

로지스틱 회귀와 분류 기초

분류 문제의 확률적 접근과 계수 해석

선형회귀를 알고 있다면 바로 따라올 수 있습니다.
이진 분류와 다중 분류를 모두 다룹니다.

1. 들어가며: 왜 선형회귀로는 부족한가

분류 문제의 목표는 입력 데이터가 어느 클래스에 속하는지 예측하는 것입니다. 가장 먼저 떠오르는 방법은 이미 알고 있는 선형회귀를 그대로 쓰는 것일 수 있습니다. 클래스를 0 과 1 로 수치화하고, 선형회귀로 0~1 사이 값을 예측하면 되지 않을까? 이 아이디어는 작동하는 것처럼 보이지만 두 가지 근본적인 문제를 안고 있습니다.

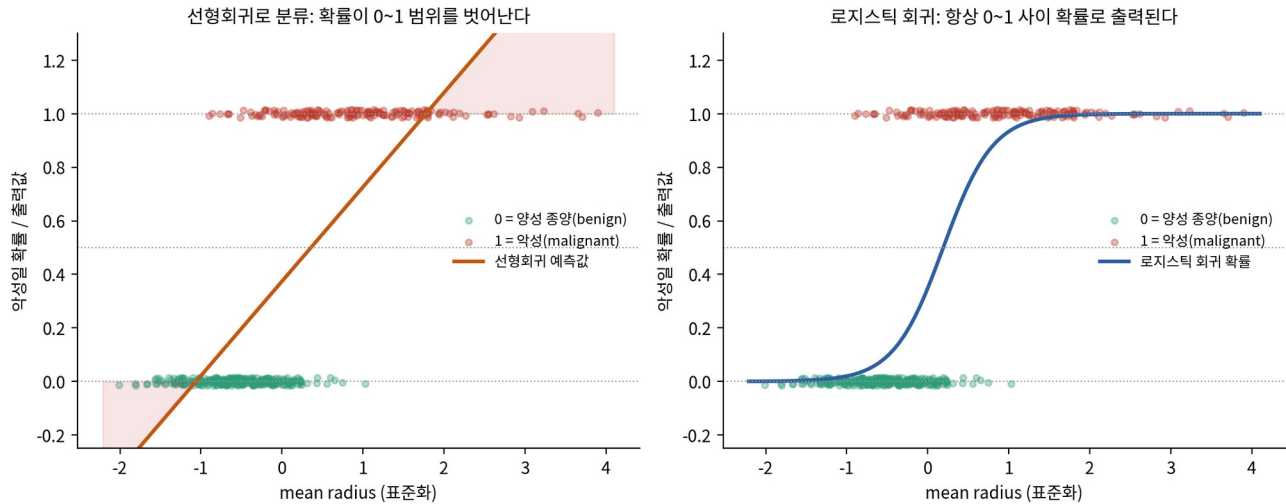


그림 1. 같은 데이터에 선형회귀(왼쪽)와 로지스틱 회귀(오른쪽)를 적용한 비교. 선형회귀는 0 보다 작거나 1 보다 큰 값을 출력해 확률로 해석할 수 없다. 로지스틱 회귀는 항상 0~1 범위의 값을 출력한다.

첫째, 선형회귀의 출력이 0~1 범위를 벗어납니다. 그림 1의 왼쪽에서 보듯, 선형회귀는 "악성 종양일 확률"로 1.3 혹은 -0.2 같은 의미 없는 값을 출력합니다. 확률이 0~1 사이여야 한다는 기본 조건을 위반합니다.

둘째, 이상치에 취약합니다. x 축 오른쪽 끝에 악성 종양 샘플이 하나 더 추가되면 회귀선 전체가 기울어져 중간 영역의 결정 경계가 바뀌어 버립니다.

로지스틱 회귀는 이 두 문제를 시그모이드 함수라는 수학적 장치로 해결합니다. 선형 결합(z)을 시그모이드 함수로 변환해 항상 0~1 사이의 확률로 출력합니다.

예제 데이터셋

이 가이드는 두 가지 데이터셋을 사용합니다.

- 이진 분류: Breast Cancer Wisconsin 데이터셋(sklearn 내장) — 유방암 조직 검사 결과로 악성(1)/양성 종양(0) 분류. sklearn 원본은 반대로(양성 종양=1) 코딩되어 있어, 교차검증과 모델 평가 가이드와 마찬가지로 $y = 1 - \text{target}$ 으로 뒤집어 사용합니다. 이하 "positive 클래스"는 1(=악성)을, "양성 종양"은 benign 을 뜻합니다.
- 다중 분류: Wine 데이터셋(sklearn 내장) — 화학 성분 분석으로 3 종류 와인 분류. 4 장(다중 분류)에서 사용.

2. 로지스틱 회귀의 원리

2-1. 시그모이드 함수

로지스틱 회귀는 선형 결합 z 를 계산한 뒤, 시그모이드 함수를 통해 0~1 확률로 변환합니다.

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p$$

$$P(y=1 | X) = \sigma(z) = 1 / (1 + \exp(-z))$$

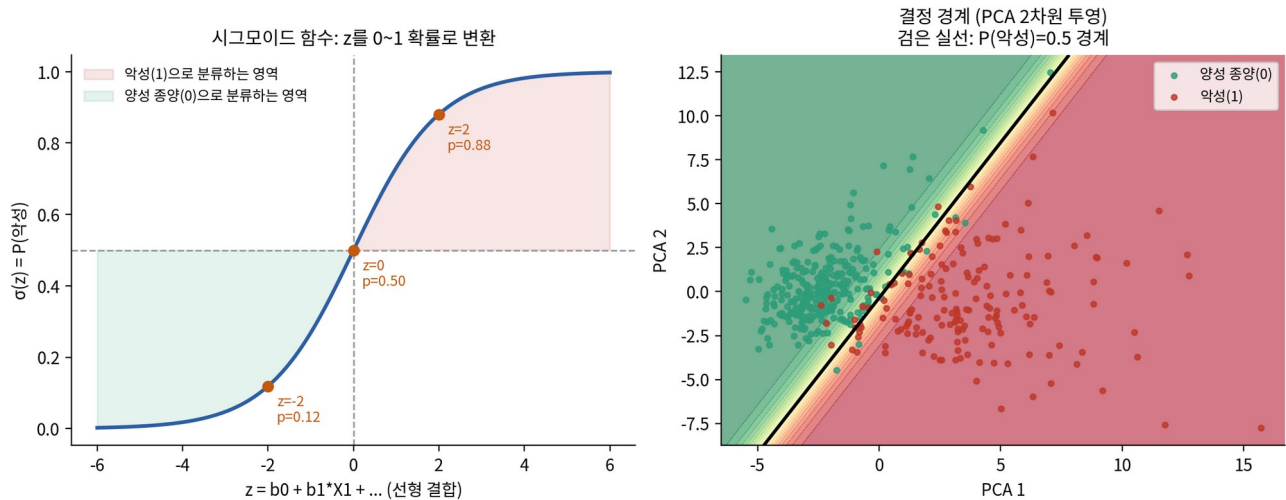


그림 2. 시그모이드 함수(왼쪽): z 가 0 일 때 확률은 정확히 0.5, z 가 양의 방향으로 커질수록 1에 가까워지고 음의 방향으로 작아질수록 0에 가까워진다. 오른쪽: Breast Cancer 데이터를 PCA로 2차원에 투영한 결정 경계. 검은 실선($P(\text{악성})=0.5$)을 기준으로 오른쪽(빨강)은 악성, 왼쪽(초록)은 양성 종양으로 분류된다.

결정 경계는 $\sigma(z) = 0.5$ 인 지점, 즉 $z = 0$ 인 선입니다. $z > 0$ 이면 악성(1, positive 클래스), $z < 0$ 이면 양성 종양(0)으로 분류합니다. 기본 임계값은 0.5이지만, 상황에 따라(예: 암 진단에서 민감도를 높이려면) 더 낮은 값(예: 0.3)으로 조정할 수 있습니다.

2-2. 손실함수: 왜 RSS 대신 Cross-Entropy를 쓰는가

선형회귀는 RSS(잔차제곱합)를 최소화합니다. 분류에 RSS를 그대로 쓰면 어떨까요? 확률값이 0~1로 제약되는 시그모이드 함수와 결합하면 RSS는 볼록(convex)하지 않은 목적함수를 만들어, 경사하강법이 전역 최솟값(global minimum)이 아닌 지역 최솟값(local minimum)에 갇힐 수 있습니다.

대신 로지스틱 회귀는 Cross-Entropy(이진 교차 엔트로피) 손실함수를 사용합니다.

$$\text{Loss} = -\sum [y_i \log(\hat{p}_i) + (1-y_i) \log(1-\hat{p}_i)]$$

이 손실함수의 직관은 간단합니다. 실제 클래스가 1인데 예측 확률($\hat{p}$)이 1에 가까우면 손실이 작고($\log(1) \approx 0$), 0에 가까우면 손실이 매우 크게($-\log(\epsilon) \rightarrow \infty$) 부과됩니다. 이 형태는 시그모이드와 결합했을 때 볼록 함수가 되어 경사하강법이 안정적으로 수렴합니다.

Cross-Entropy vs RSS 요약

RSS: 예측값과 실제값의 거리(차이)를 최소화 → 회귀에 적합

Cross-Entropy: 예측 확률과 실제 클래스의 불일치를 최소화 → 분류에 적합

분류 문제에 RSS 를 쓰면 수학적으로 불룩하지 않아 최적화가 불안정해집니다.

3. 계수 해석

3-1. 로그 오즈(log-odds)와 오즈비(odds ratio)

로지스틱 회귀의 계수를 해석하려면 오즈(odds)라는 개념이 필요합니다. 오즈는 사건이 일어날 확률(p)과 일어나지 않을 확률(1-p)의 비율입니다.

$$\text{오즈} = p / (1-p) \quad \text{로그 오즈} = \log(p / (1-p)) = z = \beta_0 + \beta_1 X_1 + \dots$$

로지스틱 회귀는 실제로 "로그 오즈"를 선형 결합으로 모델링하는 것입니다. 그래서 계수 β_j 의 의미는 다음과 같습니다. X_j 가 1 증가할 때 로그 오즈가 β_j 만큼 증가합니다. 이를 지수화하면 오즈비(odds ratio) = $\exp(\beta_j)$ 가 됩니다.

오즈비 해석 방법

오즈비 > 1: 해당 변수가 증가할수록 악성(1) 확률이 증가

오즈비 = 1: 해당 변수는 분류에 영향 없음

오즈비 < 1: 해당 변수가 증가할수록 악성(1) 확률이 감소

예: worst texture의 오즈비 = 4.20 이면, 다른 변수가 고정된 상태에서 이 변수가 1 표준편차 증가할 때 악성 오즈가 4.20 배가 됩니다

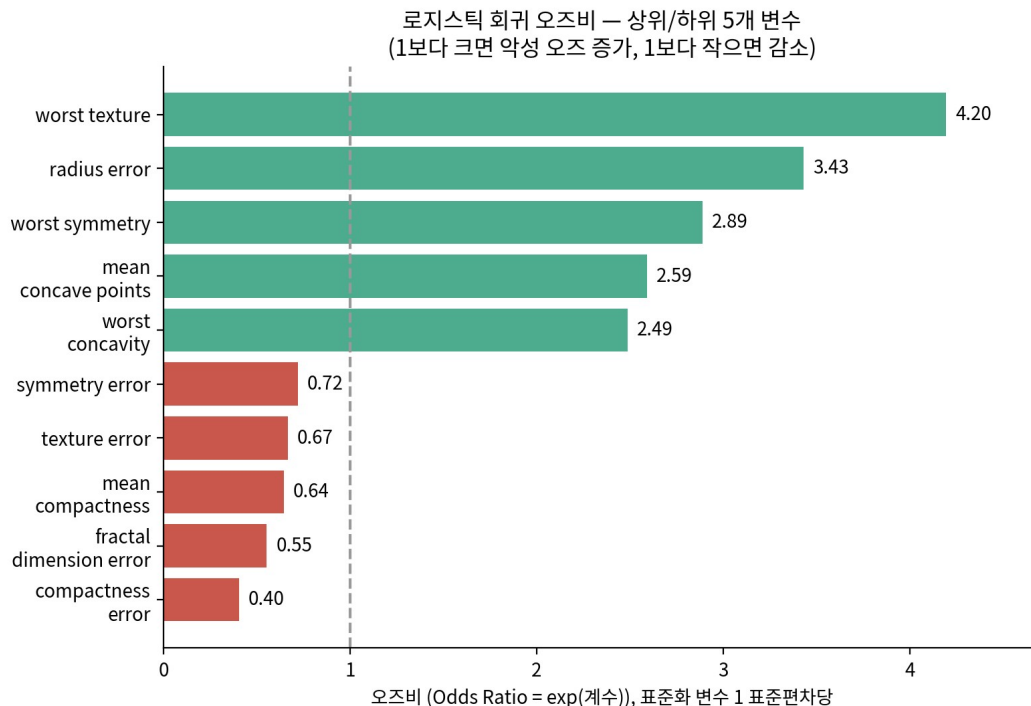


그림 3. Breast Cancer 데이터 로지스틱 회귀의 오즈비(상위/하위 5개 변수). 1보다 큰(초록) 변수는 악성 가능성을 높이고, 1보다 작은(빨강) 변수는 악성 가능성을 낮춘다. 값은 표준화된 변수가 1 표준편차 증가할 때의 오즈 배수다. 주의: 오즈비는 다른 변수들을 통제된 상태에서 하나의 변수만 변화했을 때의 효과이므로, 다중공선성이 있으면 해석에 신중해야 한다.

3-2. 규제가 있는 로지스틱 회귀

로지스틱 회귀도 변수가 많거나 다중공선성이 있을 때 계수가 불안정해질 수 있습니다. 이를 방지하기 위해 Ridge(L2), Lasso(L1) 규제를 그대로 적용할 수 있습니다. scikit-learn의 LogisticRegression은 기본적으로 L2 규제를 포함하고 있으며, 규제 강도는 $C = 1/\lambda$ 파라미터로 조절합니다(C 가 작을수록 규제가 강합니다).

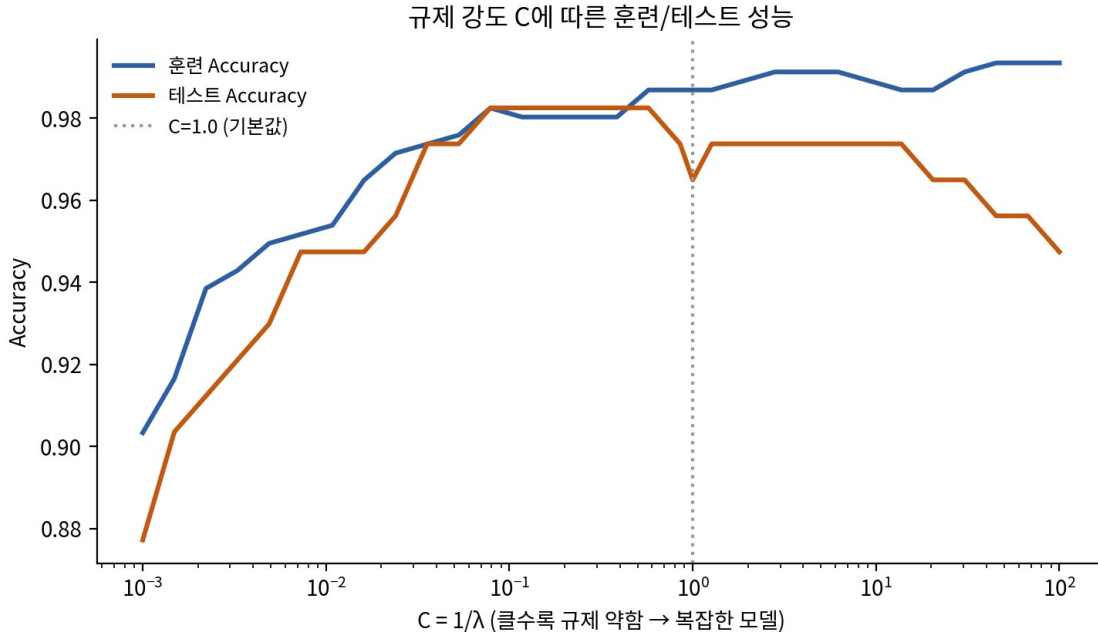


그림 4. 규제 강도 C 에 따른 훈련/테스트 Accuracy. C 가 매우 작으면(강한 규제) 과소적합, 매우 크면(약한 규제) 훈련 성능만 오르고 테스트 성능은 떨어진다. 이 분할에서는 기본값 $C=1.0$ (테스트 0.965)보다 조금 강한 규제($C \approx 0.08 \sim 0.57$, 테스트 0.982)에서 테스트 성능이 가장 높다. 다만 테스트셋이 114 개뿐이라 1~2 개 차이로 순위가 바뀌므로, 실무에서는 C 를 교차검증으로 정해야 한다.

규제의 원리(Ridge vs Lasso, 편향-분산 트레이드오프)는 규제 선형회귀 가이드에서 자세히 다루었습니다. 로지스틱 회귀에서도 동일한 원리가 적용됩니다.

R

```
library(glmnet)

# glmnet으로 로지스틱 회귀 + 규제
x <- scale(as.matrix(X_train)); y <- y_train
cv_fit <- cv.glmnet(x, y, family='binomial', alpha=0, nfolds=5) # L2
best_lambda <- cv_fit$lambda.min
coef(cv_fit, s='lambda.min')
```

Python

```
from sklearn.linear_model import LogisticRegression

# C = 1/λ. penalty='l1'로 Lasso도 가능
lr = LogisticRegression(C=1.0, penalty='l2', max_iter=1000)
lr.fit(X_train_s, y_train)
print(lr.coef_) # 계수
print(lr.intercept_) # 절편
```


4. 다중 분류

4-1. One-vs-Rest (OvR) 전략

로지스틱 회귀는 본래 이진 분류기이지만, 클래스가 3 개 이상인 경우에도 사용할 수 있습니다. 가장 직관적인 방법은 OvR(One-vs-Rest) 전략입니다. 클래스가 k 개이면 k 개의 이진 분류기를 훈련시킵니다. 각 분류기는 "나머지 모든 클래스 vs 이 클래스"를 구별합니다. 예측 시에는 k 개 분류기의 확률 출력 중 가장 높은 클래스를 선택합니다.

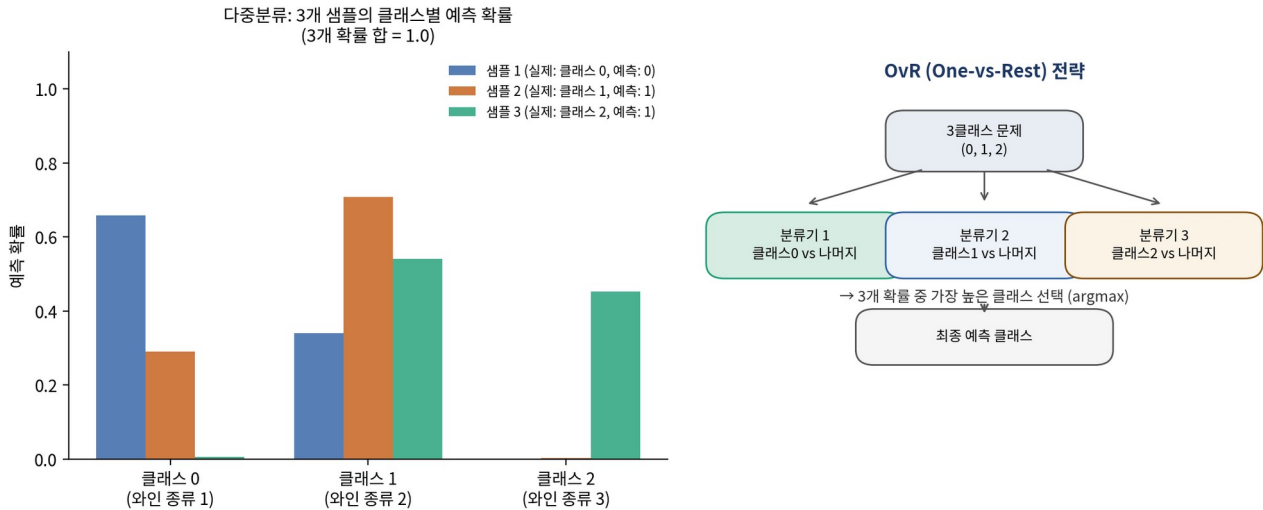


그림 5. OvR 전략 개념도(오른쪽)와 3 개 샘플의 클래스별 예측 확률(왼쪽). 샘플 3(실제 클래스 2)은 클래스 1 과 2 사이에서 확률이 비슷해 불확실한 경우로, 클래스 1 로 오분류됐다.

4-2. Softmax 로의 확장

OvR 의 대안은 다항 로지스틱 회귀(Multinomial Logistic Regression)입니다. k 개의 이진 분류기를 독립적으로 훈련하는 대신, k 개 클래스의 확률을 동시에 모델링하는 Softmax 함수를 사용합니다.

$$P(y=k \mid X) = \exp(z_k) / \sum_j \exp(z_j)$$

Softmax 는 k 개의 출력 확률의 합이 항상 1 이 되도록 정규화합니다. OvR 보다 이론적으로 더 일관성 있는 확률을 출력하며, sklearn 의 LogisticRegression 은 기본적으로 다중 분류 시 Softmax(lbfgs 솔버 기준)를 사용합니다.

	OvR (One-vs-Rest)	Softmax (Multinomial)
분류기 수	k 개 (독립 훈련)	1 개 (동시 훈련)
확률 합산	k 개 합이 반드시 1 이 아님	k 개 합 = 1 (보장)
속도	클래스 수 증가 시 선택 증가	단일 최적화, 일반적으로 빠름
적합한 상황	클래스가 매우 많을 때	클래스가 상호 배타적일 때

R

```
library(nnet)
```

```
# nnet 으로 다항 로지스틱 회귀
wine_df <- data.frame(X_train_s, target=factor(y_train))
lr_multi <- multinom(target ~ ., data=wine_df, maxit=1000)
pred <- predict(lr_multi, data.frame(X_test_s))
mean(pred == y_test) # Accuracy
```

Python

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# sklearn은 자동으로 다중분류 전략을 선택
lr_multi = LogisticRegression(max_iter=1000, random_state=42)
lr_multi.fit(X_train_s, y_train)

print('Accuracy:', accuracy_score(y_test, lr_multi.predict(X_test_s)))
# predict_proba: (샘플 수, 클래스 수) 형태의 확률 행렬
print('확률 형태:', lr_multi.predict_proba(X_test_s).shape)
```

5. 이진 분류 전체 파이프라인

Breast Cancer 데이터에 로지스틱 회귀를 적용하는 전체 흐름을 하나의 코드로 정리합니다.

R

```
library(pROC) # glm()은 R 기본 함수

# 데이터 준비
# X_train, X_test, y_train, y_test: Python과 같은 분할 (y: 악성=1)
xs <- scale(X_train) # 훈련 데이터로만 평균·표준편차 계산
df_train <- data.frame(xs, target=y_train)
xt <- scale(X_test, center=attr(xs,'scaled:center'),
scale=attr(xs,'scaled:scale'))
df_test <- data.frame(xt, target=y_test) # 테스트는 훈련 기준으로 변환만

# 모델 훈련
lr_fit <- glm(target ~ ., data=df_train, family=binomial)

# 예측 및 평가
prob <- predict(lr_fit, df_test, type='response')
pred <- ifelse(prob > 0.5, 1, 0)
cat('Accuracy:', mean(pred == df_test$target), ' ')
roc_obj <- roc(df_test$target, prob)
cat('AUC:', auc(roc_obj), ' ')
```

Python

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score

bc = load_breast_cancer(as_frame=True)
X, y = bc.data, 1 - bc.target # 악성=1로 뒤집음
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test) # fit 없이 transform만

lr = LogisticRegression(C=1.0, max_iter=1000, random_state=42)
lr.fit(X_train_s, y_train)

prob = lr.predict_proba(X_test_s)[:, 1]
pred = lr.predict(X_test_s)
print('Accuracy:', accuracy_score(y_test, pred))
print('AUC:', roc_auc_score(y_test, prob))
```

이 가이드는 로지스틱 회귀의 원리(시그모이드, Cross-Entropy), 계수 해석(오즈비), 규제, 다중 분류(OvR/Softmax)를 다루었습니다. SVM(서포트 벡터 머신), 나이브 베이즈 같은 다른 분류 알고리즘과의 비교, 다중 분류에서의 macro/micro 평균 지표 등은 후속 자료에서 다룰 수 있는 주제입니다.