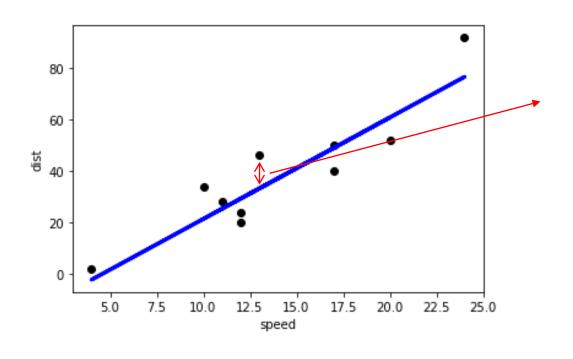
```
# Plot outputs
plt.scatter(test_X, test_y, color='black')
plt.plot(test_X, pred_y, color='blue', linewidth=3)
plt.xlabel('speed')
plt.ylabel('dist')
plt.show()
                           80
                           60
                           20
                               5.0
                                    7.5
                                       10.0
                                               15.0
                                                    17.5
                                                        20.0
                                           12.5
                                                            22.5
                                              speed
```

- Definition
  - Multiple linear regression
  - 독립변수(설명변수)가 2개 이상인 경우
  - 예) 키(x1) 와 몸무게(x2)를 가지고 혈당수치(y)를 를 예측
    - 독립변수(설명변수) : 키, 몸무게
    - 종속변수(반응변수) : 혈당수치
  - 중선형 회귀식의 형태





#### Motivation



이 차이는 어디서 발생을 하는가?

제동거리는 단순히 주행속도에 의해서만 결정이 되는가

타이어 년수 브레이크 성능 당일 풍속

. . .

단순 선형회귀 : (주행속도) → (제동거리) 중 선형회귀 : (주행속도,타이어년수,당일풍속) → (제동거리)

- 사례: 연봉 예측 모델
  - 특정 직군의 연봉을 3가지 변수(교육년수, 여성비율, 평판)를 가지고 예측해보자
  - 데이터셋 : prestige.csv

	A	В	С	D	Е	F	G
1		education	income	women	prestige	census	type
2	gov.administrators	13.11	12351	11.16	68.8	1113	prof
3	general.managers	12.26	25879	4.02	69.1	1130	prof
4	accountants	12.77	9271	15.7	63.4	1171	prof
5	purchasing.officers	11.42	8865	9.11	56.8	1175	prof
6	chemists	14.62	8403	11.68	73.5	2111	prof
7	physicists	15.64	11030	5.13	77.6	2113	prof
8	biologists	15.09	8258	25.65	72.6	2133	prof
9	architects	15.44	14163	2.69	78.1	2141	prof
10	civil.engineers	14.52	11377	1.03	73.1	2143	prof
	지어그 -						

식업군

교육년수 / 여성비율 \

수입(연봉) 직업에 대한 평판



04.regression\_multiple.py

```
import pandas as ps
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
```

```
# Load the prestge dataset

df = ps.read_csv('D:/data/prestige.csv')
print(df)

df_X = df[['education','women','prestige']]

df_y = df['income']
```

```
In [127]: print(df)
            Unnamed: 0 education income women prestige census type
     gov.administrators
                           13.11
                                   12351
                                         11.16
                                                    68.8
                                                           1113
                                                                prof
0
1
       general.managers
                           12.26
                                   25879
                                         4.02
                                                   69.1
                                                           1130
                                                                prof
            accountants
                           12.77
                                  9271
                                         15.70
                                                   63.4
                                                           1171
2
                                                                prof
    purchasing.officers
                                  8865 9.11
                                                    56.8
                                                           1175
                           11.42
                                                                prof
              chemists
                           14.62
                                                    73.5
                                                           2111
                                                                prof
4
                                    8403
                                         11.68
                             . . .
                                    . . .
                                                     . . .
                            7.58
97
           bus.drivers
                                   5562 9.47
                                                    35.9
                                                           9171
                                                                  bc
98
          taxi.drivers
                            7.93
                                  4224 3.59
                                                   25.1
                                                           9173
                                                                  bc
99
           longshoremen
                            8.37
                                    4753
                                         0.00
                                                    26.1
                                                           9313
                                                                  bc
100
           typesetters
                           10.00
                                    6462
                                         13.58
                                                   42.2
                                                           9511
                                                                  bc
           bookbinders
101
                            8.55
                                   3617 70.87
                                                   35.2
                                                           9517
                                                                  bc
[102 rows x 7 columns]
```

```
In [128]: df_X
Out[128]:
    education women prestige
        13.11 11.16
                          68.8
0
1
        12.26 4.02
                          69.1
2
        12.77 15.70
                          63.4
        11.42 9.11
                          56.8
4
        14.62 11.68
                          73.5
. .
                          35.9
97
         7.58 9.47
98
         7.93 3.59
                          25.1
99
         8.37
                          26.1
                0.00
        10.00 13.58
                          42.2
100
101
         8.55 70.87
                          35.2
[102 rows x 3 columns]
```

```
In [129]: df_y
Out[129]:
0
       12351
       25879
        9271
        8865
        8403
       . . .
97
        5562
98
        4224
99
        4753
100
        6462
101
        3617
Name: income, Length: 102, dtype: int64
```

```
# Split the data into training/testing sets
train_X, test_X, train_y, test_y = \
    train test split(df X, df y, test size=0.2,
                      random state=123)
# Dfine learning model
model = LinearRegression()
# Train the model using the training sets
model.fit(train X, train y)
# Make predictions using the testing set
pred y = model.predict(test X)
print(pred y)
In [130]: print(pred_y)
```

```
# The coefficients & Intercept
print('Coefficients: {0:.2f},{1:.2f},{2:.2f} Intercept {3:.3f}'\
     .format(model.coef_[0], model.coef_[1], model.coef_[2],\
             model.intercept ))
# The mean squared error
print('Mean squared error: {0:.2f}'.\
      format(mean squared error(test y, pred y)))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2 score(test y, pred y))
```

```
Coefficients: 108.88,-48.80,168.59 Intercept -775.183
Mean squared error: 3007520.31
Coefficient of determination: 0.74
```

```
income = 108.88×education -48.80×women + 168.59× prestige – 775.183
```





education	women	prestige			
11.44	8.13	54.1			

Income?

```
# Test single data
my_test_x = np.array([11.44,8.13,54.1]).reshape(1,-1)
my_pred_y = model.predict(my_test_x)
print(my_pred_y)
```

```
In [150]: print(my_pred_y)reshape(1,-1):[9194.5642717]행의 갯수는 1, 열은 알아서 맞춤
```

Income = 9194.5642717

- 일반적인 회귀 문제에서는 종속변수가 수치데이터(양적 자료)임
- 예측 해야 할 종속 변수가 수치데이터가 아닌 범주형 데이터 (Yes or No, Patient or Healthy) 일 때 이를 회귀방법으로 해결하고자 하는 시도를 ' 로지스틱 회귀'라고 한다.
- 예) iris 데이터셋에서 4개의 측정 데이터로 부터 품종(Species) 를 예측
- \* 범주나 그룹을 예측하는 문제를 '분류(classification)' 문제 라고 한다

Iris 품종 예측

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
           5.1
                       3.5
                                    1.4
                                                0.2
                                                     setosa
2
3
4
5
6
           4.9
                       3.0
                                                     setosa
          4.7
                       3.2
                                                0.2 setosa
          4.6
                       3.1
                                                0.2 setosa
          5.0
                       3.6
                                    1.4
                                                0.2 setosa
           5.4
                       3.9
                                                0.4 setosa
                                    1.7
                꽃받침
                                        꽃잎
                                                      품종
```

- iris dataset 은 3개의 품종이 있음
- Scikit-learn 에서 제공하는 iris 품종정보는 0,1,2 로 표시됨

```
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
In [180]: print(iris_X.shape) # (150, 4)
(150, 4)
```

```
# Define learning model
model = LogisticRegression()
# Train the model using the training sets
model.fit(train X, train y)
# Make predictions using the testing set
pred y = model.predict(test X)
print(pred y)
acc = accuracy_score(test_y, pred_y)
print('Accuracy : {0:3f}'.format(acc))
```

```
In [179]: print('Accuracy : {0:3f}'.format(acc))
Accuracy : 0.977778
Accuracy = \frac{ 예측과 정답이 일치하는 instance 수}{ 전체 test instance 수}
```



#### Note

로지스틱 회귀의 경우에도 종속 변수가 숫자여야 하기 때문에 문자형으로 되어 있는 범주데이터는 숫자(0,1,2,..)로 변환한 후 작업을 해야한다.
 Scikit-learn에서는숫자변환 안해도됨

```
from sklearn.preprocessing import LabelEncoder
import numpy as np
number = LabelEncoder()

label_str = np.array(['M','F','M','F','M'])
label_num = number.fit_transform(label_str).astype('int')
print(label_str)
print(label_num)
```

```
In [198]: print(label_str)
['M' 'F' 'M' 'F' 'M']
In [199]: print(label_num)
[1 0 1 0 1]
```

# [실습 1] simple linear regression

- 1. Iris 데이터셋에서 꽃잎의 길이(Petal.Length)로 꽃잎의 폭(Petal.Width)을 예측하는 회귀 모델을 만드시오
- 2, 회귀 모델의 mean\_squared\_error 와 r2\_score 값을 보이시오
- 3. 회귀 모델의 coefficients 와 Intercept 값을 보이시오
- 4. 회귀 모델을 이용하여 꽃잎의 길이가 1.0, 1.2, 1.4 일 때 꽃잎의 폭을 예측하여 보시오.

\*\* 모든 실습 과제에서 train:test = 7:3 으로, random\_state = 1234 로 한다

## [실습 2] 중회귀

- 1. 실습용으로 제공한 BostonHousing 데이터셋은 보스턴 지역의 지역정보 및 평균주택 가격 (medv) 정보를 담고 있다. 다른 변수들을 이용하여 medv 를 예측하는 모델을 만드시오 (feature중 chas는 제외)
- 2, 회귀 모델의 mean\_squared\_error 와 r2\_score 값을 보이시오
- 3. 회귀 모델의 coefficients 와 Intercept 값을 보이시오

1	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	Istat (	medv
2	0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
3	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
4	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
5	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
6	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
7	0.02985	0	2.18	0	0.458	6.43	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
8	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9
9	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	396.9	19.15	27.1
10	0.21124	12.5	7.87	0	0.524	5.631	100	6.0821	5	311	15.2	386.63	29.93	16.5
11	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2	386.71	17.1	18.9
12	0.22489	12.5	7.87	0	0.524	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15
13	0 11747	12 5	7 87	n	0 524	6 009	82 9	6 2267	5	311	15.2	396.9	13 27	189

# [실습 3] logistic regression

- 1. scikit-learn 에 포함된 wine 데이터셋으로 부터 wine 의 등급을 분류하는 모델을 만드시오
- 2. 만들어진 모델의 예측 정확도를 보이시오
  - train set 에 대한 예측 정확도
  - test set 에 대한 예측 정확도