



Day51; 20221117

날짜	@2022년 11월 17일
유형	@2022년 11월 17일
태그	

GitHub - u8yes/AI

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or window. Reload to refresh your session. Reload to refresh your session.

<https://github.com/u8yes/ai>

u8yes/AI



1 Contributor 0 Issues 0 Stars 0 Forks

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a13ce9f1-3340-4b52-856a-899dfa0cabb6/05-1_Gaussian_naive_bayes_20221117.ipynb

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a2735a29-02d7-42de-a17e-282f2871755f/05-2_Bernoulli-Nave-Bayes_20221117.ipynb

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/96cc8640-b18d-4182-9c2d-69d814a090b3/data_221117\(1\).txt](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/96cc8640-b18d-4182-9c2d-69d814a090b3/data_221117(1).txt)

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e9e3cb06-7b66-4c69-993a-479621d33fd8/data_221117\(2\).txt](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e9e3cb06-7b66-4c69-993a-479621d33fd8/data_221117(2).txt)

나이브 베이즈

나이브 베이즈 - 스팸 메일 분류로 가장 많이 쓰인다.

확률 통계 기반 알고리즘, 조건부 확률

빅분기 실기 - 머신러닝)

```
# [3] 모델 학습 및 성능 평가 (train 데이터 사용)
from sklearn.preprocessing import StandardScaler
def ModelTrain(model, df):
    # X, Y 데이터 준비
    Y = df['학점']
    X = df.drop(columns=['학점'])
    # X_scaled = StandardScaler(X).fit_transform(X)

    # train, test 데이터 분할
    xtrain, xtest, ytrain, ytest = train_test_split(X, Y,
                                                    test_size=0.3,
                                                    stratify=Y,
                                                    random_state=0)

    # 모델 선택 및 학습
    #model = LogisticRegression(max_iter=5000)
    model.fit(xtrain, ytrain)

    # 성능평가 - accuracy, roc_auc_s
```

조건부 확률은 이미 벌어진 일에 대해서만 전체로 보기 시작해서 표본집단이 줄어든다.

나이브 베이즈(Naïve Bayes)

- 확률 기반 머신러닝 분류 알고리즘.
- 데이터를 나이브(단순)하게 독립적인 사건으로 가정하고, 이 독립 사건들을 베이즈 이론에 대입시켜 가장 높은 확률의 레이블로 분류를 실행하는 알고리즘.
- 베이즈 이론

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

- $P(A | B)$: 어떤 사건 B가 일어났을 때 사건 A가 일어날 확률.
- $P(B | A)$: 어떤 사건 A가 일어났을 때 사건 B가 일어날 확률.
- $P(A)$: 어떤 사건 A가 일어날 확률.

- 조건부 확률

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

B가 이미 일어난 일, 거기에 A가 일어남.

전체를 B로 봄.

겹쳐진 A(교집합)을 중점으로 출발한 것이 베이즈 이론.

영화 리뷰에 대한 데이터를 모아서 좋은, 나쁜 평가를 찾는 게 대표적 텍스트 마이닝.

베이즈 이론

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

- $P(A | B)$: 어떤 사건 B가 일어났을 때 사건 A가 일어날 확률.
- $P(B | A)$: 어떤 사건 A가 일어났을 때 사건 B가 일어날 확률.
- $P(A)$: 어떤 사건 A가 일어날 확률.

예) 이미 일어난 B를 feature라고 부른다.(column)

A를 label, 정답으로 본다.

예: 군대B를 다녀왔느냐에 따라 성별A를 알고 싶은 것이 결과. 그래서 A가 중요하다.

이전에는 데이터가 많이 필요했다.

베이지스이론은 가지고 있는 데이터(기존데이터)를 가지고 예측 모델을 만들 수 있다.

기존 데이터의

이전에서는 모든 데이터를 기준으로 했는데 하지만 베이지 이론은 B 데이터를 기준으로 한다.

치킨을 먹었는데 술을 주문했는지 안 했는지 궁금.

이 손님이 술을 주문할지 안 할지를 예측.

나이브 베이즈 알고리즘의 기초 응용 사례

- 치킨 집에서 손님이 주문을 할 때 맥주를 주문할지 안 할지 예측.
- 저녁에 손님이 한 명 와서 주문 시 나이브 베이즈 공식에 따르면 손님이 맥주를 주문 할 확률.

- $$P(\text{주문}|\text{저녁}) = P(\text{저녁}|\text{주문}) * P(\text{주문}) / P(\text{저녁})$$
$$= (3 / 4) * (4 / 10) / (5 / 10) = 0.6$$

시간	맥주
오전	주문 안 함
오전	주문 안 함
점심	주문함
점심	주문 안 함
점심	주문 안 함
저녁	주문함
저녁	주문함
저녁	주문함
저녁	주문 안 함
저녁	주문 안 함

- 기존 손님들의 주문 내역

$$= P(\text{저녁, 성인}|\text{주문}) * P(\text{주문}) / P(\text{저녁, 성인})$$

시간	성인여부	맥주
오전	성인	주문 안 함
오전	미성년자	주문 안 함
점심	성인	주문함
점심	미성년자	주문 안 함
점심	미성년자	주문 안 함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문함
저녁	성인	주문 안 함
저녁	미성년자	주문 안 함

• 기존 손님들의 주문 내역

독립은 배반이 아니다. 독립의 반대는 종속이다.

배반은 A가 일어나면 B가 일어나지 않는다.

시그마는 +이지만 파이는 곱해줌.

➤ 여러 확률의 곱을 나타내는 $P(x_1 | y) * P(x_2 | y) * \dots * P(x_n | y)$ 부분을 간단하게 공식화하면

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1)P(x_2) \dots P(x_n)}$$

➤ 독립 사건일 경우의 조건부 확률은 다음과 같이 단순해 짐.

- $P(A, B) = P(A) * P(B)$

P(A)와 P(B)는 독립이기 때문에 P(A)만 구하면 됨.

독립 사건일 경우의 조건부 확률

여기서 실제로 필요한 것은 두 값의 대소를 비교해서 큰 값을 선택하는 것이므로 다음과 같이 공통 분모를 계산식에서 제거해서 계산량을 더 줄일 수 있음.



- $P(\text{주문} | \text{저녁, 성인}) = P(\text{저녁}|\text{주문}) * P(\text{성인}|\text{주문}) * P(\text{주문}) = 0.3$

- $P(\text{주문 안함} | \text{저녁, 성인}) = P(\text{저녁}|\text{주문 안함}) * P(\text{성인}|\text{주문 안함}) * P(\text{주문 안함}) = 0.066$

P(주문 | 저녁, 성인)이 더 크므로 저녁에 성인이 치킨 집에 올 경우 맥주를 주문할 것으로 분류.

인간이 독립인지 종속인지 판단할 수 없다.

naive

미국·영국 [naɪˈiːv]  영국식 


형용사

1 (경험·지식 부족 등으로) 순진해 빠진, (모자랄 정도로) 순진한

to be politically naive 

정치적으로 순진해 빠졌다[모자라다]

2 순진무구한 (→sophisticated), (=artless)

Their approach to life is refreshingly naive. 

삶에 대한 그들의 접근법은 신선할 정도로 순진무구하다.

3 천진난만한 스타일의

[영어사전 결과 더보기](#)

분류 알고리즘이 이산형.

하지만 연속형에도 잘 맞는다, 단 전제조건은 그 data가 정규분포를 이루고 있을 때.

가우시안 나이브 베이즈 분류

- 앞서 다룬 예제는 데이터의 특징들이 이산적(discrete)인 간단한 예임.
- 데이터의 특징들이 연속적인 경우에는 가우시안 나이브 베이즈 분류를 사용하는 것을 추천.
- 특징들의 값들이 정규 분포(가우시안 분포)에 있다는 가정하에 조건부 확률을 계산.
- 연속적인 성질이 있는 특징이 있는 데이터를 분류하는데 적합.
- 예) 붓꽃(iris) 데이터셋 분류

cf) 스무딩(smoothing) 이란?

- 이산적인 데이터의 경우 빈도수가 0인 경우가 발생
- 예를 들어, 나이브 베이즈 기반 스팸 메일 필터를 가동시키는데, 학습 데이터에 없던 단어가 실제 상황에서 나타나게 되면 확률이 0이 되어 스팸 분류가 어려워짐.
- 스무딩은 학습 데이터에 없던 데이터가 출현해도 빈도수에 1을 더해서 확률이 0이 되는 현상을 방지

실수값을 가지는 것은 항상 연속변수이다.

다항 분포 나이브 베이즈(Multinomial Naïve Bayes) 분류

주사위(1번,2번,3번,4번,5번,6번주사위번호) - 4번주사위번호가 4번 나왔다.

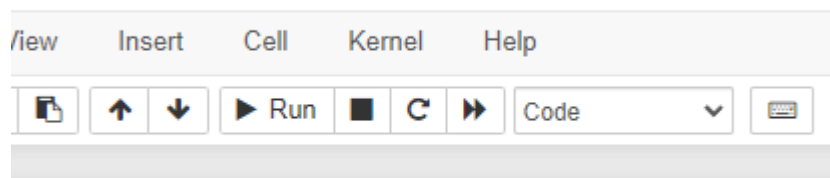
(1, 2, 3, 4, 0, 0):

베르누이 나이브 베이즈 모델(Bernoulli Naïve Bayes)

주사위(1번,2번,3번,4번,5번,6번주사위번호) - 나왔다 1 나오지 않았다 0

(1, 1, 1, 1, 0, 0).

05-1_Gaussian_naive_bayes Last Checkpoint: 몇 초 전 (



```
] : # 라이브러리 임포트 : 실습에 필요한 라이브러리를 임포트
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
from sklearn.metrics import accuracy_score

np.random.seed(5)
```

iris 데이터 시각화

```
: # iris 데이터를 불러옴.  
dataset = load_iris()  
  
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)  
df.head()
```

```
:  
      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  
0          5.1          3.5          1.4          0.2  
1          4.9          3.0          1.4          0.2  
2          4.7          3.2          1.3          0.2  
3          4.6          3.1          1.5          0.2  
4          5.0          3.6          1.4          0.2
```

```
: # 분류값을 데이터프레임에 저장  
df['target'] = dataset.target  
df.head()
```

```
:  
      sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target  
0          5.1          3.5          1.4          0.2          0  
1          4.9          3.0          1.4          0.2          0  
2          4.7          3.2          1.3          0.2          0  
3          4.6          3.1          1.5          0.2          0  
4          5.0          3.6          1.4          0.2          0
```

```
: df.target.value_counts()  
:  
0    50  
1    50  
2    50  
Name: target, dtype: int64
```

```

: # 숫자만 분류값의 이해를 돕기 위해 문자로 변경.
df.target = df.target.map({0:'setosa', 1:'vesicolor', 2:'virginica'})
df.head()

```

```

:

```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
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

map이 아니라 mapping한다는 의미.

iris 데이터의 분포도 확인_20221117

- iris 데이터의 분포도를 확인해보도록 하겠습니다. 결과를 통해서, iris의 데이터 분포도가 정규분포(Gaussian Distribution)을 이루고 있는지를 확인합니다.

```

: # 분류값 별로 데이터 프레임을 나눕니다.
setosa_df = df[df.target == 'setosa']
vesicolor_df = df[df.target == 'vesicolor']
virginica_df = df[df.target == 'virginica']

```

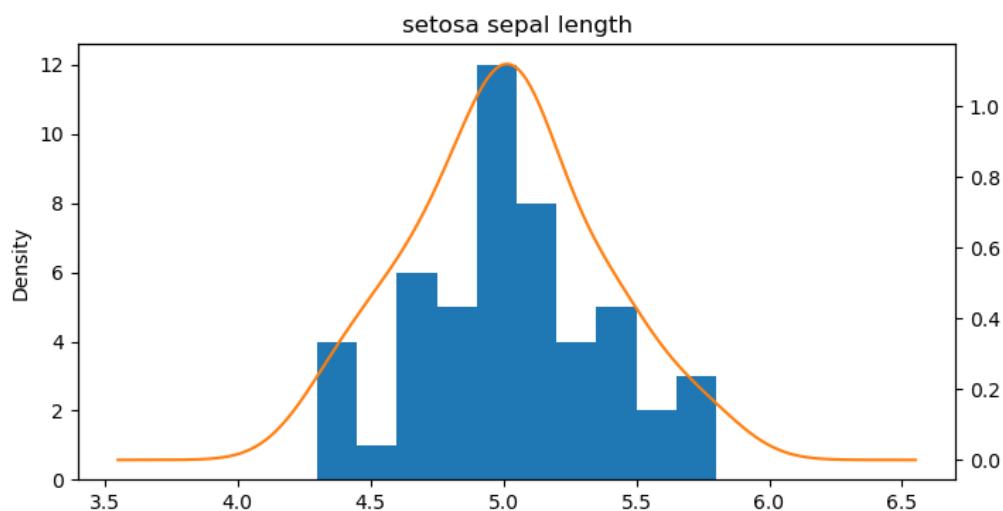
setosa sepal length (cm)

```

ax = setosa_df['sepal length (cm)'].plot(kind='hist')
setosa_df['sepal length (cm)'].plot(kind='kde', ax=ax, secondary_y=True, title='setosa sepal length', figsize=(8,4))
# kde - 표준밀도의 형태로 출력 # ax 추가 식별값 # secondary_y - y축에 표시를 할 건지 말 건지 셋팅

```

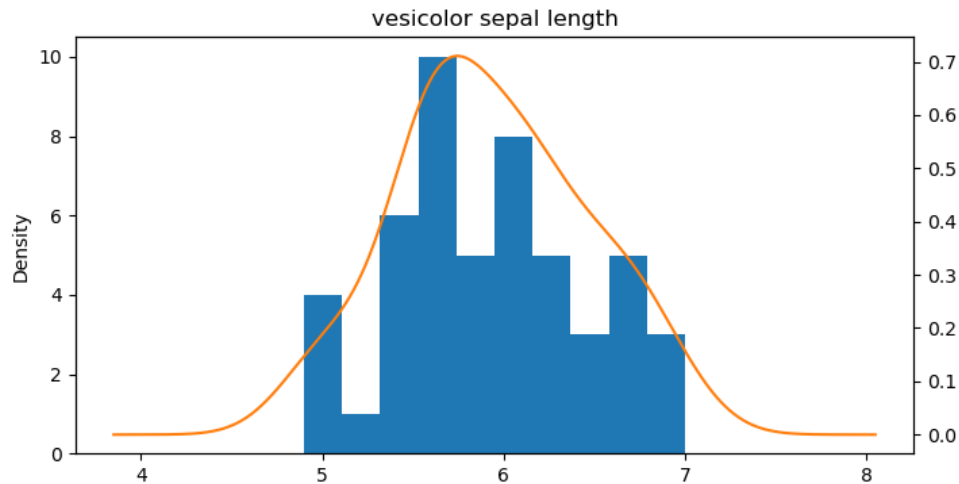
<AxesSubplot :>



vesicolor sepal length (cm) ¶

```
ax = vesicolor_df['sepal length (cm)'].plot(kind='hist')
vesicolor_df['sepal length (cm)'].plot(kind='kde', ax=ax, secondary_y=True, title='vesicolor sepal length', figsize=(8,4))
# kde - 표준밀도의 형태로 출력 # ax 추가 식별값 # secondary_y - y축에 표시를 할 건지 말 건지 셋팅
```

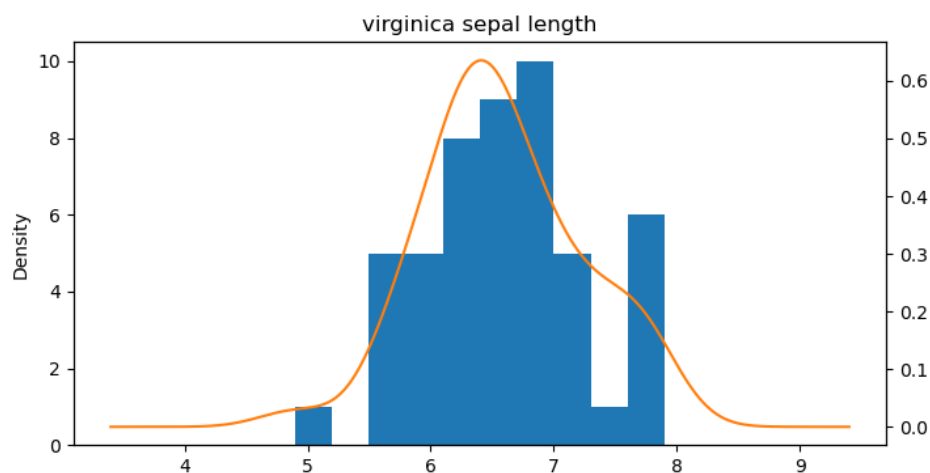
<AxesSubplot:>



virginica sepal length (cm)

```
ax = virginica_df['sepal length (cm)'].plot(kind='hist')
virginica_df['sepal length (cm)'].plot(kind='kde', ax=ax, secondary_y=True, title='virginica sepal length', figsize=(8,4))
# kde - 표준밀도의 형태로 출력 # ax 추가 식별값 # secondary_y - y축에 표시를 할 건지 말 건지 셋팅
```

<AxesSubplot:>



... 9개를 더해보아야함

데이터를 학습 데이터와 테스트 데이터로 나누기

```
|: # 20%를 테스트 데이터로 분류합니다.  
X_train, x_test, y_train, y_test = train_test_split(dataset.data, dataset.target, test_size=0.2)  
# test_size 검증 20% 설정
```

fit (fit)

미국·영국[fit]  영국식 

- 1 (모양크기가 어떤 사람사물에) 맞다
- 2 (어느 장소에 들어가기에) 맞다
- 3 (특히 규칙적인 운동으로 몸이) 건강한[탄탄한] (→keep-fit), (↔unfit)
- 4 적합한, 알맞은, 어울리는 (↔unfit)

Gaussian Naive Bayes 분류하기

```
: # 학습데이터 모델을 학습합니다.  
model = GaussianNB()  
model.fit(X_train, y_train) # 데이터의 80%를 넣어줄(train)  
  
# 테스트 데이터로 모델을 테스트합니다. # predict는 무조건 test  
predicted = model.predict(X_test) # X_test - 20% 데이터를 예측하라고 시킴
```

```
: print(metrics.classification_report(y_test, predicted))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.62	1.00	0.77	5
2	1.00	0.80	0.89	15
accuracy			0.90	30
macro avg	0.88	0.93	0.89	30
weighted avg	0.94	0.90	0.91	30

JAVA)

실행결과

ArrayList 걸린시간: 20248953 ns
LinkedList 걸린시간: 4279517 ns

구분	순차적으로 추가/삭제	중간에 추가/삭제	검색
ArrayList	빠르다	느리다	빠르다
LinkedList	느리다	빠르다	느리다

accuracy

미국·영국[ˈækjərəsi] 영국식

명사

정확, 정확도 (↔inaccuracy)

They questioned the **accuracy** of the information in the file.

그들은 그 파일 속 정보의 정확성에 대해 의문을 제기했다.

영어사전 다른 뜻 1

```
]: # 라이브러리 임포트 : 실습에 필요한 라이브러리를 임포트

import numpy as np
import pandas as pd

# 베르누이 나이브베이즈를 위한 라이브러리
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import BernoulliNB

# 모델의 정확도 평가를 위해 임포트합니다.
from sklearn.metrics import accuracy_score # 정확도

np.random.seed(5)
```

문제 정의

- 베르누이 나이브베이즈 분류 모델을 사용하여 스팸 메일을 분류해보겠습니다.

데이터 수집

- 이번 실습에서는 간단한 스팸 메일 분류실습을 위해 아래 이메일 타이틀과 스팸 여부가 있는 데이터를 사용하겠습니다.

feature는 email title이고, 정답은 spam이다.

```
email_list = [
    {'email title': 'free game only today', 'spam': True},
    {'email title': 'cheapest flight deal', 'spam': True},
    {'email title': 'limited time offer only today only today', 'spam': True},
    {'email title': 'today meeting schedule', 'spam': False},
    {'email title': 'your flight schedule attached', 'spam': False},
    {'email title': 'your credit card statement', 'spam': False}
]

df = pd.DataFrame(email_list)
df
```

	email title	spam
0	free game only today	True
1	cheapest flight deal	True
2	limited time offer only today only today	True
3	today meeting schedule	False
4	your flight schedule attached	False
5	your credit card statement	False

데이터 전처리

- sklearn의 베르누이 나이브베이지 분류기는 숫자만을 다루기 때문에, True와 False를 1과 0으로 치환하겠습니다.

```
df['label'] = df['spam'].map({True: 1, False: 0})
df
```

	email title	spam	label
0	free game only today	True	1
1	cheapest flight deal	True	1
2	limited time offer only today only today	True	1
3	today meeting schedule	False	0
4	your flight schedule attached	False	0
5	your credit card statement	False	0

【 지도학습과 비지도학습 】***

지도학습		비지도학습	
회귀 (연속형)	<ul style="list-style-type: none"> 선형회귀분석(Linear Regression) 의사결정나무(회귀트리모형) SVR(Support Vector Regression) 신경망 모형 릿지(Ridge) 라쏘(Lasso) 	군집	<ul style="list-style-type: none"> K-means SOM DBSCAN(밀도 기반 군집) 병합 군집 계층 군집
		연관	<ul style="list-style-type: none"> Apriori
분류 (범주형)	<ul style="list-style-type: none"> 로지스틱 회귀분석 신경망 모형 의사결정나무(분류트리모형) k-NN(k-최근접 이웃 알고리즘) 앙상블모형 SVM(Support Vector Machine) 나이브 베이지 분류 	차원축소	<ul style="list-style-type: none"> PCA(주성분분석) LDA(선형판별분석) SVD(특잇값 분해) MDS(다차원 척도법)

(2) 데이터 분할을 통한 검증 ★★★

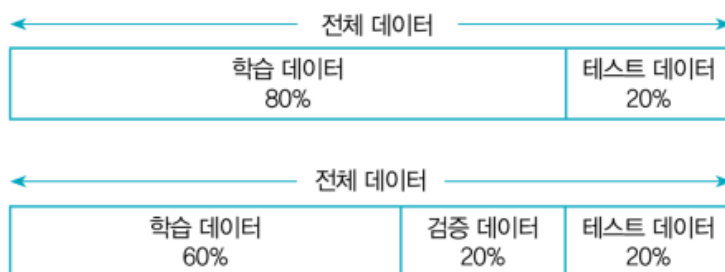


TIP _홀드아웃에서는 데이터의 수가 적을 경우 각 데이터셋이 전체 데이터를 대표하지 못할 가능성이 큼니다.

① 홀드아웃 **

- 홀드아웃은 가장 보편적인 데이터 분할을 통한 검증 방법이다.
- 홀드아웃은 전체 데이터를 랜덤하게 추출해 학습 데이터와 테스트 데이터로 분리하는 방식이다.
- 일반적으로 학습 데이터는 80%, 테스트 데이터는 20%, 혹은 학습 데이터 60%, 검증 데이터 20%, 테스트 데이터 20%가 되게 데이터를 분할한다. 검증 세트를 이용하여 다른 하이퍼파라미터 값에서 모델을 훈련하는 것을 계속 반복하고 성능을 평가한 뒤, 만족할 만한 성능이 나온 하이퍼파라미터를 이용하여 테스트 세트에서 모델의 일반화 성능을 추정한다.

【홀드아웃】



【오분류표 & 평가지표】***

		예측집단		합계
		Positive 1(True)	Negative 0(False)	
실제집단	Positive 1(True)	TP (Correct)	FN (Incorrect)	민감도(재현율) $\frac{TP}{TP+FN}$
	Negative 0(False)	FP (Incorrect)	TN (Correct)	특이도 $\frac{TN}{FP+TN}$
오분류율		정밀도	(없음)	정분류율
$\frac{FN+FP}{TP+FN+FP+TN}$		$\frac{TP}{TP+FP}$		$\frac{TP+TN}{TP+FN+FP+TN}$

```
: # 학습에 사용할 데이터와 분류값을 나눕니다.
df_x = df[['email', 'title']] # 사용할 데이터
df_y = df[['label']] # 정답
```

베르누이 나이브베이지의 입력 데이터는 고정된 크기의 벡터로써, 0과 1로 구분된 데이터이어야 합니다.
 sklearn의 CountVectorizer를 사용하여 쉽게 구현할 수 있습니다.
 CountVectorizer는 입력된 데이터(6개의 이메일)에 출현된 모든 단어의 갯수만큼의 크기의 벡터를 만든 후,
 각각의 이메일을 그 고정된 벡터로 표현합니다.
 binary=True를 파라미터를 넘겨줌으로써, 각각의 이메일마다 단어가 한번 이상 출현하면 1, 출현하지 않을 경우 0으로 표시하게 합니다.

```
: cv = CountVectorizer(binary=True) # binary - 중복되도 표시는 '1'로 표시하라
x_traincv = cv.fit_transform(df_x) # fit_transform - 사용하고자 하는 알고리즘으로 변형시켜줄,
x_traincv
```

```
: <6x17 sparse matrix of type '<class 'numpy.int64'>'
   with 23 stored elements in Compressed Sparse Row format>
```

```
: encoded_input = x_traincv.toarray()
encoded_input
```

```
: array([[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],
        [0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0],
        [1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
        [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1]], dtype=int64)
```

```
cv.inverse_transform(encoded_input[[0]]) # array(['free', 'game', 'only', 'today'])
(array(['free', 'game', 'only', 'today'], dtype='<U9'))
```

```
cv.get_feature_names()
```

C:\ProgramData\Anaconda3\envs\tf_cpu\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprecated: get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
 warnings.warn(msg, category=FutureWarning)

```
['attached',
 'card',
 'cheapest',
 'credit',
 'deal',
 'flight',
 'free',
 'game',
 'limited',
 'meeting',
 'offer',
 'only',
 'schedule',
 'statement',
 'time',
 'today',
 'your']
```

베르누이 나이브베이즈 분류 모델 생성

- 스팸 메일 분류
- BernoulliNB는 기본적으로 스무딩을 지원하므로, 학습데이터에 없는 단어가 테스트에 출현해도 분류를 이상없이 진행합니다.

```
# 학습 데이터로 베르누이 분류기를 학습합니다.
```

```
bnb = BernoulliNB()
y_train = df_y.astype('int')

bnb.fit(x_traincv, y_train)
```

```
BernoulliNB()
```

```
: test_email_list = [
    {'email title': 'free flight offer', 'spam': True},
    {'email title': 'hey traveler free flight deal', 'spam': True},
    {'email title': 'limited free game offer', 'spam': True},
    {'email title': 'today flight schedule', 'spam': False},
    {'email title': 'your credit card attached', 'spam': False},
    {'email title': 'free credit card offer only today', 'spam': False}
]
```

```
test_df = pd.DataFrame(test_email_list)
```

```
test_df['label'] = test_df['spam'].map({True:1,False:0})
```

```
test_x = test_df["email title"]
```

```
test_y = test_df["label"]
```

```
x_testcv = cv.transform(test_x)
```

```
: # 테스트
```

```
predict = bnb.predict(x_testcv)
predict
```

```
: array([1, 1, 1, 0, 0, 1])
```

```
: # 정확도
```

```
accuracy_score(test_y, predict)
```

```
: 0.8333333333333334
```

MultinomialNaiveBayes

```

]: # 라이브러리 임포트 : 실습에 필요한 라이브러리를 임포트
import numpy as np
import pandas as pd

# 다항분포 나이브베이지를 위한 라이브러리를 임포트합니다.
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score

np.random.seed(5)

```

문제 정의

- 다항분포 나이브베이지 분류 모델(MultinomialNB)을 사용하여 간단한 영화리뷰 분류 실습을 위해 아래 영화 리뷰 분류 실습을 위해 아래 영화 리뷰와 함께 영화에 대한 평가(긍정적/부정적) 정보가 있는 데이터를 사용하겠습니다.

```

]: review_list = [
    {'movie_review': 'this is great great movie. I will watch again', 'type': 'positive'},
    {'movie_review': 'I like this movie', 'type': 'positive'},
    {'movie_review': 'amazing movie in this year', 'type': 'positive'},
    {'movie_review': 'cool my boyfriend also said the movie is cool', 'type': 'positive'},
    {'movie_review': 'awesome of the awesome movie ever', 'type': 'positive'},
    {'movie_review': 'shame I wasted money and time', 'type': 'negative'},
    {'movie_review': 'regret on this move. I will never never what movie from this director', 'type': 'negative'},
    {'movie_review': 'I do not like this movie', 'type': 'negative'},
    {'movie_review': 'I do not like actors in this movie', 'type': 'negative'},
    {'movie_review': 'boring boring sleeping movie', 'type': 'negative'}
]

test_feedback_list = [
    {'movie_review': 'great great great movie ever', 'type': 'positive'},
    {'movie_review': 'I like this amazing movie', 'type': 'positive'},
    {'movie_review': 'my boyfriend said great movie ever', 'type': 'positive'},
    {'movie_review': 'cool cool cool', 'type': 'positive'},
    {'movie_review': 'awesome boyfriend said cool movie ever', 'type': 'positive'},
    {'movie_review': 'shame shame shame', 'type': 'negative'},
    {'movie_review': 'awesome director shame movie boring movie', 'type': 'negative'},
    {'movie_review': 'do not like this movie', 'type': 'negative'},
    {'movie_review': 'I do not like this boring movie', 'type': 'negative'},
    {'movie_review': 'aweful terrible boring movie', 'type': 'negative'}
]

df = pd.DataFrame(review_list)
df

```

```

]:

```

	movie_review	type
0	this is great great movie. I will watch again	positive
1	I like this movie	positive
2	amazing movie in this year	positive
3	cool my boyfriend also said the movie is cool	positive
4	awesome of the awesome movie ever	positive
5	shame I wasted money and time	negative
6	regret on this move. I will never never what m...	negative
7	I do not like this movie	negative
8	I do not like actors in this movie	negative
9	boring boring sleeping movie	negative

```

: # target -> 숫자로 매핑
df['label'] = df['type'].map({'positive':1, 'negative':0})
df

```

```

:

```

	movie_review	type	label
0	this is great great movie. I will watch again	positive	1
1	I like this movie	positive	1
2	amazing movie in this year	positive	1
3	cool my boyfriend also said the movie is cool	positive	1
4	awesome of the awesome movie ever	positive	1
5	shame I wasted money and time	negative	0
6	regret on this move. I will never never what m...	negative	0
7	I do not like this movie	negative	0
8	I do not like actors in this movie	negative	0
9	boring boring sleeping movie	negative	0

```
] : # 학습을 위해, 학습에 사용할 특징값과 분류값을 분리합니다.  
df_x = df['movie_review']  
df_y = df['label']
```

```
] : cv = CountVectorizer()  
x_traincv = cv.fit_transform(df_x)
```

```
] : encoded_input = x_traincv.toarray()  
encoded_input
```

```
] : array([[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 0, 1, 0, 0,  
          0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0],  
          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,  
          0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],  
          [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,  
          0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1],  
          [0, 0, 1, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,  
          0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],  
          [0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,  
          0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],  
          [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,  
          0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0],  
          [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 2,  
          0, 0, 1, 1, 0, 0, 0, 0, 2, 0, 0, 0, 1, 1, 0],  
          [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,  
          1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],  
          [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0,  
          1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],  
          [0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,  
          0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)
```

```
] : cv.inverse_transform(encoded_input[[0]])
```

```
] : [array(['again', 'great', 'is', 'movie', 'this', 'watch', 'will'],  
          dtype='<U9')]
```

```
] : cv.get_feature_names()
C:\ProgramData\Anaconda3\envs\tf_cpu\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_names is deprecated: get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)

] : ['actors',
      'again',
      'also',
      'amazing',
      'and',
      'awesome',
      'boring',
      'boyfriend',
      'cool',
      'director',
      'do',
      'ever',
      'from',
      'great',
      'in',
      'is',
      'like',
      'money',
      'move',
      'movie',
      'my',
      'never',
      'not',
      'of',
      'on',
      'regret',
      'said',
      'shame',
      'sleeping',
      'the',
      'this',
      'time',
      'wasted',
      'watch',
      'what',
      'will',
      'year']
```
