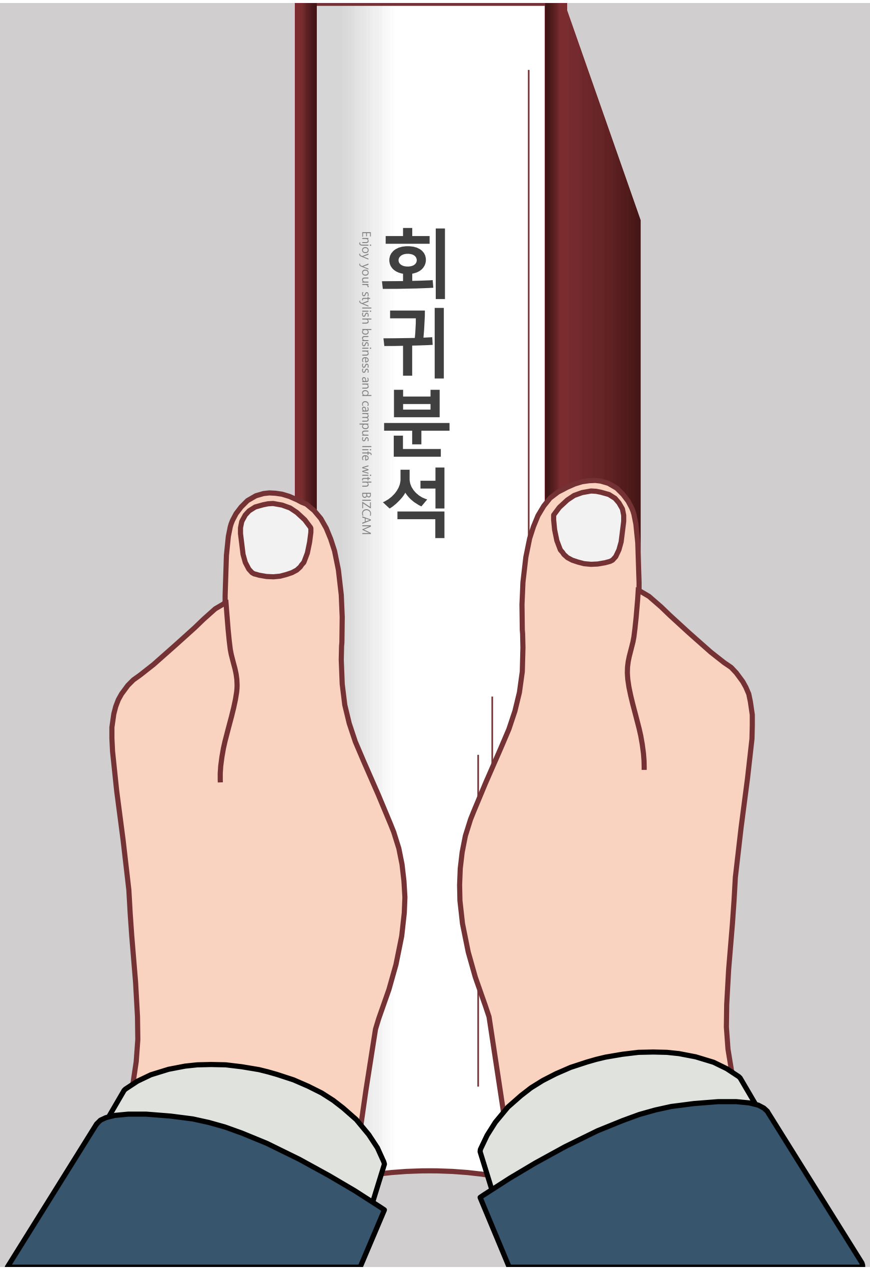


2021210088 허지혜

1. LOGISTIC REGRESSION

로지스틱 회귀 알고리즘

: 회귀를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측하고 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해주는 지도 학습 알고리즘

예) 스팸 메일 분류기
메일이 스펨일 확률 0.5 이상 -> 스팸으로 분류
메일이 스펨일 확률 0.5 미만 -> 정상으로 분류

로지스틱 회귀 단계

1. 모든 속성들의 계수와 절편을 0으로 초기화
2. 각 속성들의 값에 계수를 곱해 log-odds를 구한다.
3. Log-odds 를 sigmoid 함수에 넣어 [0,1] 범위의 확률을 구한다.

1. LOGISTIC REGRESSION

선형 회귀 : 각 속성의 값에다가 계수를 곱하고 절편을 더해 예측 값을 구한다.

로지스틱 회귀 : 선형 회귀와 비슷한데
마지막에 예측 값 말고 log-odds를 구한다.

$$Odds = \frac{P(event\ occuring)}{P(event\ not\ occuring)}$$

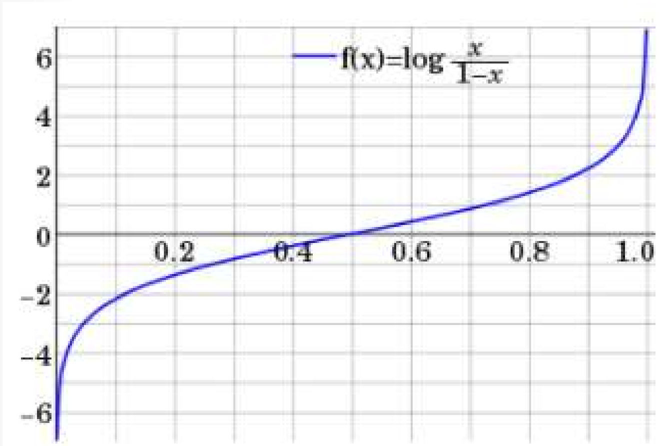
Odds 값에 log를 취한 값이 log-odds 값이다.

$$Log - Odds = Log\frac{P(event\ occuring)}{P(event\ not\ occuring)}$$

확률 [0,1] $\neq$ 범위 : $(-\infty, +\infty)$

$y = ax + b$

log-odds 범위 : $(-\infty, +\infty)$



P = ?

1. LOGISTIC REGRESSION

Enjoy your stylish business and campus life with BIZCAM

$$y = ax + b$$

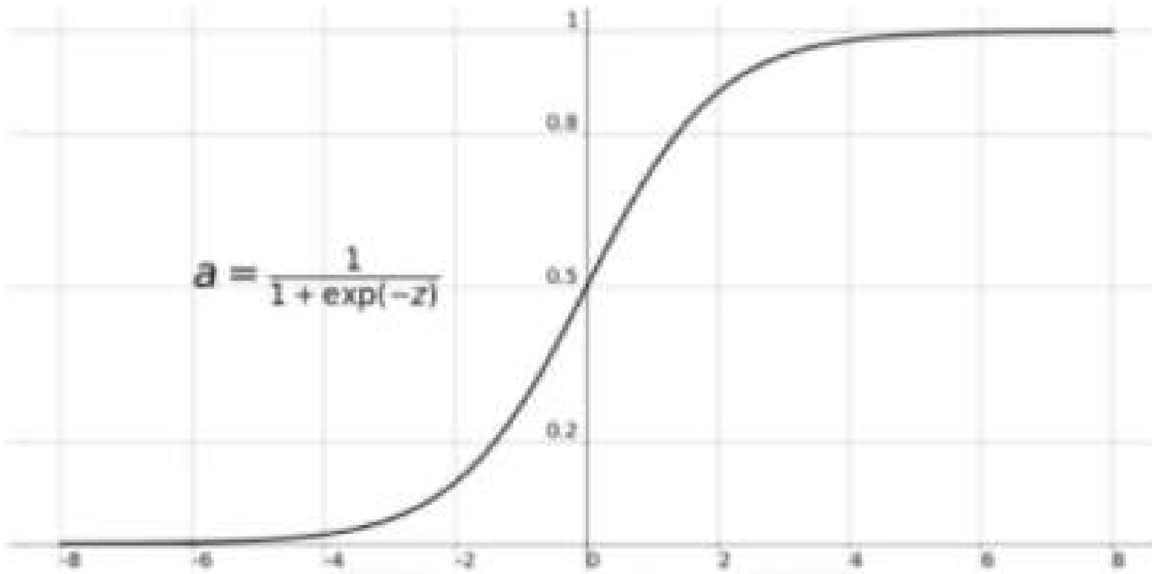
$$\mathrm{Log}\left(\frac{P}{1-P}\right) = ax + b$$

$$e^{\mathrm{Log}\left(\frac{P}{1-P}\right)} = e^{ax+b}$$

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

Sigmoid 함수

$$P = \frac{e^{ax+b}}{1+e^{ax+b}} = \frac{1}{1+e^{-(ax+b)}}$$



1. LOGISTIC REGRESSION

예측할 범주형 변수의 클래스가 3개 이상인 경우,

1. One Vs All(OVA)

Red/blue/green 3개를
Red/blue,green
Red,blue/green
Red,green/blue
를 구분하는 3개의 이진 분류기를 만들어 확률이 가장 높게 나온 범주를 선택한다.

2. Multinomial

이진 분류면 앞처럼 sigmoid 함수로 변하고, 일반화하면 특정 클래스를 기준으로 삼지 않고 softmax 함수 형태로 바뀌어서 사용한다.

$$Pr(Y=k) = \frac{e^{\beta_k^T X}}{e^{\beta_1^T X} + \dots + e^{\beta_K^T X}}$$

EX1. IRIS DATA

Enjoy your stylish business and campus life with BIZCAM

```
# 데이터 불러오기
import seaborn as sns # seaborn을 불러오고 SNS로 축약
import numpy as np
iris = sns.load_dataset('iris')  # iris라는 변수명으로 Iris data를 download
print(iris.head())
X = iris.drop('species', axis=1) # 독립변수 = 'species'열을 drop하고 input X를 정의
y = iris['species']  # 종속변수 = 'species'열
print(np.unique(y))
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
['setosa' 'versicolor' 'virginica']
```

IRIS 데이터

Sepal Length(꽃받침의 길이)
Sepal Width(꽃받침의 너비)
Petal Length(꽃잎의 길이)
Petal Width(꽃잎의 너비)
Species(꽃의 종류)

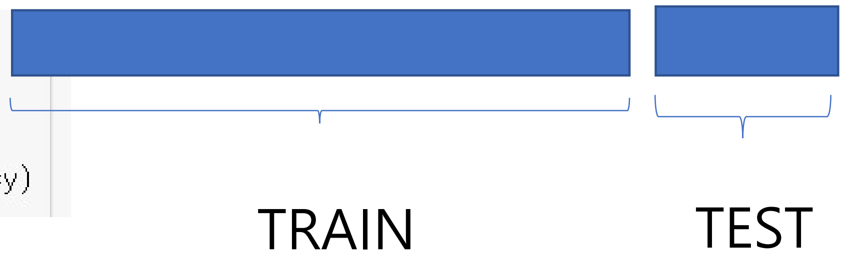
EX1. IRIS DATA

Enjoy your stylish business and campus life with BIZCAM

```
# y data를 범주형으로 변환
import numpy as np
from sklearn.preprocessing import LabelEncoder    # LabelEncoder() method를 불러옴
label = LabelEncoder()
y = label.fit_transform(iris['species'].values) # species 열의 문자열을 categorical 값으로 전환
print(np.unique(y))
```

```
[0 1 2]
```

```
# 전체 data를 training set과 test set으로 split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=1, stratify=y)
```



EX1. IRIS DATA

Enjoy your stylish business and campus life with BIZCAM

```
# 표준화
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

데이터의 특성을 평균이 0, 표준편차가 1이 되도록 변환한다.

```
# Logistic regression
from sklearn.linear_model import LogisticRegression
Logit = LogisticRegression(C=1e2, random_state=1)  # C = 1/λ. 디폴트: L2, One-versus-Rest.
l_1=Logit.fit(X_train_std, y_train)
y_train_pred = Logit.predict(X_train_std)
y_test_pred = Logit.predict(X_test_std)
```

EX1. IRIS DATA


```
# Accuracy score
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,y_test_pred))
print(accuracy_score(y_train,y_train_pred))
```

```
1.0
0.9809523809523809
```

```
# Confusion matrix
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_test_pred))  # Confusion matrix
```

```
[[15  0  0]
 [ 0 15  0]
 [ 0  0 15]]
```

Confusion Matrix

	0 예측	1 예측	2 예측
0 실제	15	0	0
1 실제	0	15	0
2 실제	0	0	15

EX1. IRIS DATA

Enjoy your stylish business and campus life with BIZCAM

```
from sklearn.linear_model import LogisticRegression
Logit = LogisticRegression(C=1e2, random_state=1, max_iter=200)  # C = 1/λ
Logit.fit(X_train, y_train)
y_train_pred = Logit.predict(X_train)
y_test_pred = Logit.predict(X_test)
y_test_pred_proba=Logit.predict_proba(X_test)
print(y_test_pred[:5])
print(y_test_pred_proba[:5])
```

```
[2 0 0 2 1]
[[5.84256910e-10 5.70998618e-04 9.99429001e-01]      2
 [9.98969344e-01 1.03065564e-03 1.73821807e-20]      0
 [9.99784715e-01 2.15285394e-04 3.22356613e-21]      0
 [2.85530842e-05 4.44101814e-01 5.55869633e-01]      2
 [2.47870967e-05 9.99324950e-01 6.50262844e-04]]     1
```

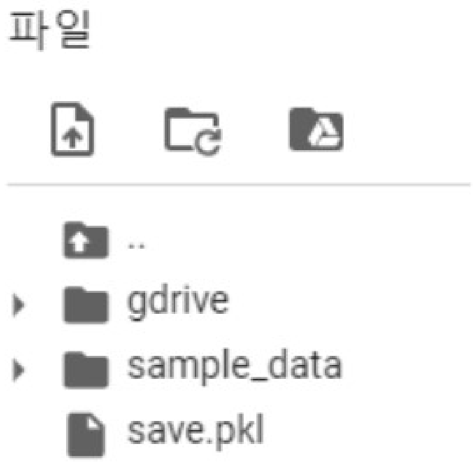
EX1. IRIS DATA

Enjoy your stylish business and campus life with BIZCAM

```
[11] !pip install joblib

     Requirement already satisfied: joblib ir
```

```
[14] import os
     os.getcwd()

     '/content'
```

```
[17] import joblib
     joblib.dump(Logit,'/content/save.pkl')

     ['/content/save.pkl']
```



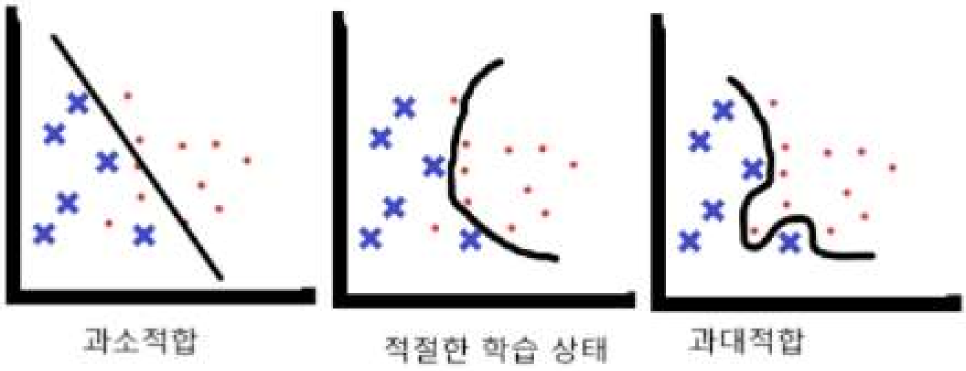
Joblib.dump()
: 임의의 객체를 pkl 형식으로 저장해준다.

EX1. IRIS DATA


```
[18] logit_from_joblib=joblib.load('/content/save.pkl')
     logit_pred=logit_from_joblib.predict(X_test)
     print(accuracy_score(y_test,logit_pred))
     print(confusion_matrix(y_test,logit_pred))
     print(logit_pred)
```

```
1.0
[[15  0  0]
 [ 0 15  0]
 [ 0  0 15]]
[2 0 0 2 1 1 2 1 2 0 0 2 0 1 0 1 2 1 1 2 2 0 1 2 1 1 1 2 0 2 0 0 1 1 2 2 0
 0 0 1 2 2 1 0 0]
```

2. 과대 적합에 대한 규제



과대 적합?

훈련 데이터에서는 잘 동작하지만 테스트 데이터(본 적이 없는 데이터)에서는 잘 일반화되지 않는 현상

규제(Regularization)

공산성 (특성 간 높은 상관 관계)를 다루거나 데이터에서 잡음을 제거하여 과대 적합을 방지할 수 있는 방법이다. 규제는 과도한 파라미터 값을 제한하기 위해 추가적인 정보를 주입하는 개념이다.

2. 과대 적합에 대한 규제

로지스틱 회귀분석 안 파라미터에서는
penalty와 C가 있다.

Penalty : 규제의 유형을 설정
C : 규제 강도를 조절하는 alpha의 역수

LogisiticRegression(penalty= *'l2'*, *,
dual=*False*, tol=*0.0001*, C= *1.0*, fit_intercept=*True*,
intercept_scaling=*1*, class_weight=*None*,
random_state=*None*, solver= *'lbfgs'*,
max_iter=*100*, multi_class=*'auto'*, verbose=*0*,
warm_start=*False*, n_jobs=*None*, l1_ratio=*None*)

위를 비용함수에 넣어서 계산 값을
조정하여 과대 적합을 방지한다.

L1 규제

$$L1 = C_0 + \frac{\lambda}{2N}\sum_{w}|W|$$

L2 규제

$$L2 = C_0 + \frac{\lambda}{2N}\sum_{w}W^2$$

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
[20]  import pandas as pd
      import numpy as np
```

```
[21]  # 데이터 불러오기. y값은 이미 범주형으로 되어있음.
      dat_wine=pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/'
                           'wine/wine.data',header=None)
      print(dat_wine.head())
      dat_wine.columns = ['class label', 'alchohol', 'malic acid', 'ash',
                          'alcalinity of ash', 'magnesium', 'total phenols',
                          'flavanoids', 'nonflavanoid phenols',
                          'proanthocyanins', 'color intensity', 'hue',
                          'OD208', 'proline']  # Column names
      print('class label:', np.unique(dat_wine['class label']))  # Class 출력
      dat_wine.head()
```

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

Pirnt(dat_wine.head())

```
   0      1     2     3     4    5  ...     8     9    10    11    12    13
0  1  14.23  1.71  2.43  15.6  127  ...  0.28  2.29  5.64  1.04  3.92  1065
1  1  13.20  1.78  2.14  11.2  100  ...  0.26  1.28  4.38  1.05  3.40  1050
2  1  13.16  2.36  2.67  18.6  101  ...  0.30  2.81  5.68  1.03  3.17  1185
3  1  14.37  1.95  2.50  16.8  113  ...  0.24  2.18  7.80  0.86  3.45  1480
4  1  13.24  2.59  2.87  21.0  118  ...  0.39  1.82  4.32  1.04  2.93   735

[5 rows x 14 columns]
class label: [1 2 3]
```

Dat_wine.head()

	class label	alchohol	malic acid	ash	alcalinity of ash	magnesium	total phenols	flavanoids	nonflavanoid phenols	proanthocyanins
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	2.29
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	2.81
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	2.18
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	1.82

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
# 전체 data를 training set과 test set으로 split
from sklearn.model_selection import train_test_split
X, y = dat_wine.iloc[:,1:].values, dat_wine.iloc[:,0].values
X_train, X_test, y_train,y_test = \
    train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
```

```
# 표준화
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
X_train_std = std.fit_transform(X_train)
X_test_std = std.transform(X_test)
```

```
# Logistic Regression with L2 or L1 Regularization
from sklearn.linear_model import LogisticRegression
lr2_10 = LogisticRegression(penalty='l2', C=10.0)   # L2 with C(=1/λ)=10
lr2_1 = LogisticRegression(penalty='l2', C=1.0)     # L2 with C(=1/λ)=1
lr2_0_1 = LogisticRegression(penalty='l2', C=0.1)   # L2 with C(=1/λ)=0.1
lr1_10 = LogisticRegression(penalty='l1', C=10.0, solver='liblinear')  # L1 with C(=1/λ)=10
lr1_1 = LogisticRegression(penalty='l1', C=1.0, solver='liblinear')    # L1 with C(=1/λ)=1
lr1_0_1 = LogisticRegression(penalty='l1', C=0.1, solver='liblinear')  # L1 with C(=1/λ)=0.1
```

EX2. WINE DATA


```
lr2_10.fit(X_train, y_train)
print('Training accuracy with L2 and λ=0.1:', lr2_10.score(X_train, y_train))
print('Test accuracy with L2 and λ=0.1:', lr2_10.score(X_test, y_test))

lr2_1.fit(X_train, y_train)  # warning..
print('Training accuracy with L2 and λ=1:', lr2_1.score(X_train, y_train))
print('Test accuracy with L2 and λ=1:', lr2_1.score(X_test, y_test))

lr2_0_1.fit(X_train, y_train)
print('Training accuracy with L2 and λ=10:', lr2_0_1.score(X_train, y_train))
print('Test accuracy with L2 and λ=10:', lr2_0_1.score(X_test, y_test))
```

```
Training accuracy with L2 and λ=0.1: 0.9838709677419355
Test accuracy with L2 and λ=0.1: 0.9074074074074074
Training accuracy with L2 and λ=1: 0.9758064516129032
Test accuracy with L2 and λ=1: 0.9259259259259259
Training accuracy with L2 and λ=10: 0.9758064516129032
Test accuracy with L2 and λ=10: 0.9074074074074074
```

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
lr1_10.fit(X_train, y_train)
print('Training accuracy with L1 and λ=0.1:', lr1_10.score(X_train, y_train))
print('Test accuracy with L1 and λ=0.1:', lr1_10.score(X_test, y_test))

lr1_1.fit(X_train, y_train)
print('Training accuracy with L1 and λ=1:', lr1_1.score(X_train, y_train))
print('Test accuracy with L1 and λ=1:', lr1_1.score(X_test, y_test))

lr1_0_1.fit(X_train, y_train)
print('Training accuracy with L1 and λ=10:', lr1_0_1.score(X_train, y_train))
print('Test accuracy with L1 and λ=10:', lr1_0_1.score(X_test, y_test))
```

```
Training accuracy with L1 and λ=0.1: 1.0
Test accuracy with L1 and λ=0.1: 0.9074074074074074
Training accuracy with L1 and λ=1: 0.9838709677419355
Test accuracy with L1 and λ=1: 0.9074074074074074
Training accuracy with L1 and λ=10: 0.9354838709677419
Test accuracy with L1 and λ=10: 0.8888888888888888
```

EX2. WINE DATA


```
print(lr2_10.intercept_)
print(lr2_1.intercept_)
print(lr2_0_1.intercept_)

print(lr2_10.coef_)
print(lr2_1.coef_)
print(lr2_0_1.coef_)
```

```
[-0.09116465  0.22505155 -0.1338869 ]
[-0.04189598  0.09954385 -0.05764787]
[-0.0309764   0.06088599 -0.02990959]
```

L2 규제, C=10, coef

```
[[-0.36216032  0.52066013  0.33179466 -0.44162529 -0.07071387  0.72966071
   1.39451652 -0.1096653   0.51589549  0.20122888 -0.00897607  0.97827549
   0.01535182]
 [ 1.42855357 -1.58159886 -0.19927044  0.40908749 -0.02458016  0.14964267
   0.48873105  0.19752745  0.26519887 -2.192684    0.53108438  0.57453491
  -0.01814738]
 [-1.06639325  1.06093873 -0.13252422  0.0325378   0.09529403 -0.87930338
  -1.88324757 -0.08786215 -0.78109436  1.99145512 -0.52210831 -1.5528104
   0.00279556]]
```

EX2. WINE DATA


```
print(lr2_10.intercept_)
print(lr2_1.intercept_)
print(lr2_0_1.intercept_)

print(lr2_10.coef_)
print(lr2_1.coef_)
print(lr2_0_1.coef_)
```

```
[-0.09116465  0.22505155 -0.1338869 ]
[-0.04189598  0.09954385 -0.05764787]
[-0.0309764   0.06088599 -0.02990959]
```

L2 규제, C=1, coef

```
[[-1.71041809e-01  1.91611125e-01  1.28111028e-01 -3.28204117e-01
  -3.08786127e-02  3.10327283e-01  5.97743767e-01 -4.90519057e-02
   2.18950161e-01  4.72643281e-02 -3.77134551e-04  4.12541432e-01
   1.06590357e-02]
 [ 5.86780948e-01 -7.21731557e-01 -7.20041689e-02  2.61706509e-01
   6.30902880e-04  9.28641465e-02  2.69737363e-01  7.92748363e-02
   1.41175686e-01 -1.06515162e+00  2.36460774e-01  3.03824855e-01
  -1.06943181e-02]
 [-4.15739139e-01  5.30120432e-01 -5.61068593e-02  6.64976077e-02
   3.02477098e-02 -4.03191429e-01 -8.67481130e-01 -3.02229306e-02
  -3.60125846e-01  1.01788729e+00 -2.36083639e-01 -7.16366287e-01
   3.52824181e-05]]
```

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
print(lr2_10.intercept_)
print(lr2_1.intercept_)
print(lr2_0_1.intercept_)

print(lr2_10.coef_)
print(lr2_1.coef_)
print(lr2_0_1.coef_)
```

```
[-0.09116465  0.22505155 -0.1338869 ]
[-0.04189598  0.09954385 -0.05764787]
[-0.0309764   0.06088599 -0.02990959]
```

L2 규제, C=0.1, coef

```
[[-9.58124583e-02  6.20053775e-02  4.10703877e-02 -3.11476436e-01
  -2.36130063e-02  1.41964156e-01  2.70588460e-01 -2.50027000e-02
   9.18238829e-02  4.43004437e-04  2.06996901e-03  1.81604933e-01
   1.06189984e-02]
 [ 2.07351116e-01 -3.27651861e-01 -2.06190696e-02  2.00134007e-01
   1.70593903e-02  4.22076917e-02  1.50310439e-01  3.39526032e-02
   8.46143308e-02 -4.55087589e-01  1.03949518e-01  1.46428596e-01
  -8.41779799e-03]
 [-1.11538658e-01  2.65646483e-01 -2.04513180e-02  1.11342429e-01
   6.55361593e-03 -1.84171847e-01 -4.20898899e-01 -8.94990321e-03
  -1.76438214e-01  4.54644584e-01 -1.06019487e-01 -3.28033529e-01
  -2.20120039e-03]]
```

규제 강도가 클수록(c 값이 작을 수록) 추정된 계수의 절대값이 작아진다.

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
print(lr1_10.intercept_)
print(lr1_1.intercept_)
print(lr1_0_1.intercept_)

print(lr1_10.coef_)
print(lr1_1.coef_)
print(lr1_0_1.coef_)
```

```
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
```

L1 규제, C=10, coef

```
[[-1.00320141e+00  2.37545806e+00  1.39742788e-01 -1.84109789e+00
   8.22579810e-02  0.00000000e+00  7.09381179e+00  0.00000000e+00
  -2.96696478e+00 -8.11418226e-01  0.00000000e+00  0.00000000e+00
   3.46734711e-02]
 [ 1.32038068e+00 -2.73659990e+00 -3.36104137e+00  9.40629981e-01
  -5.04544304e-05 -1.95903666e+00  1.91336108e+00  1.23569355e+01
   2.60166228e+00 -3.25195851e+00  4.30443043e+00 -5.09057016e-01
  -2.74350580e-02]
 [-2.65394137e-01  1.39431959e+00  0.00000000e+00  3.26664966e-02
   1.16594625e-01  0.00000000e+00 -7.99401027e+00  0.00000000e+00
   0.00000000e+00  1.25062705e+00 -2.23535222e+00 -3.79473252e+00
  -5.56775509e-04]]
```

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
print(lr1_10.intercept_)
print(lr1_1.intercept_)
print(lr1_0_1.intercept_)

print(lr1_10.coef_)
print(lr1_1.coef_)
print(lr1_0_1.coef_)
```

```
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
```

L1 규제, C=1, coef

```
[[-2.49941553e-02  8.08647169e-02  0.00000000e+00 -7.05065667e-01
  -4.57903114e-02  0.00000000e+00  1.97017748e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   1.76091083e-02]
 [ 6.24899398e-01 -1.24676574e+00  0.00000000e+00  4.29381417e-01
   2.22527333e-02  0.00000000e+00  5.11055803e-01  0.00000000e+00
   1.50312466e-01 -1.72802110e+00  0.00000000e+00  0.00000000e+00
  -1.45159890e-02]
 [-1.76462410e-01  4.51974881e-01  0.00000000e+00  1.71620687e-02
   1.77907635e-02  0.00000000e+00 -3.17989658e+00  0.00000000e+00
   0.00000000e+00  9.13097220e-01  0.00000000e+00 -1.00807164e+00
   4.44542492e-04]]
```

EX2. WINE DATA

Enjoy your stylish business and campus life with BIZCAM

```
print(lr1_10.intercept_)
print(lr1_1.intercept_)
print(lr1_0_1.intercept_)

print(lr1_10.coef_)
print(lr1_1.coef_)
print(lr1_0_1.coef_)
```

```
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]
```

L1 규제, C=0.1, coef

```
[[ 0.          0.          0.          -0.47503216 -0.03842898  0.
   0.          0.          0.           0.          0.          0.
   0.01561511]
 [ 0.         -0.05603441  0.           0.18747928  0.05081683  0.
   0.          0.          0.          -0.85632169  0.          0.
  -0.00781363]
 [ 0.          0.          0.           0.          -0.0089824   0.
  -1.13620496  0.          0.           0.68642313  0.          0.
  -0.00244825]]
```

규제 강도가 클수록(c 값이 작을 수록) 추정된 계수가 0이 증가한다.

= L1 규제는 계수추정치가 0인 계수에 대응하는 특성변수를 제거하는 역할

3. 로버스트 회귀

Enjoy your stylish business and campus life with BIZCAM

기존 Linear Regression : 최소 제곱법을 이용하여 회귀 계수를 추정

그치만, 아웃라이어와 같이 잔차가 다른 데이터에 비해 매우 큰 경우, 제곱을 하면 그 값이 엄청 커지기 때문에 전체 추정치가 왜곡되기 쉽다.

따라서 로버스트 회귀는 이러한 문제를 완화하기 위한 회귀 모델 기법이다.

로버스트 회귀 : 잔차의 제곱 대신 절대값의 합이 최소가 되도록 계수를 추정!

Classical linear regression: $\underset{\beta}{\operatorname{argmin}} \sum (\varepsilon_i)^2$

Robust regression: $\underset{\beta}{\operatorname{argmin}} \sum |\varepsilon_i|$

4. Lidge, Lasso, ElasticNet

Enjoy your stylish business and campus life with BIZCAM

다중공산성이 있는 데이터는 선형 회귀 모형을 만들면 회귀 계수의 영향력이 과다 추정될 수 있다.

규제 방식을 이용한 회귀 모델링 기법이다.

Ridge : 회귀 계수의 제곱합을 더한다.

Lasso : 회귀 계수의 절대값을 더한다.

Elastic Net : Ridge 와 Lasso를 결합.

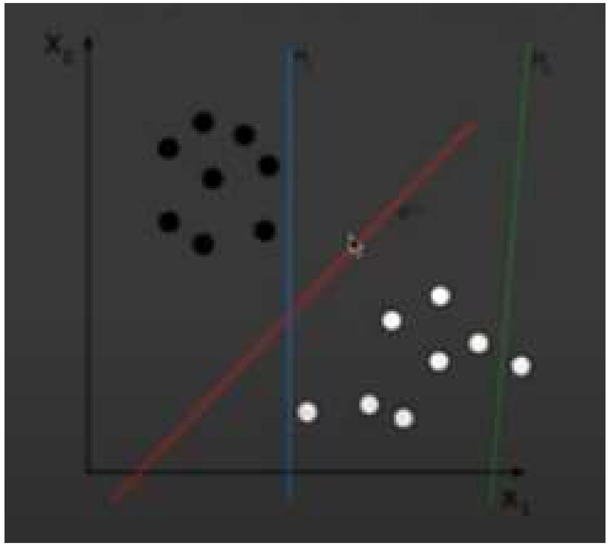
Classical linear regression: $\underset{\beta}{\operatorname{argmin}} \sum \varepsilon_i^2$

Ridge: $\underset{\beta}{\operatorname{argmin}} \sum \varepsilon_i^2 + \lambda \sum \beta_k^2$

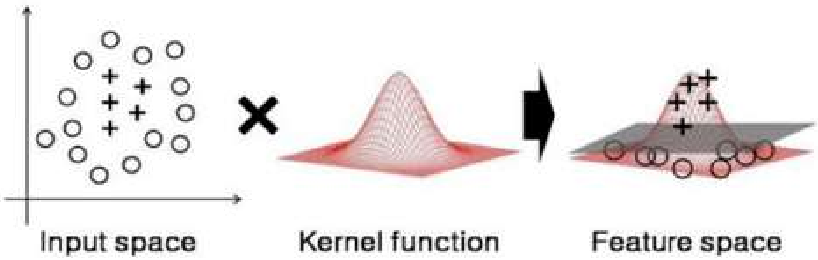
Lasso: $\underset{\beta}{\operatorname{argmin}} \sum \varepsilon_i^2 + \lambda \sum |\beta_k|$

Elastic net: $\underset{\beta}{\operatorname{argmin}} \sum \varepsilon_i^2 + \lambda_1 \sum \beta_k^2 + \lambda_2 \sum |\beta_k|$

5. SVM(Support Vector Machine) 회귀



- 회귀, 분류, 이상치 탐지 등에도 사용되는 지도 학습 방법이다.
- 클래스 사이의 경계에 위치한 데이터 포인트를 서포트 벡터라고 한다.
- 각 서포트 벡터가 클래스 사이의 결정 경계를 구분하는데 얼마나 중요한지를 학습한다.
- 각 서포트 벡터사이의 마진이 가장 큰 방향으로 학습한다.
- 서포트 벡터까지의 거리와 서포트 벡터의 중요도를 기반으로 예측을 수행한다.



Kernal SVR

커널 기법
입력 데이터를 고차원 공간에 사상해서 비선형 특징을 학습할 수 있도록 확장하는 방법

EX3. BOSTON 집값 DATA

1) 선형 회귀

```
# 미국 Boston 지역의 집값 data 불러오기
import pandas as pd
house = pd.read_csv('https://raw.githubusercontent.com/rasbt/'
                    'python-machine-learning-book-2nd-edition'
                    '/master/code/ch10/housing.data.txt',header=None,sep='₩s+')
house.columns = ['CRIM', 'ZN', 'INDUS', 'CHAS','NOX', 'RM', 'AGE', 'DIS', 'RAD','TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
house.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

EX3. BOSTON 집값 DATA

Enjoy your stylish business and campus life with BIZCAM

1) 선형 회귀

```
#일부 변수에 대한 log 변환
import numpy as np
house['LLSTAT']=np.log(house['LSTAT'])
house['LINDUS']=np.log(house['INDUS'])
house.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	LLSTAT	LINDUS
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0	1.605430	0.837248
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6	2.212660	1.955860
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7	1.393766	1.955860
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4	1.078410	0.779325
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2	1.673351	0.779325

EX3. BOSTON 집값 DATA

1) 선형 회귀

```
#전체 data를 종속변수 y와 특성변수 X의 data로 나누기
y = house['MEDV'].values
house1=house.drop(['LSTAT','INDUS','MEDV'],axis=1)
X = house1.values
```

```
#전체 data를 traning data(70%)와 test data(30%)로 나누기
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

EX3. BOSTON 집값 DATA

Enjoy your stylish business and campus life with BIZCAM

1) 선형 회귀

```
#regression module 불러오기와 모형추정
from sklearn.linear_model import LinearRegression
mlr = LinearRegression()
mlr.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#추정값 확인
print('Slope:' ,mlr.coef_)
print('Intercept:' ,mlr.intercept_)
```

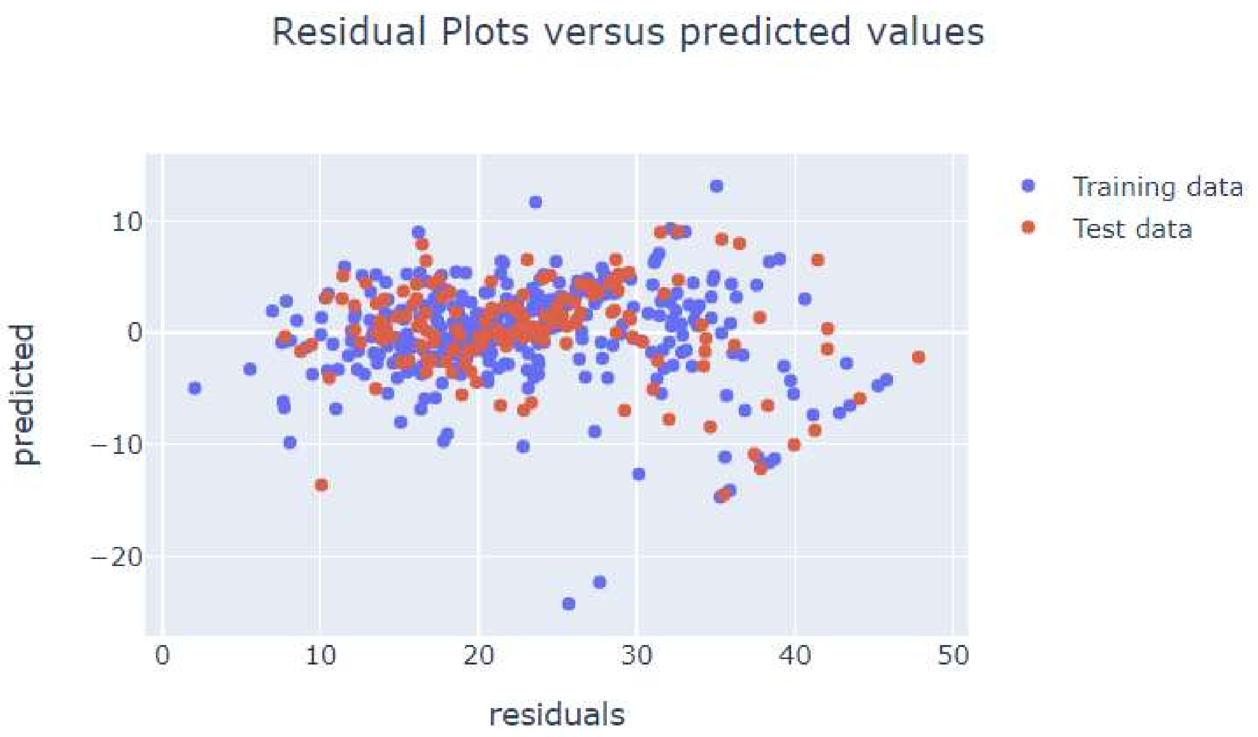
```
Slope: [-1.36676828e-01  3.13177997e-02  2.52393199e+00 -1.70295629e+01
  1.23977704e+00  3.06818458e-02 -1.28840466e+00  2.61968148e-01
 -6.58141653e-03 -8.27862485e-01  4.90558897e-03 -9.97211822e+00
 -6.04522425e-01]
Intercept: 65.69779117501716
```

```
#예측치 구하기
y_train_pred=mlr.predict(X_train)
y_test_pred=mlr.predict(X_test)
```

EX3. BOSTON 집값 DATA

Enjoy your stylish business and campus life with BIZCAM

1) 선형 회귀



TRAIN MAE : 18.139
TEST MAE : 17.416

TRAIN R^2 : 0.777
TEST R^2 : 0.810

EX3. BOSTON 집값 DATA

2) 로버스트 회귀

```
############## 로버스트 회귀 'RANSAC Regressor'
from sklearn.linear_model import RANSACRegressor  #RANSAC module import
rans = RANSACRegressor(max_trials=100,min_samples=45,loss='absolute_loss',residual_threshold=5.0, random_state=1)
rans.fit(X_train,y_train)
y_train_pred=rans.predict(X_train)
y_test_pred=rans.predict(X_test)
print(rans.estimator_.coef_)
print(rans.estimator_.intercept_)
```

```
[-0.33997467  0.02628575  1.26584351 -6.7774809   3.20459419 -0.03651634
 -1.13372086  0.24298439 -0.00712198 -0.59996172  0.01186441 -3.85624577
 -1.01175348]
31.427983601251846
```

EX3. BOSTON 집값 DATA

Enjoy your stylish business and campus life with BIZCAM

2) 로버스트 회귀

TRAIN MAE : 26.959
TEST MAE : 22.296

```
# RANSAC에서 회귀 계수를 추정하는데 사용한데이터(inner)와 특이치(outlier)출력
inlier_mask=rans.inlier_mask_
print('inner',inlier_mask)
```

```
inner [ True  True False  True False  True  True  True False False  True  True
  True  True  True  True  True  True  True False  True  True  True  True
  True  True False  True  True  True  True  True  True  True  True False
  True  True  True  True  True  True False  True False  True False False
  True False  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
```

```
outlier_mask=np.logical_not(inlier_mask)
print('outlier',outlier_mask)
```

```
outlier [False False  True False  True False False False  True  True False False
 False False False False False False False  True False False False False
 False False  True False False False False False False False False  True
 False False False False False False  True False  True False  True  True
 False  True False False False False False False False False False False
 False False False False False False False False False False False False
```

EX3. BOSTON 집값 DATA

Enjoy your stylish business and campus life with BIZCAM

2) SVR

```
###########SVR.Regression.....
from sklearn.svm import SVR  #SVR module import
svl=SVR(kernel='linear', C=1.0,epsilon=0.1)      #선형 SVM회귀
svr=SVR(kernel='rbf', C=1.0,epsilon=0.1)      #비선형 SVM회귀
svl.fit(X_train,y_train)   #선형 SVM model fitting
svr.fit(X_train,y_train)   #비선형 SVM model fitting
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

선형 SVR

TRAIN MSE : 22.270

TEST MSE : 16.382

R^2 : 0.7257

비선형 SVR

TRAIN MSE : 66.443

TEST MSE : 75.256

R^2 : 0.1816

Reference

https://kkokkilkon.tistory.com/19

https://hleecaster.com/ml-logistic-regression-concept/

https://hongl.tistory.com/133?category=933150

https://brunch.co.kr/@gimmesilver/38

https://hongl.tistory.com/133?category=933150

https://bskyvision.com/163