6

GLM

Sometimes life is going to hit you in the head with a brick. Don’t lose faith.

–Steve Jobs

개념

이산형 종속변수

전통적 회귀분석에서 목표변수 Y는 연속형 metric, continuous를 가정하고 확률분포함수는 정규분포를 가정한다. 그러므로 목표변수가 이산형인 경우 전통적 방법으로 추정은 불가능한다.

이산형 목표변수의 종류는 이진형, 이산형 정수, 순서형, 명목형으로 나뉜다.

- 이진형 : 성공 실패를 갖는 변수로 베르누이 분포를 따른다.
- 이산형 (양) 정수 : 교통사고 건수, 감염 환자 수 포아송 분포를 따른다.
- 순서형 : 순서 크기를 갖는 것으로 알파벳 학점, 리커트 척도 등이 여기에 속하며 다항 분포를 따른다.
- 명목형 : 거주지역, 직업 등과 같이 분류형 변수이며 다항 분포를 따른다.

LINK 함수

일반 회귀모형은 $E(y) = \mu = Xb$, (Xb 를 선형예측함수, linear predictor라 한다)이므로 μ가 가질 수 있는 값은 $(-\infty, \infty)$ 연속형이므로 Xb에 의해 설명가능하다. 그러나 이진형 $y \sim B(p)$ 의 경우 $E(y) = p$이므로 (0, 1) 값만을 갖는다. 그래서 다음과 같은 링크함수 g를 이용하여 $g(E(Y \mid x)) = g(\mu) = Xb$, $g(\mu)$가 가질 수 있는 값을 $(-\infty, \infty)$ 연속형으로 만들어 선형예측함수에 의해 예측한다.

- Y~정규분포 : 링크함수 g는 항등함수이다. $g(\mu) = \mu = Xb$
- Y~포아송분포 : $g(\mu) = ln(\mu) = Xb$, negative inverse
- Y~Gamma 분포 : $g(\mu) = -\frac{1}{\mu} = Xb$, log
- Y~베르누이 분포 : $g(\mu) = ln\frac{p}{1-p} = Xb$, logit

Common distributions with typical uses and canonical link functions

Distribution	Support of distribution	Typical uses	Link name	Link function, $\mathbf{X}\boldsymbol{\beta} = g(\mu)$
Normal	real: $(-\infty, +\infty)$	Linear-response data	Identity	$\mathbf{X}\boldsymbol{\beta} = \mu$
Exponential Gamma	real: $(0, +\infty)$	Exponential-response data, scale parameters	Negative inverse	$\mathbf{X}\boldsymbol{\beta} = -\mu^{-1}$
Inverse Gaussian	real: $(0, +\infty)$		Inverse squared	$\mathbf{X}\boldsymbol{\beta} = \mu^{-2}$
Poisson	integer: $0, 1, 2, \ldots$	count of occurrences in fixed amount of time/space	Log	$\mathbf{X}\boldsymbol{\beta} = \ln(\mu)$
Bernoulli	integer: $\{0, 1\}$	outcome of single yes/no occurrence	Logit	$\mathbf{X}\boldsymbol{\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$
Binomial	integer: $0, 1, \ldots, N$	count of # of "yes" occurrences out of N yes/no occurrences		
Categorical	integer: $[0, K)$	outcome of single K-way occurrence		
	K-vector of integer: $[0, 1]$, where exactly one element in the vector has the value 1			
Multinomial	*K*-vector of integer: $[0, N]$	count of occurrences of different types (1 .. *K*) out of *N* total *K*-way occurrences		

위키피디아

추정방법

모형 $g(E(y)) = Xb + e,\ e \sim N(0,\sigma^2)$

추정방법

- 이진형 분포 $P(y_i = 1) = p,\ p = e^{(-\theta x)} => g(p_x) = ln(p_x) = -\theta x$
- Poisson $E(y) = \lambda,\ \lambda = ne^{\theta}, g(\lambda_i) = ln(n_i) + \theta * i$

지수족 exponential family

- $f(y;\theta) = h(x)exp(\eta(\theta)T(x) - A(\theta))$로 표현되는 확률변수 y는 지수족이다. θ는 모수, $T(x)$는 충분통계량이다.
- (성질) 지수족의 T(x)는 완비통계량이며 T(x)의 함수 중 불편성을 만족하는 통계량이 MVUE이다.
- (성질) 로그우도함수 $ln(f(y;\theta))$는 다음 성질을 갖는다. (1) score 함수 $U = \frac{dln(f(y;\theta))}{d\theta}$의 기대값은 $E(U) = 0$ 이다. (2) U의 분산을 Information이라 정의한다. $V(U) = J$ (3) 스코어 함수의 θ 1차 미분의 기대값은 $E(\frac{dU}{d\theta}) = -V(U) = -J$ 관계식을 갖는다.
- 정규분포, 감마분포, 포아송분포, 이항분포, 베타분포 등 대부분의 유명한 분포는 지수족이다.

스코어 함수 $U=0$을 N-R 방식으로 풀면 θ의 MVUE 추정치를 얻는다. $\hat{\theta}^{(m)}=\hat{\theta}^{(m-1)}+\frac{U^{(m-1)}}{J^{(m-1)}}$

추정량의 샘플링분포 $\hat{\theta}\ \sim ?$

대표본 이론에 의해 [이차형식] $U'J^{-1}U\sim\chi^2(p)$, p=모수의 개수

MLE의 공분산 : $E[(\hat{\theta}-\theta)'(\hat{\theta}-\theta)]=J(\hat{\theta})$

Wald 검정통계량 $(\hat{\theta}-\theta)'J^{-1}((\hat{\theta})(\hat{\theta}-\theta)\sim\chi^2(p)$

LR 우도비 검정 $H_0:\theta=0$

$$\lambda=\frac{L(y;\hat{\theta}\ under\ H_0)}{L(y;\hat{\theta})}\sim 2ln\lambda\sim\chi^2(1)$$

Binary

모형

개별 관측치 z_i $B(\theta = \pi)$이므로 $i-$구간의 성공 회수는 $y_i \sim B(n, \pi)$

	Subgroups			
	1	2	...	N
Successes	Y_1	Y_2	...	Y_N
Failures	$n_1 - Y_1$	$n_2 - Y_2$	...	$n_N - Y_N$
Totals	n_1	n_2	...	n_N

그러므로 $\frac{y_i}{n_i} \sim b(\pi_i) => E(y_i/n_i) = \pi_i$

링크함수 $g(\pi_i) = Xb$? 회귀모형 적용을 위해서는 링크함수의 범위는 반드시 $(-\infty, \infty)$이어야 한다. 그러므로 가장 간단한 링크함수 $\pi_i = Xb$는 적절하지 않음

링크함수

1) Probit 모형 : $\Phi^{-1}(\pi) = Xb$, Φ^{-1} : 역정규분포함수

- 중앙값, 백분위 => 개체를 50% 감소시킬 수 있는 dose 양

2) Logit 함수 : $ln(\frac{\pi}{1-\pi}) = Xb$,

- 회귀계수 b_k의 부호는 개체의 성공에 미치는 예측변수(X_k) 영향 부호와 동일함
- e^{b_k}는 예측변수 X_k 오즈비

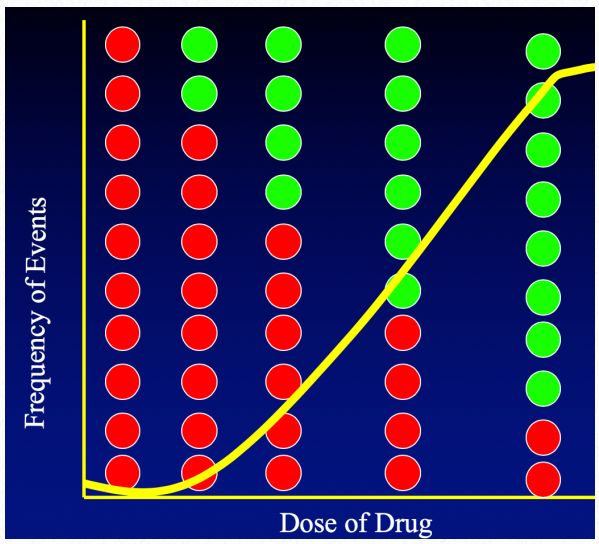
3) Complementary log-log function (Gumpertz) : $log(-log(1-\pi)) = Xb$

추정

1절에서 설명하였듯이 모수 b의 추정량은 MLE 방법, N_R 방법으로 추정한다.

종속변수의 관측치는 (0, 1)의 이진형 값이나 추정 회귀계수에 의해 계산된 적합치는 (0, 1) 사이의 확률 값이다. 다음 예제는 약물 사용량에 따라 무당벌레 죽는 여부를 측정한 자료이다. 65마리 무당벌레에 1.69 용량을 살포했을 때 6마리는 죽고 나머지 59마리는 생존하였다.

Dose, x_i ($\log_{10}CS_2mgl^{-1}$)	Number of beetles, n_i	Number killed, y_i
1.6907	59	6
1.7242	60	13
1.7552	62	18
1.7842	56	28
1.8113	63	52
1.8369	59	53
1.8610	62	61
1.8839	60	60



모형 : $log(\frac{\pi_i}{1-\pi_i}) = a + b(Dose) + e,\ \pi_i = P(Success|Dose = x_i)$

오즈비 : $\frac{\pi}{1-\pi} = exp(a + b(Dose))$, Dose=X 1단위 증가하면 오즈비는 e^b만큼 증가한다.

모수 모형 : $\pi = \frac{1}{1 - exp(a + b(Dose))}$ - 회귀계수의 부호가 양수이고 값이 커지면 (성공: π_i , event)가 커지므로 성공 확률이 높아지고 부호가 음수=> 절대값이 커지면 π_i 가 작아지므로 성공 확률이 낮아진다.

모형평가

Confussion 행렬

정분류표를 활용 :

실제. 예측->	성공	실패
성공	TP	FP
실패	FN	TN

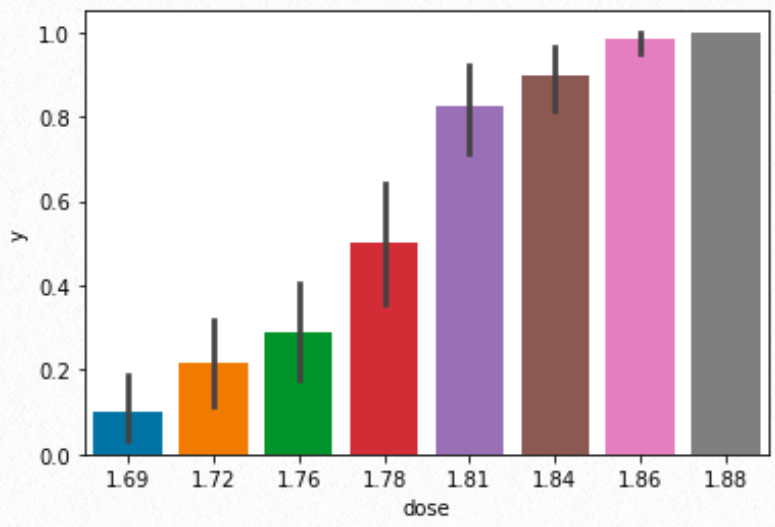
정분류 비율 : (TP+TN)/(TP+FP+FN+TN) 높은 모형이 최적 모형임

민감도 sensitivity : (TP)/(TP+FP) / 특이도 specificity : (TN)/(FN+TN)

예제 데이터 분석 (빈도표가 주어진 경우)

```
import pandas as pd;import numpy as np
dose=pd.Series([1.69,1.72,1.76,1.78,1.81,1.84,1.86,1.88])
n=pd.Series([59,60,62,56,63,59,62,60]); y=pd.Series([6,13,18,28,52,53,61,60])
a=pd.Series(np.repeat(dose,n-y)) ; a2=pd.Series(np.repeat(dose,y))
a.reset_index(drop=True,inplace=True); a2.reset_index(drop=True,inplace=True)
b=pd.Series(np.repeat(0,sum(n-y))); b2=pd.Series(np.repeat(1,sum(y)))
b.reset_index(drop=True,inplace=True); b2.reset_index(drop=True,inplace=True)
ab=pd.concat([a,b], axis=1) ; ab.columns=['dose','y']
ab2=pd.concat([a2,b2], axis=1); ab2.columns=['dose','y']
df=pd.concat([ab,ab2],axis=0)
```

```
import seaborn as sns
sns.barplot(x=df['dose'].astype(str),y='y',data=df)
```



```
df['int'] = 1
import statsmodels.api as sm
logit_model=sm.Logit(df['y'],df[['int','dose']])
result=logit_model.fit()
print(result.summary2())
```

추정식 : $log(\frac{p}{1-p}) = -60.1 + 33.93 * dose$

```
                          Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.419
Dependent Variable: y                AIC:              378.8720
Date:               2020-06-15 08:43 BIC:              387.2237
No. Observations:   481              Log-Likelihood:   -187.44
Df Model:           1                LL-Null:          -322.72
Df Residuals:       479              LLR p-value:      8.5287e-61
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
----------------------------------------------------------------
            Coef.    Std.Err.      z      P>|z|    [0.025    0.975]
----------------------------------------------------------------
int        -60.1033    5.1642  -11.6385  0.0000  -70.2249  -49.9817
dose        33.9342    2.9029   11.6898  0.0000   28.2446   39.6237
================================================================
```

dose 회귀계수 부호가 +이므로 벌레 죽음에 양의 영향을 미치므로 dose 사용량이 증가하면 벌레 죽일 가능성이 높아진다.

```
[ ]  1  np.exp(result.params)
```

```
int     7.897277e-27
dose    5.462866e+14
```

사용량의 오즈비는 5.46이므로 사용량을 한단위 증가할 때마다 벌레 죽을 확률은 5.4배 증가한다고 해석하면 된다.

타이타닉 사례

데이터 가져오기

```
import pandas as pd
df=pd.read_csv('http://wolfpack.hnu.ac.kr/Stat_Notes/example_data/titanic_data.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   survived   1309 non-null   float64
 2   name       1309 non-null   object
 3   sex        1309 non-null   object
 4   age        1046 non-null   float64
 5   sibsp      1309 non-null   float64
 6   parch      1309 non-null   float64
 7   ticket     1309 non-null   object
 8   fare       1308 non-null   float64
 9   cabin      295 non-null    object
 10  embarked   1307 non-null   object
 11  boat       486 non-null    object
 12  body       121 non-null    float64
 13  home.dest  745 non-null    object
```

가능하면 예측변수(survived)를 (0,1) 숫자로 넣는 것이 함수 활용하기 편리하다. 그리고 예측변수와 목표변수를 선택하고 Nan(결측치)을 제거한다.

```
df0=df[['survived','age','sibsp','fare','sex','pclass']]
df0.dropna(inplace=True); df0.shape
```

데이터 차원 (1045,6)

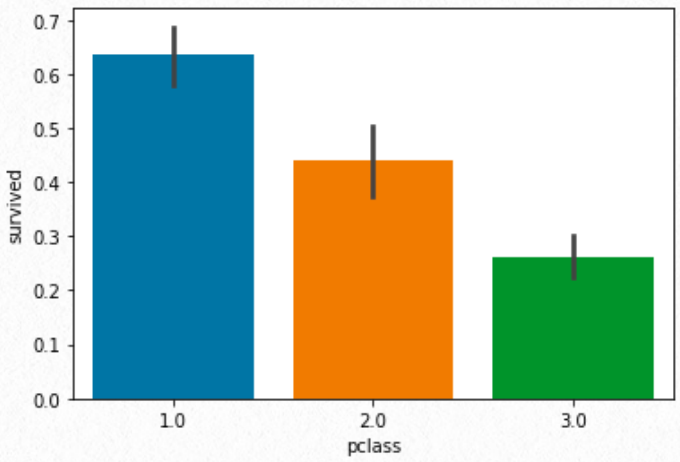
범주형변수 활용 생존율 보기

```
[ ]  1  df0.groupby(['pclass','sex']).mean()['survived']
```

```
pclass  sex
1.0     female    0.962406
        male      0.350993
2.0     female    0.893204
        male      0.145570
3.0     female    0.473684
        male      0.169540
```

(여성, 1등석) 생존율 = 96%

그래프 표현




```
import seaborn as sns
sns.barplot(x='pclass', y='survived', data=df0)
```

로지스틱 회귀분석 추정

```
genders = {"male": 0, "female": 1}
df0['sex']=df0['sex'].map(genders)
df0['int']=1; y=df0['survived']; X=df0[['int','age','pclass','fare','sex']]
import statsmodels.api as sm
logit_model=sm.Logit(y,X); result=logit_model.fit()
print(result.summary2())
```

모형에 삽입되는 예측변수는 반드시 숫자형으로 변환되어야 하며, 수준이 3개 이상인 경우에는 더미 변수를 (수준 수-1)만큼 만들어야 한다. 가능하면 이진형 예측변수만 사용하자.

```
         Iterations 6
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.305
Dependent Variable: survived         AIC:              992.8071
Date:               2020-06-15 09:14 BIC:              1017.5660
No. Observations:   1045             Log-Likelihood:   -491.40
Df Model:           4                LL-Null:          -706.79
Df Residuals:       1040             LLR p-value:      6.2537e-92
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
-----------------------------------------------------------------
            Coef.    Std.Err.      z      P>|z|     [0.025    0.975]
-----------------------------------------------------------------
int          2.0110     0.4235    4.7486   0.0000    1.1810    2.8410
age         -0.0337     0.0063   -5.3489   0.0000   -0.0460   -0.0213
pclass      -1.1080     0.1285   -8.6222   0.0000   -1.3598   -0.8561
fare         0.0007     0.0017    0.3836   0.7013   -0.0027    0.0040
sex          2.4900     0.1670   14.9111   0.0000    2.1627    2.8173
=================================================================
```

나이, 좌석 등급이 올라갈수록 생존 가능성 낮아지고(회귀계수 음) 요금 올라갈수록 여성이면 생존 가능성 높아진다.

오즈비

```
1  import numpy as np
2  np.exp(result.params)
```

```
int         7.470678
age         0.966874
pclass      0.330224
fare        1.000662
sex        12.061147
```

sex 여성=이므로 남성에 비해 생존 가능성은 12배 높다. 나이가 올라갈수록 생존 가능성은 낮아지며(회귀계수 음수) 1/0.97=1.03배 사망 가능성이 높아진다.

정분류 비율

정분류 비율 (생존-> 생존 76%=(300)/(127+300), 사망->사망 80%=(523)/523+95) 78.8%이다.

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X,y)
y_pred=logreg.predict(X)
round(logreg.score(X,y) * 100, 2)
```

```
78.76
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y, y_pred)
print(confusion_matrix)
```

```
[[523  95]
 [127 300]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.80      0.85      0.82       618
         1.0       0.76      0.70      0.73       427

    accuracy                           0.79      1045
   macro avg       0.78      0.77      0.78      1045
weighted avg       0.79      0.79      0.79      1045
```

사후확률

사후확률

```
[ ]   1   logreg.predict_proba(X)
```

```
array([[0.07504328, 0.92495672],
       [0.27433915, 0.72566085],
       [0.0337357 , 0.9662643 ],
       ...,
       [0.89600832, 0.10399168],
       [0.89754085, 0.10245915],
       [0.90342425, 0.09657575]])
```

각 승객의 생존확률이 출력된다. 0.5 이상이면 생존으로, 미만이면 사망으로 분류된다. 잠정 threshold (cut-off)는 0.5이다.

ROC 커브

```
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y, logreg.predict(X))
fpr, tpr, thresholds = roc_curve(y, logreg.predict_proba(X)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```

- y축은 민감도, x-축은 (1-특이도)
- 45도 선(기준선)은 민감도=(1--특이도)가 되는 선으로 랜덤(검사 키트 효용 제로) 검사를 의미함
- AUC(area under curve) excellent 검사 90~100%, Good 80~90%, 70%이상이면 fair

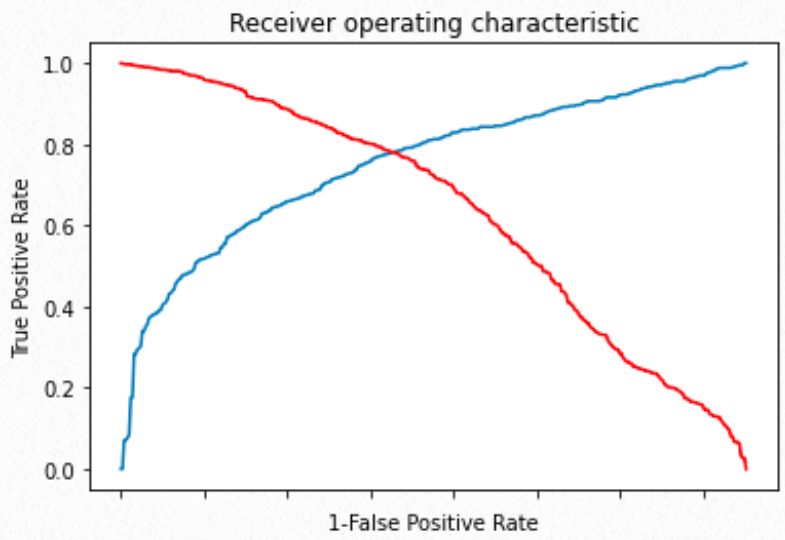
optimal cut-off 값 : 사후확률이 0.42 이상이면 사망, 미만이면 사망으로 분류하는 것이 민감도, 특이도 가장 높은 기준값이다. Youden's J-Score 방법은 붉은선으로부터 가장 멀리 떨어진 값을 최적 기준값으로 제안한다.

```
i = np.arange(len(tpr)) # index for df
roc = pd.DataFrame({'fpr' : pd.Series(fpr, index=i),'tpr' : pd.Series(tpr, index = i), '1-fpr' : pd.Series(1-fpr, index = i), 'tf' : pd.Series(tpr - (1-fpr), index = i), 'thresholds' : pd.Series(thresholds, index = i)})
roc.iloc[(roc.tf-0).abs().argsort()[:1]]
import pylab as pl
# Plot tpr vs 1-fpr
fig, ax = pl.subplots()
pl.plot(roc['tpr'])
pl.plot(roc['1-fpr'], color = 'red')
pl.xlabel('1-False Positive Rate')
pl.ylabel('True Positive Rate')
pl.title('Receiver operating characteristic')
ax.set_xticklabels([])
```

Youden's J-Score

```
[26]  1  cutoff_youdens_j(fpr,tpr,thresholds
```

```
0.42373518300781265
```



Recursive Feature Elimination in Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
y=df0['survived']
X=df0[['age','pclass','fare','sex']]
# Build a logreg and compute the feature importances
model = LogisticRegression()
# create the RFE model and select k=2 attributes
rfe = RFE(model, 2)
rfe = rfe.fit(X, y)
# summarize the selection of the attributes
print('Selected features: %s' % list(X.columns[rfe.support_]))
```

```
Selected features: ['pclass', 'sex']
```

가장 유의한 2개 에측변수 선택 결과

다른 빅데이터 방법 적용

Stochastic Gradient Descent (SGD)

```
from sklearn import linear_model
sgd = linear_model.SGDClassifier(max_iter=5, tol=None)
sgd.fit(X,y)
Y_pred=sgd.predict(X)
round(sgd.score(X,y) * 100, 2)
```

정분류 비율 매번 변동 = ????%

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X,y)
Y_prediction = random_forest.predict(X)
round(random_forest.score(X,y) * 100, 2)
```

정분류 비율 = 97.7%

Linear Support Vector Machine

```
from sklearn.svm import SVC, LinearSVC
linear_svc = LinearSVC()
linear_svc.fit(X,y)
Y_pred = linear_svc.predict(X)
round(linear_svc.score(X,y) * 100, 2)
```

정분류 비율 매번 변동 = ????%

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X,y)
Y_pred = decision_tree.predict(X)
round(decision_tree.score(X,y) * 100, 2)
```

정분류 비율 = 97.7%

K Nearest Neighbor

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X,y)
Y_pred = knn.predict(X)
round(knn.score(X,y) * 100, 2)
```

정분류 비율 = 81.2%