

AI 보안 기술개발 교육

# 머신러닝(Machine Learning) 모델 2

AI 보안 기술개발 교육

# 머신러닝 모델 2

1. 로지스틱회귀 모델
2. K-근접이웃 모델
3. 나이브베이지스 모델
4. K-평균 모델

# 로지스틱 회귀(Logistic Regression)

- 로지스틱 회귀(Logistic Regression)

회귀(Regression)를 사용하여 데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측  
확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류해 주는 지도 학습 알고리즘

메일에 대해 스팸일 확률이 0.5 이상이면 spam으로 분류하고, 확률이 0.5보다 작은 경우 ham으로 분류  
데이터가 2개의 범주 중 하나에 속하도록 결정하는 것을 2진 분류(binary classification)라고 함

로지스틱 회귀에서 데이터가 특정 범주에 속할 확률을 예측하기 위한 과정

- ① 모든 속성(feature)들의 계수(coefficient)와 절편(intercept)을 0으로 초기화
- ② 각 속성들의 값(value)에 계수(coefficient)를 곱해서 log-odds를 계산
- ③ log-odds를 sigmoid 함수에 넣어서  $[0,1]$  범위의 확률 계산

# 로지스틱 회귀(Logistic Regression)

- Odds 란?

특정 사건이 발생할 확률을 발생하지 않을 확률로 나누어 준 값

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

- log- Odds 란?

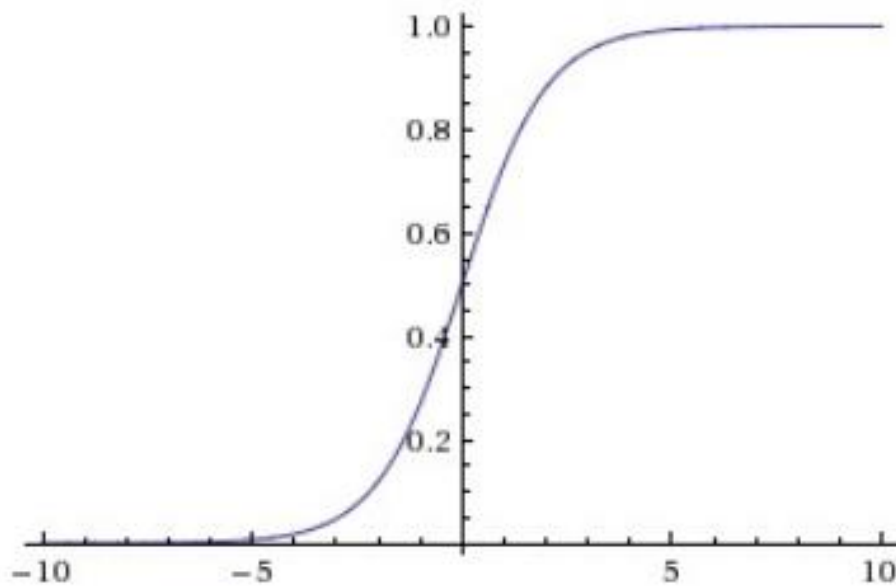
Odds에 로그를 취한 값

$$\log \left( Odds = \frac{P(event\ occurring)}{P(event\ not\ occurring)} \right)$$

# 로지스틱 회귀(Logistic Regression)

- 시그모이드(Sigmoid) 함수

어떠한 값이 입력되더라도 0 ~ 1 사이의 값 만을 출력하는 함수



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

# 로지스틱 회귀(Logistic Regression)

- 로그 손실(log loss)

- 로지스틱 회귀에 대한 손실 함수는 Log Loss(로그 손실)
- 로지스틱 함수를 구성하는 계수와 절편에 대해 Log Loss(로그 손실)을 최소화하는 값을 찾는 것
- 경사하강법(Gradient Descent)을 사용하여 모든 데이터에서 로그 손실(Log Loss)을 최소화하는 계수를 찾을 수 있음

- Classification Threshold (임계값)

- 대부분의 알고리즘에서 기본 임계 값은 0.5
- 필요에 따라 모델의 임계값을 변경
- 암 진단과 같은 민감한 모델의 경우 0.3이나 0.4로 임계값을 낮춰 모델의 민감도를 높임

# 로지스틱 회귀(Logistic Regression)

- 로지스틱 회귀 모델 fitting

```
from sklearn.linear_model.logistic import LogisticRegression
```

```
clf = LogisticRegression()
```

```
from sklearn.model_selection import GridSearchCV
```

```
params = {'C': [0.01, 0.1, 1], "penalty": ["l1", "l2"]}  
grid_search = GridSearchCV(clf, params, n_jobs=-1, cv=3, scoring="roc_auc")  
grid_search.fit(X_train, y_train)
```

```
print(grid_search.best_params_)
```

```
{'C': 1, 'penalty': 'l1'}
```

```
clf_best = grid_search.best_estimator_  
y_pred = clf_best.predict(X_test)
```

# 로지스틱 회귀(Logistic Regression)

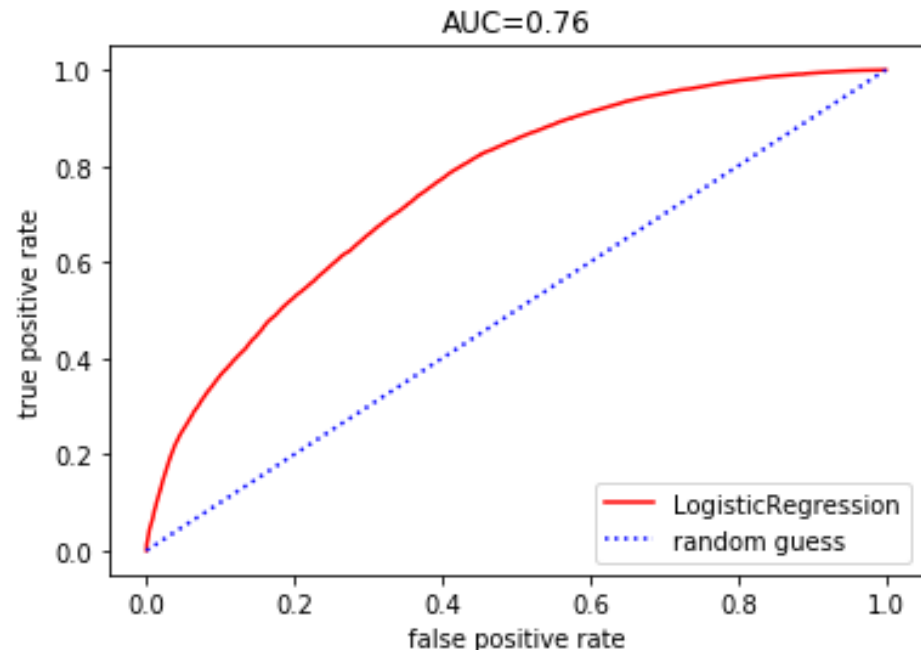
## ■ 모델 성능 평가

```
: from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score, roc_curve
accuracy = accuracy_score(y_test, y_pred)
confusion_matrix = confusion_matrix(y_test, y_pred)
print(accuracy, confusion_matrix)
y_pred_proba = clf_best.predict_proba(X_test)
```

```
0.83197 [[80838 1672]
        [15131 2359]]
```

```
: import matplotlib.pyplot as plt

fpr, tpr, _ = roc_curve(y_test, y_pred_proba[:, 1])
auc = roc_auc_score(y_test, y_pred_proba[:, 1])
plt.plot(fpr, tpr, "r-", label="LogisticRegression")
plt.plot([0, 1], [0, 1], "b:", label="random guess")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.title("AUC={0:.2f}".format(auc))
plt.legend(loc="lower right");
plt.show()
```





# K-최근접 이웃(Nearest Neighbors)

- KNN(K Nearest Neighbors) : 최근접 이웃 알고리즘

새로운 데이터의 분류를 알기 위해 사용

분류나 회귀에 사용할 수 있는 알고리즘으로 단순해 보이지만 강력하고 유용한 기법

훈련 단계에서 학습을 하지 않기 때문에 ‘게으른 학습’이라 부름

테스트/검증 단계에서 테스트 관측값과 가장 근접한 훈련 관측값을 비교

거리에만 의존하므로 차원의 저주에 따라 예측에 필요한 특징의 개수가 늘어나면 성능이 크게 저하



K값에 의해 결정된 분류를 새로운 데이터의 분류로 확정

# K-최근접 이웃(Nearest Neighbors)

- KNN(K Nearest Neighbors) : 최근접 이웃 알고리즘  
유클리드 공간의 점과 점 사이의 직선거리를 사용  
2차원 공간의 유클리드 거리의 계산방법 예)

$$d(p, q) = d(q, p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

K개의 가장 가까운 훈련 인스턴스를 골라 가장 많은 레이블을 분류로 선택  
특징의 표준화된 스케일링 필요

# K-최근접 이웃(Nearest Neighbors)

## ▪ 데이터 로드

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

```
%matplotlib inline
```

```
breast_cancer = pd.read_csv("data/breast-cancer-wisconsin.data", header=None)
```

```
breast_cancer.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

# K-최근접 이웃(Nearest Neighbors)

## ▪ 데이터 셋 컬럼명 지정

```
breast_cancer.columns = ["id_number", "clump_thickness", "unif_cell_size", "unif_cell_shape",  
                        "marg_adhesion", "single_epith_cell_size", "bare_nuclei", "bland_chromatin",  
                        "normal_nucleoli", "mitoses", "class"]
```

```
breast_cancer.head()
```

	id_number	clump_thickness	unif_cell_size	unif_cell_shape	marg_adhesion	single_epith_cell_size	bare_nuclei	bland_chromatin	normal_nucleoli	mitoses	class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

# K-최근접 이웃(Nearest Neighbors)

## ■ 이상 데이터 탐색

```
breast_cancer[breast_cancer.bare_nuclei=='?']
```

	id_number	clump_thickness	unif_cell_size	unif_cell_shape	marg_adhesion	single_epith_cell_size	bare_nuclei	bland_chromatin	normal_nucleoli	mitoses	class
23	1057013	8	4	5	1	2	?	7	3	1	4
40	1096800	6	6	6	9	6	?	7	8	1	2
139	1183246	1	1	1	1	1	?	2	1	1	2
145	1184840	1	1	3	1	2	?	2	1	1	2
158	1193683	1	1	2	1	3	?	1	1	1	2
164	1197510	5	1	1	1	2	?	3	1	1	2
235	1241232	3	1	4	1	2	?	3	1	1	2
249	169356	3	1	1	1	2	?	3	1	1	2
275	432809	3	1	3	1	2	?	2	1	1	2
292	563649	8	8	8	1	2	?	6	10	1	4
294	606140	1	1	1	1	2	?	2	1	1	2
297	61634	5	4	3	1	2	?	2	3	1	2
315	704168	4	6	5	6	7	?	4	9	1	2
321	733639	3	1	1	1	2	?	3	1	1	2
411	1238464	1	1	1	1	1	?	2	1	1	2
617	1057067	1	1	1	1	1	?	1	1	1	2

# K-최근접 이웃(Nearest Neighbors)

- 이상 데이터 NaN 처리 및 Data Imputation

```
breast_cancer.bare_nuclei = breast_cancer.bare_nuclei.replace("?", np.NaN)
```

```
breast_cancer.bare_nuclei.value_counts()
```

1	402
---	-----

10	132
----	-----

2	30
---	----

5	30
---	----

3	28
---	----

8	21
---	----

4	19
---	----

9	9
---	---

7	8
---	---

6	4
---	---

Name: bare\_nuclei, dtype: int64

```
breast_cancer.bare_nuclei.mode()[0]
```

'1'

```
breast_cancer.bare_nuclei = breast_cancer.bare_nuclei.fillna(breast_cancer.bare_nuclei.mode()[0])
```

# K-최근접 이웃(Nearest Neighbors)

- 이상 데이터 NaN 처리 및 Data Imputation

```
breast_cancer["class"].value_counts()
```

2	458
4	241

Name: class, dtype: int64

```
breast_cancer["cancer_ind"] = 0  
breast_cancer.loc[breast_cancer["class"] == 4, "cancer_ind"] = 1
```

```
breast_cancer.cancer_ind.value_counts()
```

0	458
1	241

Name: cancer\_ind, dtype: int64

```
X = breast_cancer.drop(["id_number", "class", "cancer_ind"], axis=1)  
y = breast_cancer.cancer_ind
```

# K-최근접 이웃(Nearest Neighbors)

- 데이터 분할, scaling 및 KNN 모델 fitting

```
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=42)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```



# K-최근접 이웃(Nearest Neighbors)

- 모델 성능 평가

```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_score
```

```
y_pred = knn.predict(X_test_scaled)
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(roc_auc_score(y_test, y_pred))
```

```
0.9761904761904762
[[141  2]
 [ 3 64]]
0.9706189333055005
```

# 나이브 베이즈(Naive Bayes)

- 나이브 베이즈 분류기(Naive Bayes Classifier)
  - 확률 기반 분류기(Classifier)
  - 데이터가 각 클래스에 속할 예측 특징 확률을 계산
  - 나이브(Naive) : 예측한 특징이 상호 독립적이라는 가정 하에 확률 계산을 단순화
  - 베이즈(Bayes) : 입력 특징이 클래스 전체의 확률 분포 대비 특정 클래스에 속할 확률을 베이즈 정리를 기반으로 계산

# 나이브 베이즈(Naive Bayes)

- 베이즈 정리와 나이브 베이즈

- 베이즈 정리에서  $P(A|B)$  는 사건 B가 주어졌을 때 사건 A가 일어날 확률

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- 나이브 베이즈의 목표는 n개의 특징을 가진 샘플 데이터 x가 주어졌을 때 k개의 클래스 중 하나에 속할 확률을 결정

- 샘플 데이터는  $x_1, x_2, x_3, \dots, x_n$  의 값을 가진 특징으로 구성되고,  $y_k$ 는 샘플 데이터가 k에 속하는 사건을 나타냄

$$x = (x_1, x_2, x_3, \dots, x_n)$$

$$P(y_k | (x_1, x_2, x_3, \dots, x_n)) = P(y_k | x) = \frac{P(x | y_k)P(y_k)}{P(x)}$$

# 나이브 베이즈(Naive Bayes)

- 나이브 베이즈 분류 절차

- ① 사후확률(Posterior Probability) : B가 발생했을때, A일 확률
- ② 사전확률(Prior Probability) : A일 확률
- ③ 우도(Likelihood) : B가 이전A에서 사용되었을 확률
- ④ 주변우도(Marginal Likelihood) : 모든 곳에서 B가 나타날 확률

# 나이브 베이즈(Naive Bayes)

## ■ 나이브 베이즈 분류 절차

‘A : 스팸메일, B : 광고’

- ① 사후확률(Posterior Probability) : ‘광고’가 스팸메일일 확률
- ② 사전확률(Prior Probability) : 이전 메일이 스팸메일일 확률
- ③ 우도(Likelihood) : ‘광고’가 이전 스팸메일에서 사용되었을 확률
- ④ 주변우도(Marginal Likelihood) : 모든 곳에서 ‘광고’가 나타날 확률

$$P(\text{스팸메일} \mid \text{광고}) = \frac{P(\text{광고} \mid \text{스팸메일})P(\text{스팸메일})}{P(\text{광고})}$$

Diagram illustrating the Naive Bayes formula for spam classification:

- 우도** (Likelihood) points to  $P(\text{광고} \mid \text{스팸메일})$
- 사전 확률** (Prior Probability) points to  $P(\text{스팸메일})$
- 사후 확률** (Posterior Probability) points to  $P(\text{스팸메일} \mid \text{광고})$
- 주변 우도** (Marginal Likelihood) points to  $P(\text{광고})$

# 나이브 베이즈(Naive Bayes)

- 나이브 베이즈 분류 절차

‘A : 스팸메일, B : 광고’

- ① 사후확률(Posterior Probability) : ‘광고’가 스팸메일일 확률
- ② 사전확률(Prior Probability) : 이전 메일이 스팸메일일 확률
- ③ 우도(Likelihood) : ‘광고’가 이전 스팸메일에서 사용되었을 확률
- ④ 주변우도(Marginal Likelihood) : 모든 곳에서 ‘광고’가 나타날 확률

$$P(\text{스팸메일} \mid \text{광고}) = \frac{P(\text{광고} \mid \text{스팸메일})P(\text{스팸메일})}{P(\text{광고})}$$

Diagram illustrating the Naive Bayes formula for spam classification:

- 우도 (Likelihood):**  $P(\text{광고} \mid \text{스팸메일})$
- 사전 확률 (Prior Probability):**  $P(\text{스팸메일})$
- 사후 확률 (Posterior Probability):**  $P(\text{스팸메일} \mid \text{광고})$
- 주변 우도 (Marginal Likelihood):**  $P(\text{광고})$

# 나이브 베이즈(Naive Bayes)

- 라플라스 스무딩(Laplace Smoothing)

특징의 출현 횟수 초기값을 1부터 시작해 0을 곱해 발생하는 문제를 해결  
(발견되지 않은 특징의 출현 빈도 초기값을 1로 설정)

- 언더플로우(underflow) 방지

$P(B|A)$ 가 너무 작아지면서 0에 거의 가까워질 수 있음

이러한 문제를 해결하기 위해 확률에 Log를 취하여 언더플로우를 방지함

$$\text{Log}(P(A|B)) = \text{Log}(P(B|A)P(A))$$

# 나이프 베이즈(Naive Bayes)

- enron 메일 데이터 셋 개요
  - Enron 이메일 데이터 세트 preprocessed format
  - <http://www.aueb.gr/users/ion/data/enron-spam/preprocessed/enron1.tar.gz>
  - 3,672 개의 정상 메일, 1,500 개의 스팸 메일로 구성
  - ham : 정상 메일이 담긴 폴더
  - spam : 스팸 메일이 담긴 폴더



Q&A

A 3D blue 'Q&A' text is centered on a white surface. A computer mouse with a grey cord is positioned to the right of the text, with the cord looping around the base of the 'Q'.

Thank you

A hand-drawn 'Thank you' note is shown on a white background. The text 'Thank you' is written in a cursive script, and a pen is visible at the end of the line.