스팸일까 아닐까?

로지스틱 회귀 알고리즘의 방법을 이해하고,
혼동행렬 평가 지표를 이용하여 성능을 평가

스팸일까 아닐까?

이번 활동에서는 **로지스틱 회귀(Logistic Regression) 알고리즘**을 이용하여 스팸 데이터셋에서 **스팸 문자를 분류**해 봅시다. 스팸 문자 속에 숨겨진 패턴을 발견하여 분류해 내는 인공지능 모델을 만들어 보겠습니다.

문제 정의하기	스팸 문자일까요? 아닐까요?
데이터 불러오기	캐글에서 스팸(Spam) 데이터셋 불러오기
데이터 처리하기	• 데이터 살펴보기 • 데이터 시각화하기 • 데이터 전처리하기
모델 학습하기	로지스틱 회귀로 학습하기
모델 테스트 및 평가하기	테스트 데이터로 평가하기

로지스틱 회귀의 이해

로지스틱 회귀란?

로지스틱 회귀

- 종속변수가 0과 1같은 이진분류에 적합
- 로짓(logit) 변환을 통해 분류 문제 해결
- 그래프 : S-자 형태의 곡선(그림 b)

선형 회귀

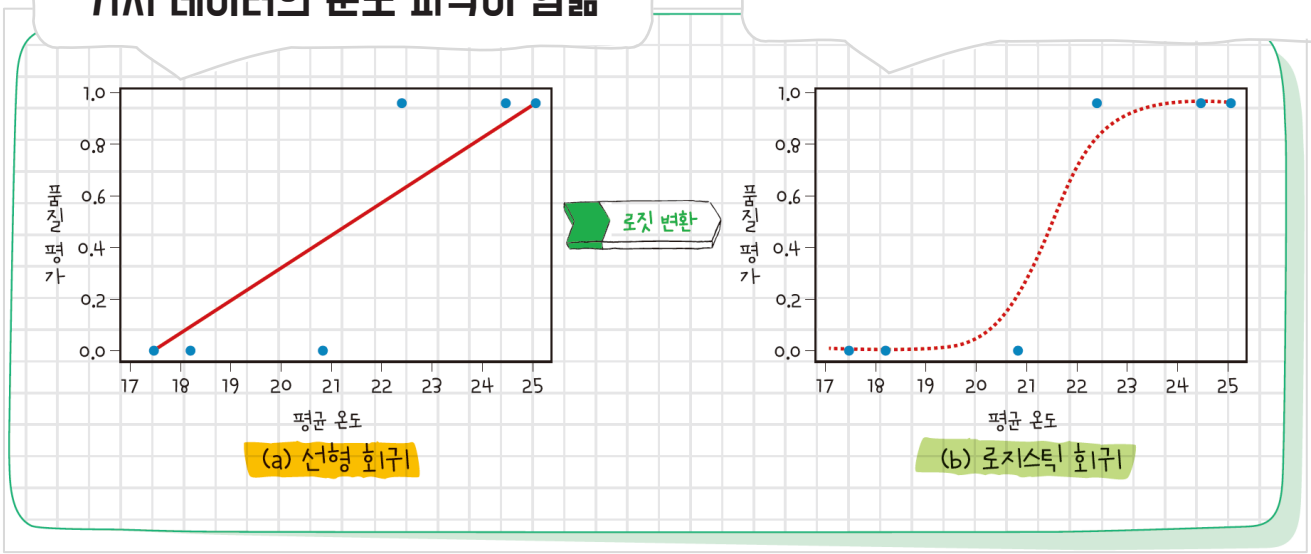
- 연속적인 값을 예측하는 데 사용
- 그래프 : 직선(그림 a)

평균 온도(x)	와인 품질 평가(y)
25	합격(1)
24.5	합격(1)
20.8	불합격(0)
18.2	불합격(0)
⋮	⋮

와인의 평균온도에 따라 와인 품질평가

데이터와 직선 간의 오차가 매우 커서 데이터의 분포 파악이 힘듦

로짓 변환을 통해 분류 문제 해결



로지스틱 회귀의 이해

로지스틱 회귀란?

선형 회귀

- 독립변수와 종속변수 사이의 선형적 관계를 표현
- 그래프 : 직선

로지스틱 회귀

- 독립변수와 종속변수 사이의 관계를 S자 곡선으로 표현
- S자 모양의 곡선은 데이터의 분포를 잘 파악하며 오차 감소
- 그래프 : S-자 형태의 곡선

로짓 변환

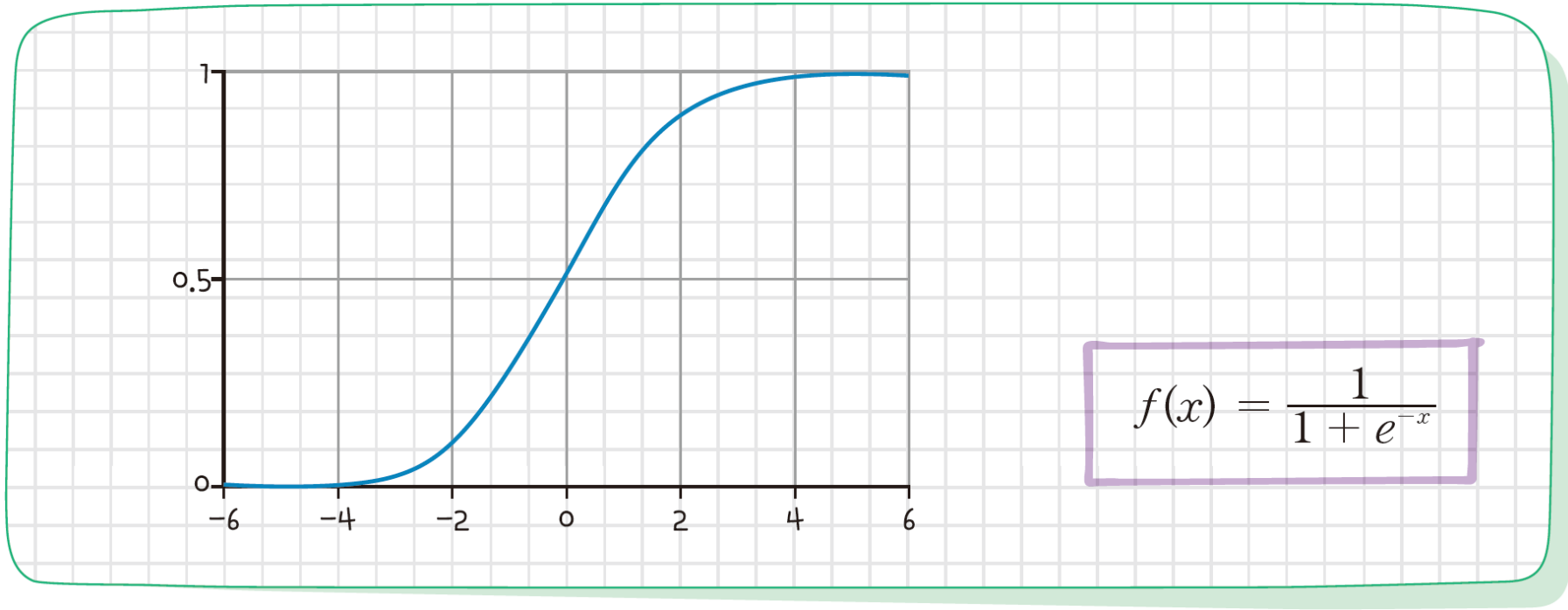
- 직선을 S자 곡선의 형태로 바꿔주는 변환
- 로지스틱 회귀는 로짓 변환을 통해 이진 분류 문제 해결 가능

"로지스틱 회귀는 종속변수가 범주형이며 0 또는 1인 경우에 사용,
선형 회귀에 로짓 변환하여 분류 문제를 해결할 수 있다."

로지스틱 회귀의 이해

로지스틱 함수

- 일반적으로 S자 모양의 곡선을 나타내는 수학 함수
- 통계학, 딥러닝, 생물학 등 여러 분야에서 사용
- 로지스틱 함수의 출력값은 보통 0과 1 사이의 값을 가짐

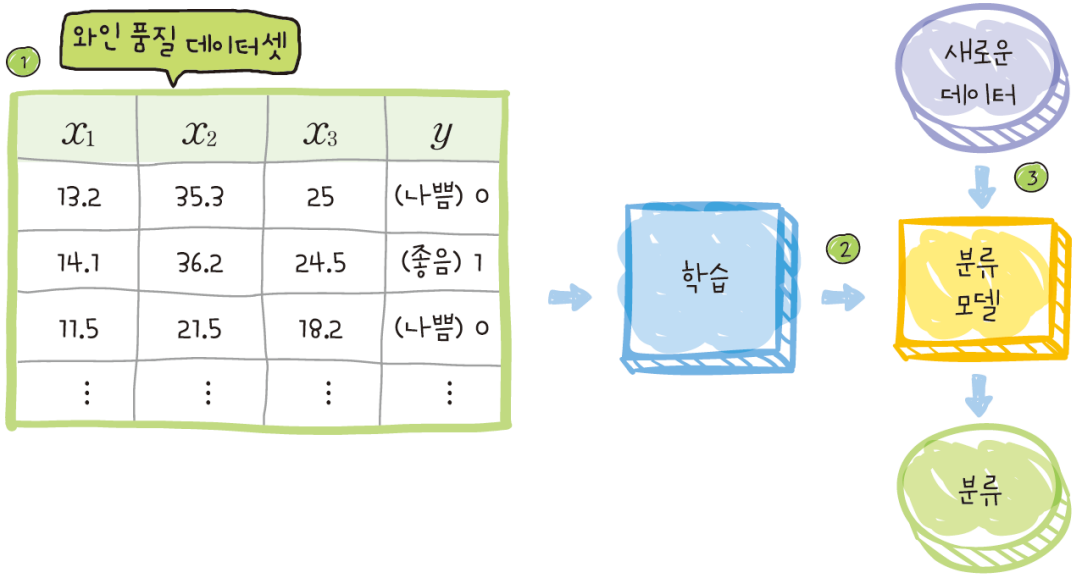


$$f(x) = \frac{1}{1 + e^{-x}}$$

로지스틱 회귀의 원리

로지스틱 회귀 알고리즘

로지스틱 회귀 알고리즘으로 와인 품질 데이터셋에서 와인의 고정 산도(x1), 잔여 설탕의 양(x2), 와인의 밀도(x3) 로 와인의 품질(y)을 분류해 보자.



1. 와인 품질 데이터셋은 3개의 독립변수와 한 개의 종속변수로 이루어져 있다.
2. 데이터를 학습시켜, 와인 품질을 분류하는 로지스틱 회귀 모델을 생성한다.
3. 새로운 데이터로 와인 품질을 분류한다.

문제 정의하기

문제 상황 이해하기

스팸 문자일까요? 아닐까요?

무수히 많은 스팸 메일은 필요한 메일 구분을 어렵게 하며, 교묘해진 피싱 스팸은 수신자에게 피해를 주지만, 인공지능은 스팸 메일의 문자 패턴을 통해 이를 분류하고 있습니다.

이번 활동에서는 스팸 문자를 구분해 내는 방법을 학습해 봅시다.

문제 정의하기

문제 해결에 필요한 정보 살펴보기

문제 해결 과정에서 필요한 정보를 미리 살펴봅시다.

1. 이 활동에 필요한 데이터셋은 무엇이고, 이 데이터셋은 어디에서 수집할 수 있나요?

데이터셋은 스팸 데이터셋으로, 캐글에서 다운로드할 수 있습니다.

2. 모델 학습에 사용할 알고리즘은 무엇인가요?

로지스틱 회귀 알고리즘을 사용합니다. 로지스틱 회귀는 결괏값을 0에서 1 사이의 범위로 예측하여 분류합니다.

문제 정의하기

문제 해결에 필요한 정보 살펴보기

3 모델 학습을 위해 어떤 처리를 해야 할까요?

데이터는 텍스트 데이터로 구성되어 있습니다. 텍스트 데이터의 시각화 방법인 워드 클라우드를 사용하고, 학습시키기 위해 텍스트를 숫자로 바꾸는 벡터 변환을 사용합니다.

데이터 불러오기

데이터셋 소개하기

스팸 데이터셋 살펴보기

- ham과 spam레이블로 이루어진 속성
- 텍스트 문자열로 이루어진 속성

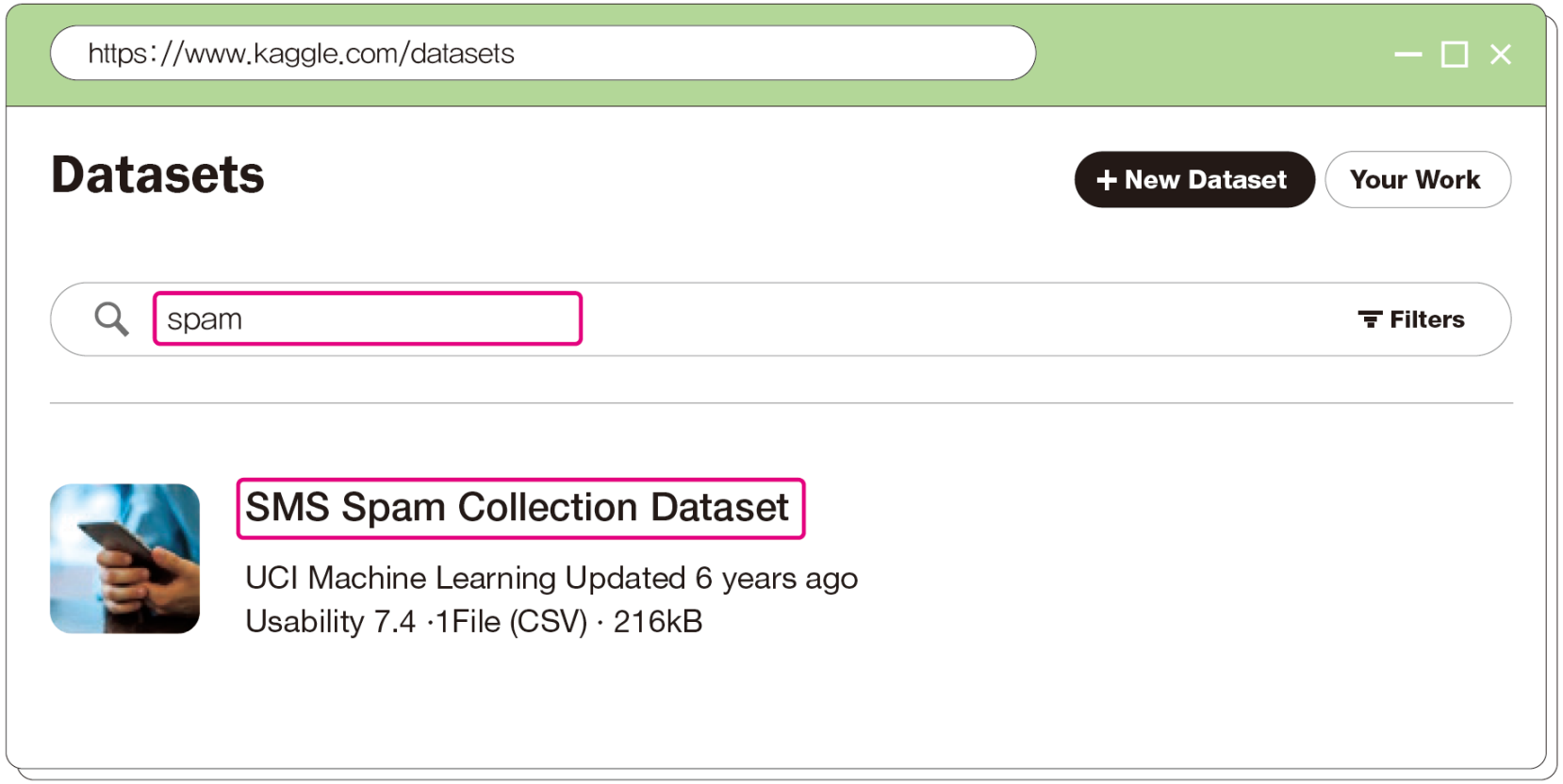
	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

데이터 불러오기

스팸 데이터셋 다운로드하기

① 캐글의 검색창에 'spam' 을 입력한 후 결과 중 'SMS Spam Collection Dataset'을 선택하기

https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset



데이터 불러오기

스팸 데이터셋 다운로드하기

② 하단의 데이터셋 미리보기 부분에서 다운로드 아이콘( ) 클릭하여 컴퓨터에 다운로드 한 후, 압축을 풀면 'spam.csv' 파일 확인 가능

데이터 불러오기

데이터셋 불러오기

파일 업로드하기

스팸 데이터셋을 코랩의 파일 업로드 기능을 이용하여 구글 드라이브에 업로드합니다.

1 아래의 코드를 실행한 후 파일 선택 튼을 클릭하고, 다운로드한 'spam.csv' 파일을 선택하여 업로드 업로드하기

```
1  from google.colab import files
2  filename = list(files.upload().keys())[0]
```

파일 선택 선택된 파일 없음 Cancel upload

2 업로드가 완료되면 'spam.csv로 저장되었다'는 안내 메시지 출력됨

파일 선택 spam.csv

•**spam.csv(text/csv)**-489242 bytes, last modified: 2023. 7. 1.-100% done

Saving spam.csv to spam.csv

데이터 불러오기

파일 읽어 들이기

'latin-1'은 csv 파일에 사용되는 문자 인코딩 방식으로 텍스트가 깨지지 않게 하기 위함입니다.

3 판다스 라이브러리를 이용하여 파일을 데이터프레임으로 읽어 들인 후, 불러온 스팸 데이터셋을 sms라는 이름으로 사용하고, 최상위 5개의 데이터를 출력해보기

```
1  import pandas as pd
2  sms = pd.read_csv(filename, encoding = 'latin - 1')  # CSV 파일을 읽어오기
3  sms.head()  # 데이터 상단의 5개 데이터 출력하기
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

해석

스팸 데이터셋을 확인해 보면 5개의 속성으로 이루어져 있습니다. 이 속성에는 ham과 spam 레이블로 이루어진 v1 열과 SMS 텍스트 문자열을 포함하는 v2 열을 확인할 수 있습니다.

데이터 처리하기

데이터 살펴보기

어떤 속성이 포함되어 있는지, 각 속성의 값은 어떤 유형의 데이터인지, 불필요한 데이터는 없는지 등을 파악하기

axis=1은 열을 의미하고, inplace = True는 변경된 내용을 원본 데이터에 바로 반영한다는 의미입니다.

```
데이터프레임 객체.drop(데이터프레임 객체.columns[[열의 번호]], axis = 1, inplace = True)
# drop( ) 함수는 데이터프레임에서 한 개 이상의 열 삭제
데이터프레임 객체.rename(columns = {'현재 칼럼명':'새로운 칼럼명', ···}, inplace = True)
# rename( ) 함수는 칼럼명을 변경할 때 사용
```

데이터 처리하기

불필요한 열 삭제하기

1 3개의 불필요한 Unnamed 열을 삭제하고, v1, v2의 열 이름을 'target'과 'message'로 변경하기

```
1  sms.drop(sms.columns[[2, 3, 4]], axis = 1, inplace = True)
2  sms.rename(columns = {'v1':'target', 'v2':'message'}, inplace = True)
3  sms.head()
```

	target	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

데이터 처리하기

데이터 통계량 살펴보기

2 판다스 라이브러리의 info( ) 메소드를 사용하여 데이터 기초 정보 확인하기

```
1  sms.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #    Column     Non-Null Count    Dtype
---   ------     --------------    -----
 0    target     5572 non-null     object
 1    message    5572 non-null     object
dtypes: object(2)
memory usage: 87.2+ KB
```

속성명	설명
target	ham 또는 spam을 나타내는 레이블 속성
message	SMS 텍스트 내용

스팸 데이터셋은 총 5,572개의 데이터로 구성되어 있고, 불필요한 데이터를 삭제하여 최종 속성은 2개입니다. 결측치가 없는 데이터 수는 5,572개로 총 데이터 수와 같습니다. 속성별 데이터 유형은 모두 문자형(object)으로 구성되어 있습니다.

데이터 처리하기

데이터 시각화하기

message 속성의 텍스트 내용에서 단어 빈도수를 파악하여 어떤 단어의 중요도가 높은지 워드 클라우드로 시각화해 보겠습니다.

워드 클라우드 표현하기

- 워드 클라우드(word cloud)는 시각적으로 강조하기 위해 중요도를 글자의 색상이나 굵기 등의 형태로 표현
- 중요도 : 단어의 빈도수 이용
- 단어의 빈도수는 글자 크기와 비례

```
WordCloud(colormap, width, height, max_words).generate(생성 대상)
# max_words가 500이면 빈도수 상위 50개만 출력
# 이밖에 max_font_size, min_font_size와 같은 속성도 사용 가능
```

데이터 처리하기

spam 워드 클라우드

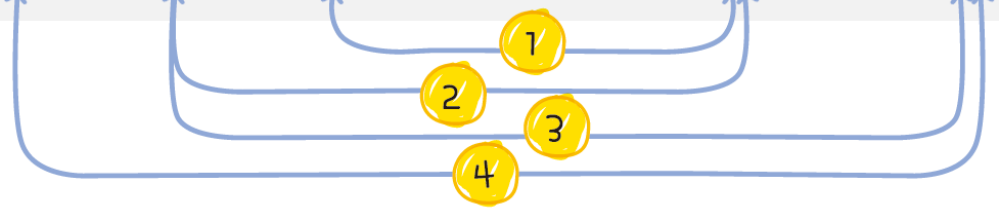
1 필요한 워드 클라우드 라이브러리와 맷플롯립 라이브러리를 불러오기

```
1  from wordcloud import WordCloud
2  import matplotlib.pyplot as plt
```

2 워드 클라우드에 사용할 문장은 target 속성값이 'spam'일 때의 message 속성의 텍스트를 합친 문장으로 만들어 보기

' '.join() # 띄어쓰기 단위(' ')로 문자열을 합치는 함수입니다.

```
1  spam_words = ' '.join(sms.loc[sms['target'] == 'spam']['message'])
```



데이터 처리하기

문자열 합치는 과정 알아보기

위의 코드에 대한 이해를 돕기 위해 아래와 같이 간단한 예시를 들어, 단계별로 알아봅시다.

1
```
sms['target'] == 'spam'
```
결과
```
0 False
1 True
2 False
3 True
```

2
```
sms.loc[sms['target'] == 'spam']
```
결과
```
  target   message
1 spam     I like grape
3 spam     I like melon
```

3
```
sms.loc[sms['target'] == 'spam']['message']
```
결과
```
1   I like grape
3   I like melon
```

4
```
' '.join(sms.loc[sms['target'] == 'spam']['message'])
```
결과
```
I like grape I like melon
```

예시 데이터

	target	message
0	ham	I like apple
1	spam	I like grape
2	ham	I like pineapple
3	spam	I like melon

데이터 처리하기

spam 워드 클라우드

3 합친 문장을 워드 클라우드로 출력하기

```
1  spam_wc = WordCloud(colormap = 'plasma', max_words = 50).generate(spam_words)
2  plt.figure(figsize = (24, 6))  # 그림의 너비와 높이(인치)
3  plt.axis('off')  # x, y축의 눈금 제거하기
4  plt.imshow(spam_wc)  # 워드 클라우드 출력하기
```

해석

target 속성값이 'spam'일 때의 message 속성값을 합친 문장을 워드 클라우드로 출력했더니 FREE, call, text와 같은 단어가 많이 등장함을 알 수 있습니다. 랜덤으로 발생합니다.

데이터 처리하기

ham
워드 클라우드

4 같은 방법으로 target 속성값이 'ham'일 때의 message 속성의 텍스트를 합친 문장으로 만들어 워드 클라우드로 출력하기

```
1  ham_words = ' '.join(sms.loc[sms['target'] == 'ham']['message'])
2  ham_wc = WordCloud(colormap = 'plasma', max_words = 50).generate(ham_words)
3  plt.figure(figsize = (24, 6))
4  plt.axis('off')
5  plt.imshow(ham_wc)
```

해석

target 속성값이 'ham'일 때를 워드 클라우드로 출력하면 will, go, ok와 같은 단어가 많이 등장함을 알 수 있습니다. 랜덤으로 발생합니다.

데이터 처리하기

데이터 전처리하기

머신러닝 모델을 학습시키기 위해서는 텍스트를 숫자로 바꾸는 데이터 전처리가 필요합니다.

텍스트를 숫자로 변환하기

- 사이킷런 라이브러리에서 제공하는 단어 카운트(CountVectorizer)를 이용
- 전체 텍스트에서 생성되는 고유 단어의 빈도수를 기준으로 주어진 문장을 벡터 변환 시 사용

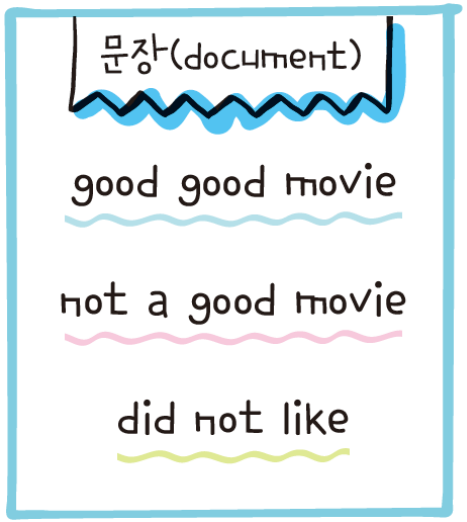
단어별로 분리한 후 빈도수에 따라 숫자로 표현한 형태를 독립변수인 feature로 사용합니다

```
CountVectorizer(max_features = 값)
# max_feature = 2500을 넘으면 최대 단어의 개수가 2500으로 제한한다는 의미
단어 카운트 객체.fit_transform(feature).toarray()
# feature 속성으로부터 각 단어의 빈도수를 배열 형태로 변환
```

데이터 처리하기

텍스트를 숫자로 변환하기

다음 3개의 문장 안에 들어 있는 단어의 종 류는 [good, movie, not, a, did, like]이다.
각 문장에서 해당 단어가 포함되는 개수를 숫자로 표시한 단어 카운트는 다음표와 같습니다.

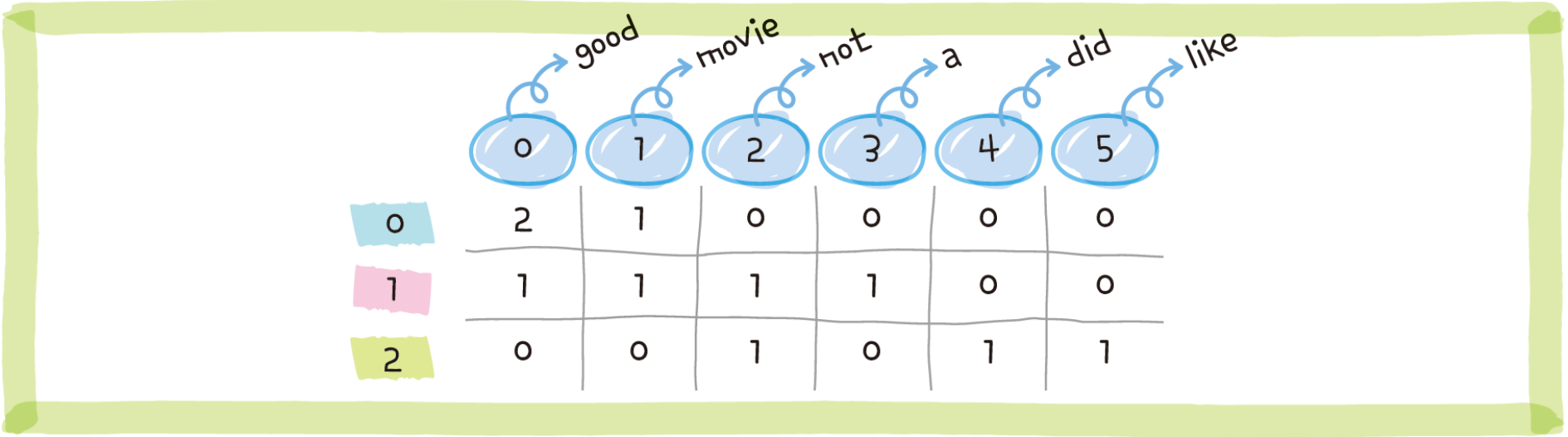


	good	movie	not	a	did	like
document[0]	2	1	0	0	0	0
document[1]	1	1	1	1	0	0
document[2]	0	0	1	0	1	1

데이터 처리하기

텍스트를 숫자로 변환하기

- 단어 카운트(CountVectorizer)는 고유 단어를 열로 표시
- 각 문장은 행으로 표시하여 행렬로 만듬.
- 행렬값은 문자열로 저장되지 않고, 특정 인덱스 값으로 지정됨
- 예) 'good'은 인덱스 0, 'movie'는 인덱스 1



데이터 처리하기

텍스트를 숫자로 변환하기

1 문서의 단어별 빈도수를 수치화하기 위해 단어 카운트 라이브러리 불러오기

```
1  from sklearn.feature_extraction.text import CountVectorizer
```

2 문자열을 수치화하기

```
1  X = sms['message']  # feature(독립변수 X)
2  y = sms['target']  # target(종속변수 y)
3  cv = CountVectorizer(max_features = 2500)
4  X = cv.fit_transform(X).toarray()  # 텍스트(단어) 수치화(배열 형태)
5  X  # 독립변수 출력하기
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

문자열로 이루어진 message 속성의 텍스트를 입력받아 fit_transform( ) 메소드를 사용하여 단어를 수치화합니다.

데이터 처리하기

훈련 데이터와 테스트 데이터 나누기

- 모델 학습에 사용할 훈련용 데이터와 모형 성능 평가에 사용할 테스트용 데이터로 나누기
- 훈련 데이터와 테스트 데이터의 비율 = 9:1

```
1  from sklearn.model_selection import train_test_split
2  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1,
3                                                      random_state = 0)
4  print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(5014, 2500) (558, 2500) (5014, ) (558, )
```

random_state를 지정하지않으면코드를실행할때마다다새롭운무작위값이생성되어 훈련 및 테스트 데이터는 매번 다른 값을 갖게 돼요. 그러나 random_state = 0, 1, 42 또는다른정수와같은고정값을지정하면코드를몇번실행해도결과는동일합니다

독립변수 X, 종속변수 y로부터 훈련 데이터 : 테스트 데이터 = 9 : 1로 분할하였습니다. 따라서 훈련용 데이터의 개수는 5014, 테스트용 데이터의 개수는 558개입니다. 데이터를 추출하는 패턴(random_ state)은 0으로 고정값을 지정하여 코드를 몇 번 실행해도 결과가 동일하게 나오도록 하였습니다.

모델 학습하기

모델 생성하기

- 분류를 위한 머신러닝 모델은 여러 종류가 있고, 각각 학습 방식과 정확도가 다름
- 스팸 여부를 판별하기 위한 이진 분류로 '로지스틱 회귀' 모델 사용
- 로지스틱 회귀는 사이킷런 라이브러리로 쉽게 불러올 수 있음

1 사이킷런 라이브러리에서 로지스틱 회귀 라이브러리 불러오기

```
1  from sklearn.linear_model import LogisticRegression
```

모델 학습하기

모델 생성하기

2 사이킷런 라이브러리에서 제공하는 LogisticRegression( )을 통해 로지스틱 회귀 모델을 생성
생성한 모델을 LR_model이라는 이름으로 사용
여기서 생성한 모델 은 사람에 비유하면 아직 학습하지 않은 뇌 구조에 해당됨

```
1  LR_model = LogisticRegression(solver = 'liblinear')
```

LogisticRegression(solver = 'liblinear')

solver는 모델의 최적화에 사용할 알고리즘 지정

'liblinear'는 작은 데이터셋에 적합한 경사 하강법 기반의 최적화 알고리즘

작은 데이터셋에서 이진 분류 또는 다중 클래스 분류 문제에 로지스틱 회귀 모델을 학습할 때 사용하는 옵션

모델 학습하기

훈련 데이터로 모델 학습하기

- 머신러닝에서 모델 학습을 할 때 모델이 학습할 훈련 데이터(독립변수)와 훈련 데이터의 레이블(종속변수)을 설정
- X_train을 훈련 데이터로, y_train을 훈련 데이터의 레이블로 설정
- fit( ) 함수를 사용하면 쉽게 학습가능

```
1  LR_model.fit(X_train, y_train)  # 훈련 데이터
```

모델 테스트 및 평가하기

테스트 및 평가하기

- accuracy_score( ) 함수를 사용하여 테스트 데이터의 독립변수값을 넣어 실제 테스트 데이터의 종속변숫값과 예측값이 얼마나 일치하는지 정확도를 확인하기

- 테스트 데이터(X_test)를 사용하여 예측값 생성하기

```
1  from sklearn.metrics import accuracy_score
2  y_pred = LR_model.predict(X_test)  # 테스트 데이터 예측값
```

- 생성한 예측값과 실젯값 사이의 정확도 계산하기

```
1  print('Accuracy:{:.3f}'.format(accuracy_score(y_test, y_pred)))
```

```
Accuracy:0.978
```

정확도 계산
(464+82)/cm.sum( )
0.978494623655914

해석

테스트 데이터를 예측할 샘플 데이터로 입력하여 클래스 레이블을 예측하여 벡터로 나타냅니다. 테스트 데이터의 클래스 레이블과 예측한 클래스 레이블과는 약 98% 일치합니다.

모델 테스트 및 평가하기

255쪽

혼동 행렬 출력하기

- 예측 결과 분포를 표로 정리하여 쉽게 알아볼 수 있는 혼동 행렬은 사이킷런 내에서 confusion_matrix( ) 함수로 지원 테스트 데이터(X_test)를 사용하여 예측값 생성하기

```
1  from sklearn.metrics import confusion_matrix
2  cm = confusion_matrix(y_test, y_pred)
3  cm
```

```
array([[464,   1],
       [ 11,  82]])
```

혼동 행렬		예측	
		True	False
실제	True	TP 464	FN 1
	False	FP 11	TN 82

Logistic Regression 문제 해결 과정

문제 정의하기

스팸 문자일까요? 아닐까요?

↓

데이터 불러오기

캐글에서 스팸 데이터셋 불러오기

↓

데이터 처리하기

- 데이터 살펴보기(불필요한 열 삭제, 기초 정보 확인하기)
- 데이터 시각화하기(워드 클라우드로 표현하기)
- 데이터 전처리하기(텍스트를 숫자로 변환하기

↓

모델 학습하기

- 로지스틱 회귀로 학습하기

↓

모델 테스트 및 평가하기

- 테스트 데이터로 평가하기

우리가 탐색한 정보

1. 이 활동에 필요한 데이터셋은 무엇이고, 이 데이터셋은 어디에서 수집할 수 있었나요?

데이터셋은 스팸 데이터셋으로, 캐글에서 다운로드할 수 있습니다.

2. 모델 학습에 사용한 알고리즘은 무엇이었나요?

로지스틱 회귀 알고리즘을 사용합니다. 로지스틱 회귀는 연속적인 값을 갖는 독립변수와 0과 1의 범주형 값을 갖는 종속변수로 이루어져 있어, 데이터를 분류하는 알고리즘입니다.

3. 모델 학습을 위해 우리가 해야 할 작업은 무엇이었나요?

스팸 데이터셋에 불필요한 속성을 삭제하여 새로운 데이터셋을 만들어 사용합니다.

4. 이 활동에서 새롭게 알게 된 정보는 무엇이었나요?

로짓 변환: 선형 회귀와 같은 직선을 S자 곡선의 형태로 변환해 주는 것을 로짓 변환이라 하며, 이를 통해 선형 회귀보다 오차를 줄이고 데이터 분포를 잘 파악하여, 분류 문제를 해결할 수 있습니다.

지금까지 로지스틱 회귀(Logistic Regression)를 살펴보았습니다.
로지스틱 회귀는 범주형 데이터의 분류 문제에 사용되는 분석 방법입니다. 주로 이진 분류에 사용되며, 입력된 데이터를 기반으로 해당 데이터가 어떤 클래스에 속할 확률을 추정하여 예측합니다.
이 활동에서는 입력받은 텍스트 문자열이 스팸인지 아닌지를 분류하기 위하여 텍스트를 숫자로 바꾸는 데이터 전처리를 합니다. 전체 텍스트에서 생성되는 고유 단어의 빈도수를 기준으로 주어진 문장을 벡터로 변환하는데 사이킷런 라이브러리에서 제공하는 단어 카운트(CountVectorizer)를 이용합니다. 벡터로 변환된 데이터를 로지스틱 회귀 라이브러리를 이용하여 모델을 생성하고, 테스트 데이터를 이용하여 평가합니다. 분류 모델의 성능 평가 지표인 혼동 행렬을 이용하여 모델의 성능을 확인합니다.
로지스틱 회귀는 오즈(odds)를 로짓(logit) 변환하여 얻을 수 있습니다. 오즈는 단순한 확률을 나타내는 것이 아닌 어느 정도의 승산이 있는지 나타내는 비율을 말합니다. 로짓 변환은 오즈에 로그 함수를 취 한 로그 오즈와 선형 회귀를 수식으로 풀어 놓은 것을 의미합니다. 로그 오즈를 이용하여 0을 기준으로 대칭인 함수를 만들면 더 좋은 성능의 로지스틱 함수를 만들 수 있습니다. 또한 로지스틱 함수는 딥러닝의 활성화 함수에 해당합니다. 이러한 내용을 바탕으로 선형 회귀와 로지스틱 회귀가 딥러닝의 원리를 파악 하는 데 도움이 될 것으로 예상합니다.