



Day52; 20221118

📅 날짜	@2022년 11월 18일
👤 유형	@2022년 11월 18일
☰ 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
Contributor 1 Issues 0 Stars 0 Forks 0

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4e13fdc9-be0a-42a9-9afb-1b971f0e84d1/02_%EB%A8%B8%EC%8B%A0%EB%9F%A%C%EB%8B%9D_%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e1f27dad-aed4-407c-89fb-fd5dec2f7e7f/05-3_Multinomial_Nave_Bayes_20221117.ipynb

화자

명사

- 무늬가 있는 도자기.

명사

- 중국의 문자. 곧, 한문자.

명사

- = 목화(木靴).

명사

- 죽은 사람.
- 변화하는 모든 것.

명사

- = 고자(鼓子).
- 조선 시대에 명(明)나라에 보내던 환관(宦官) 후보자.

명사

- 말을 하는 사람. ↔ 청자(聽者).

김천국曰 어떤 데이터를 어떤 알고리즘에 적용할지에 궁금증을 갖는 태도가 아주 중요하다.

다항 분포 나이브베이즈 분류

-다항분포 나이브베이즈로 영화 리뷰를 긍정적 평가인지, 부정적 평가인지 분류해보도록 하겠습니다.. MultinomialNB는 기본적으로 스무딩을 지원하므로, 학습데이터가 없는 단어가 테스트에 출현해도 분류를 이상없이 진행합니다.

```
] : # 기준의 데이터로 학습을 진행합니다.  
mnb = MultinomialNB()  
y_train = df_y.astype('int')  
mnb.fit(x_traincv, y_train)  
:] MultinomialNB()
```

```

: 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'}
]

test_df = pd.DataFrame(test_feedback_list)
test_df['label'] = test_df['type'].map({'positive':1, 'negative':0})

test_x = test_df['movie_review']
test_y = test_df['label']

```

```

cv = CountVectorizer()
x_traincv = cv.fit_transform(df_x)

```

```

# 테스트 진행
x_textcv = cv.transform(test_x)

```

fit()동작을 수행한 후 transform()한다는 게 fit_transform()이다.

그냥 transform()은 말 그대로...

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/41f555d8-cda-c-4090-b058-13c12231b9f2/20221118_095039.mp4

```

0]: # 테스트 진행
x_testcv = cv.transform(test_x)
predictions = mnb.predict(x_testcv)

```

```

1]: # 정확도(accuracy)
accuracy_score(test_y, predictions) # 정답과 예측값

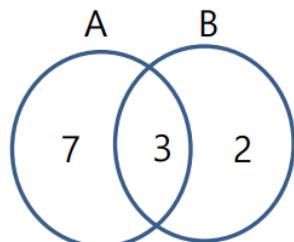
```

1]: 1.0

머신러닝은 완전히 100% 기계가 찾는 것은 아니다, 인간이 아이디어를 낸 이론을 가지고 만든 것이다.

사이킷런을 이용하여.

조건부 확률



- 사건이 발생한 횟수 : 12
- A 사건이 발생한 횟수 : $7 + 3 = 10$
- B 사건이 발생한 횟수 : $3 + 2 = 5$
- A 사건과 B 사건이 동시에 일어난 횟수 : 3

1. A 사건이 일어난 확률

$$- P(A) = 10 / 12$$

2. B 사건이 일어난 확률

$$- P(B) = 5 / 12$$

3. B 사건이 일어났을 때 A 사건이 일어날 확률

$$- P(A|B) = 3 / (3 + 2) = \textcolor{red}{0.6}$$

4. A 사건이 일어났을 때 B 사건이 일어날 확률

$$- P(B|A) = 3 / (7 + 3) = 0.3$$

5. B 사건이 일어났을 때 A 사건이 일어날 확률을 다른 방식으로 계산하는 방법

$$- P(A|B) = P(B|A) * P(A) / P(B) = 0.3 * (10/12) / (5/12) = \textcolor{red}{0.6}$$

즉, 어떤 데이터가 있을 때 그에 해당하는 레이블은 기존 데이터의 특징 및 레이블의 확률을 사용해 구할 수 있다는 것.

나이브 베이즈 알고리즘을 머신러닝에 응용하기

➤ A를 레이블, B를 데이터의 특징으로 대입해 보면

$$- P(\text{레이블} | \text{데이터 특징}) = P(\text{데이터 특징} | \text{레이블}) * P(\text{레이블}) / P(\text{데이터 특징})$$

나이브 베이즈 – 장/단점

➤ 장점

- 모든 데이터의 특징이 독립적인 사건이라는 나이브 가정에도 불구하고 실전에서 높은 정확도를 보이며, 특히 문서 분류 및 스팸 메일 분류에 강한 면모를 보임.
- 나이브 가정에 의해 계산 속도가 다른 모델들에 비해 상당히 빠름.

➤ 단점

- 모든 데이터의 특징을 독립적인 사건이라고 가정하는 것은 문서 분류에 적합할지는 모르나, 다른 분류 모델에는 제약이 될 수 있음.

앙상블(Ensemble) - 배깅(bagging), 부스팅(Boosting)

앙상블은 알고리즘이지만 알고리즘이 아니다. 알고리즘들을 장점을 취해 잘 버무려서 사용하는 것이 앙상블이다. 앙상블은 방법이다.

[국어사전]

앙상블 ([프랑스어]ensemble) ◻)

- 1 전체적인 어울림이나 통일.
- 2 드레스와 코트, 스커트와 재킷 따위를 같은 천으로 만들어서 서로 잘 어울리는 한 별의 여성복.
- 3 2인 이상이 하는 노래나 연주.

[국어사전 다른 뜻 1](#)

[영어사전]

앙상블

- 1 (전체적인 조화) ensemble
- 2 (합주단) ensemble

[영어사전 다른 뜻 1](#)

ensemble ★ +

1. 명사 (소규모의) 합주단[무용단/극단], 앙상블
2. 명사 앙상블(한 벌로 맞춰 입게 지은 옷)
3. 명사 격식 총체(總體)

미국식 [ən'sa:mbl] 영국식 [ən'sombl]

배깅(bagging)

어원 : 부트스트랩(bootstrap) + 어그리게이팅(aggregating)

앙상블(Ensemble) – 배깅(bagging)

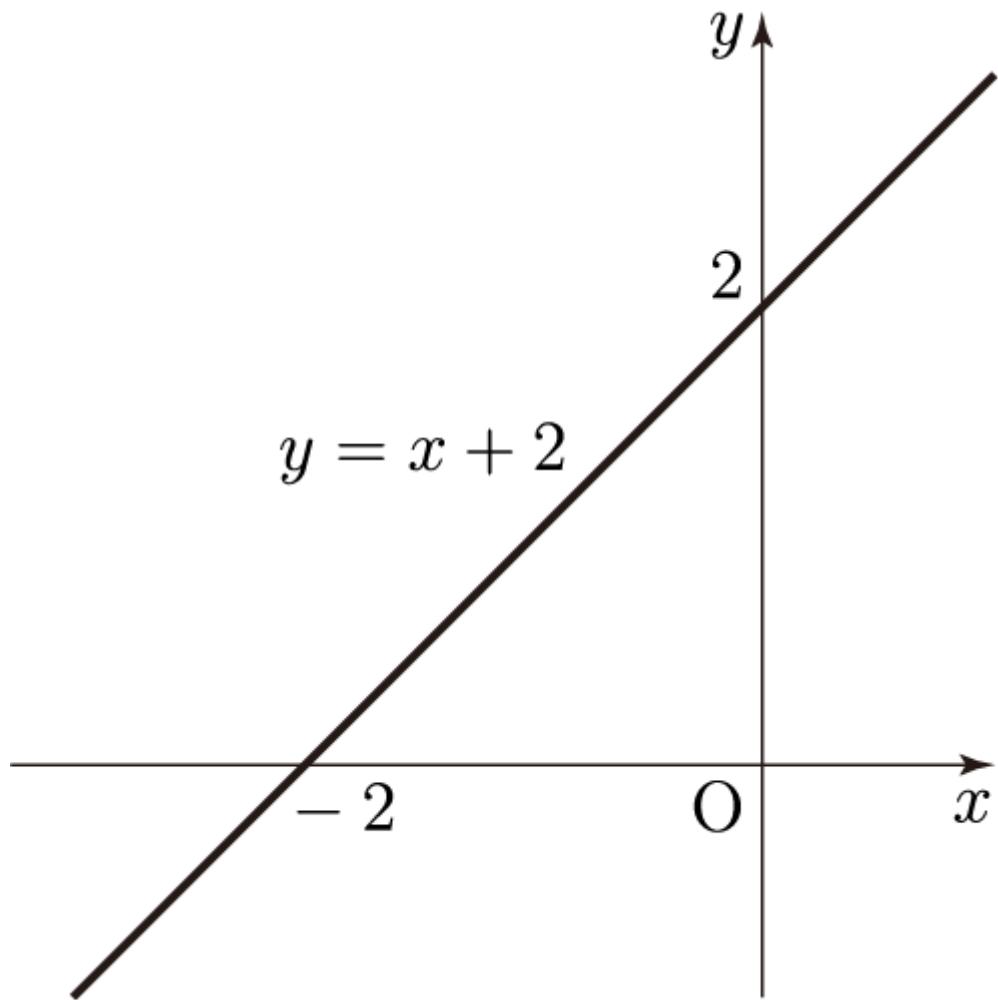
- 여러 개의 분류 모델을 조합해서 더 나은 성능을 내는 방법.
- 배깅(bagging)
 - 한 가지 분류 모델을 여러 개 만들어서 서로 다른 학습 데이터로 학습시킨 후(부트스트랩), 동일한 테스트 데이터에 대한 서로 다른 예측값들을 투표를 통해(aggregating) 가장 높은 예측값으로 최종 결론을 내리는 앙상블 기법.
 - 어원 : 부트스트랩(bootstrap) + 어그리게이팅(aggregating)
 - 과대적합이 쉬운 모델에 상당히 적합한 앙상블.

부트스트랩(Bootstrap)

- 데이터를 조금은 편향되도록 샘플링하는 기법.

예) $y(\text{치역}) = x(\text{독립변수, 정의(구)역}) + 3(\text{편향})$

예) $y(\text{치역}) = x(\text{독립변수, 정의(구)역}) - 3(\text{편향})$



위 아래로 편향에 의해 움직임(기울기 아니고)

랜덤 포레스트(RandomForest)

➤ 장점

- 양상블 효과로 의사결정 트리의 과대적합 단점을 보완한다.

➤ 단점

- 조정해야 할 하이퍼 파라미터가 많다.

편향

[1.....1]

[2.....2]

[3.....3]

[4.....4]

[5.....5]

[6.....6]

예) - 전체 데이터 : [1, 2, 3, 4, 5, 6, 7, 8, 9]

- 의사결정 트리 6개를 배깅할 경우

- 의사결정 트리1의 학습 데이터 : [1, 2, 3, 4, 5, 1]
- 의사결정 트리2의 학습 데이터 : [2, 3, 4, 5, 1, 2]
- 의사결정 트리3의 학습 데이터 : [3, 4, 5, 6, 7, 3]
- 의사결정 트리4의 학습 데이터 : [4, 5, 6, 7, 8, 4]
- 의사결정 트리5의 학습 데이터 : [5, 6, 7, 8, 9, 5]
- 의사결정 트리6의 학습 데이터 : [6, 7, 8, 9, 1, 6]

- 각 분류 모델은 총 N개의 데이터보다 적은 데이터로 학습.
- 중복된 데이터를 허용함으로써 편향이 높은 학습 데이터로 학습.
- 데이터 샘플링 크기는 보통 전체 데이터의 60~70%를 사용.

회귀분석은 $y(\text{종속변수}) = w(\text{기울기})x(\text{독립변수}) + b$ [1차식]

선형적이다.

weighted(=biased)

weight·ed

발음 미국·영국 ['weɪtɪd] 미국식 ['weɪtrɪd]



옥스퍼드

동아출판

YBM

교학사

슈프림

형용사 | 학습 정보 | 영영사전

형용사

1. [명사 앞에는 안 씀]

(한쪽에 유·불리하게) 치우친, 편중된

The proposal is **weighted** towards smaller businesses. ⓘ ⓘ
 그 제안은 소규모 기업 쪽으로 치우쳐 있다.

문형 ~ towards/against sb/sth | ~ in favour of sb/sth

유의어 biased

Aggregating

- 여러 분류 모델이 예측한 값들을 조합해서 하나의 결론을 도출하는 과정.
- 결론은 투표를 통해 결정.

- 하드 보팅(Hard Voting)
 - 배경에 포함된 K개의 분류 모델에서 최대 득표를 받은 예측값으로 결론을 도출.
 - ex) 손글씨 숫자를 보고, 숫자를 1~9로 분류하는 의사결정 트리 6개를 학습한 후, 숫자 7을 보여줬을 경우 다음과 같이 각각 분류했다고 가정.

분류모델	분류값
의사결정 트리1	1
의사결정 트리2	2
의사결정 트리3	7
의사결정 트리4	7
의사결정 트리5	7
의사결정 트리6	7

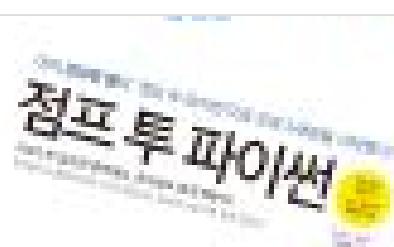
- 총 6개의 투표 중, 7이 4개의 최다 득표를 받아 하드보트 어그리게이팅은 7로 입력값을 정확히 예측.

mnist

점프 투 파이썬

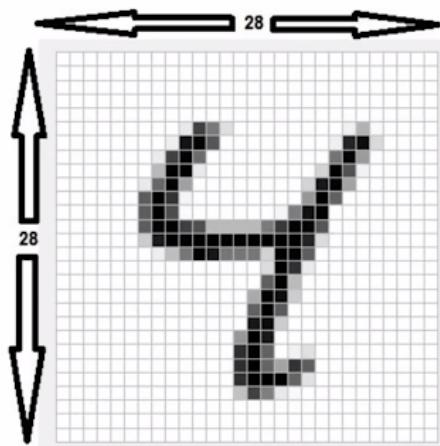
점프 투 파이썬 오프라인 책(개정판) 출간 !! (2019.06) *** [책 구입 안내](<https://wikidocs.net/4321>) 이 책은 파이썬이란 ...

 <https://wikidocs.net/60324>



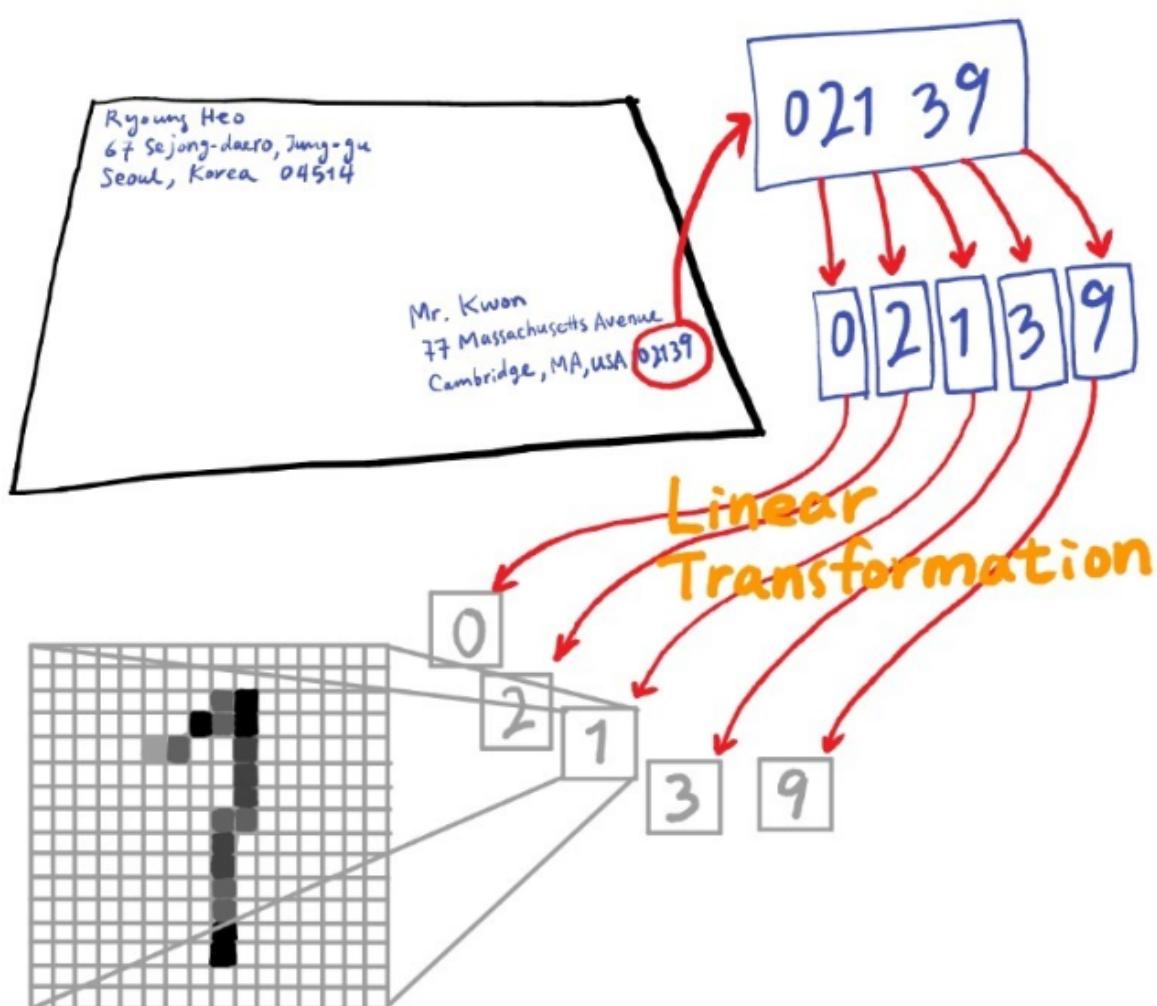
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

우선 MNIST 문제를 더 자세히 보겠습니다. 각각의 이미지는 아래와 같이 28 픽셀 × 28 픽셀의 이미지입니다.



이 문제를 풀기 위해 여기서는 28 픽셀 × 28 픽셀 = 784 픽셀이므로, 각 이미지를 총 784의 원소를 가진 벡터로 만들어줄겁니다. 이렇게 되면 총 784개의 특성을 가진 샘플이 되는데, 이는 앞서 우리가 풀었던 그 어떤 문제들보다 특성이 굉장히 많은 샘플입니다.

우편번호 손글씨 인식



딥러닝의 Hello World, MNIST 데이터셋

야만인] 인공지능 탄생의 뒷이야기 | 머신러닝의 Hello World가 iris 데이터셋이라면 딥러닝에서는 MNIST(Modified National Institute of Standards and Technology Database)입니다. MNIST는 손으로 쓴

↳ <https://brunch.co.kr/@hvnpoet/96>

6	3	8	8	0	1	5	4	1	5
2	1	9	8	0	3	3	6	4	1
7	9	1	4	9	9	2	4	5	1
3	7	3	9	3	6	7	2	4	3
3	5	1	9	7	4	4	3	4	9
0	1	6	0	5	2	8	8	5	7

Aggregating

➤ 소프트 보팅(Soft Voting)

- 하드 보팅보다 더 정교한 투표 방식.
- 하드 보팅 같은 경우 각 분류 모델은 최고의 확률을 갖는 분류값만을 aggregating할 때 리턴하는 반면, 소프트 보팅은 모든 분류값의 확률을 리턴함.
- 따라서 하드 보팅은 단순히 가장 많은 투표를 받은 분류값을 단순히 aggregatin의 결론으로 도출하는 반면, 소프트 보팅은 각 분류값별 확률을 더해준 값을 점수로 사용해 최대 점수를 가진 분류값을 결론으로 도출함.

- ex)

분류모델	1	2	3	4	5	6	7	8	9
의사결정 트리1	0.9	0.1	0	0	0	0	0	0	0
의사결정 트리2	0	0.8	0.1	0.1	0	0	0	0	0
의사결정 트리3	0	0	0.1	0	0	0	0.9	0	0
의사결정 트리4	0	0	0	0.1	0	0	0.9	0	0
의사결정 트리5	0	0	0	0	0	0	1	0	0
의사결정 트리6	0.4	0	0	0	0	0	0.6	0	0

- 숫자 7이 입력됐을 때 각 의사결정 트리의 분류값별 확률

분류값	확률값	최종점수
1	0.9+0.4	1.3
2	0.1+0.8	0.9
3	0.1+0.1	0.2
4	0.1+0.1	0.2
5	0	0
6	0	0
7	0.9+0.9+1+0.6	3.4
8	0	0
9	0	0

- 소프트 보팅의 각 분류값별 점수 계산

이러한 과정에서 램덤 포레스트는 또 한 번의 모델의 편향을 증가시켜 과대적합의 위험을 감소시킴.

랜덤 포레스트(RandomForest)

- 여러 의사결정 트리를 배깅해서 예측을 실행하는 모델.
- 배깅이 모든 분류 모델에 적용 가능하지만, 특히 과대적합되기 쉬운 의사결정 트리에 적용하면 확실히 과대적합을 줄여 성능이 높아지는 혜택을 보기 때문에 배깅은 많은 의사결정 트리 모델의 개선을 이뤘고, 여러 개의 나무들이 모여 있다는 개념에서 랜덤 프레스트라는 이름이 생겨남.
- 의사결정 트리에서는 최적의 특징으로 트리를 분기하는 반면, 랜덤 포레스트는 각 노드에 주어진 데이터를 샘플링해서 일부 데이터를 제외한 채 최적의 특징을 찾아 트리를 분기함.
- 이러한 과정에서 랜덤 포레스트는 또 한 번의 모델의 편향을 증가시켜 과대적합의 위험을 감소시킴.

랜덤 포레스트(RandomForest)

분류모델	분류값
의사결정 트리1	1
의사결정 트리2	2
의사결정 트리3	7
의사결정 트리4	7
의사결정 트리5	7
의사결정 트리6	7

분류값	확률값	최종점수
1	0.9+0.4	1.3
2	0.1+0.8	0.9
3	0.1+0.1	0.2
4	0.1+0.1	0.2
5	0	0
6	0	0
7	0.9+0.9+1+0.6	3.4
8	0	0
9	0	0

소프트 보팅의 각 분류값별 점수 계산

배깅이 모든 분류 모델에 적용 가능하지만,
한 번의 모델의 편향을 증가

의사결정 트리 Tree + Tree + Tree = 랜덤포레스트(RandomForest)

랜덤 포레스트(RandomForest)

➤ 장점

- 양상블 효과로 의사결정 트리의 과대적합 단점을 보완한다.

➤ 단점

- 조정해야 할 하이퍼 파라미터가 많다.

DAsP)

결합도와 응집도는 무엇일까?

결합도(Coupling)과 응집도(Cohesion)에 대해서 알아보기 전에 모듈(Module)과 모듈화(Modularization)에 대해 먼저 알아볼 필요가 있다. 모듈화란 소프트웨어를 각 기능별로 나누는 것을 말한다. 그리고

📎 <https://madplay.github.io/post/coupling-and-cohesion-in-software-engineering>



오늘도
MadPlay!

06-1_RandomForest_20221118

```
: # 라이브러리 임포트 : 실습에 필요한 라이브러리를 임포트
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score

np.random.seed(5)
```

```
: # 손글씨 데이터 로드
mnist = datasets.load_digits()
features, labels = mnist.data, mnist.target
```

교차검증

- 교차 검증을 10번 수행하여, 10번의 교차 검증 평균 정확도를 저장합니다.

```
: def cross_validation(classifier, features, labels):
    cv_scores = []

    for i in range(10):
        scores = cross_val_score(classifier, features, labels, cv=10, scoring='accuracy')
        # classifier는 DecisionTreeClassifier, RandomForestClassifier를 넣어서 비교를 해보겠다는 것
        # cv=10 10등분 해보겠다는 것, 훈련데이터가 적을 때 활용
        cv_scores.append(scores.mean())

    return cv_scores
```

```
?7]: dt_cv_scores = cross_validation(DecisionTreeClassifier(), features, labels)
```

```
?8]: rf_cv_scores = cross_validation(RandomForestClassifier(), features, labels)
```

랜덤포레스트(RandomForest) vs 의사결정트리(DecisionTree) 시작화

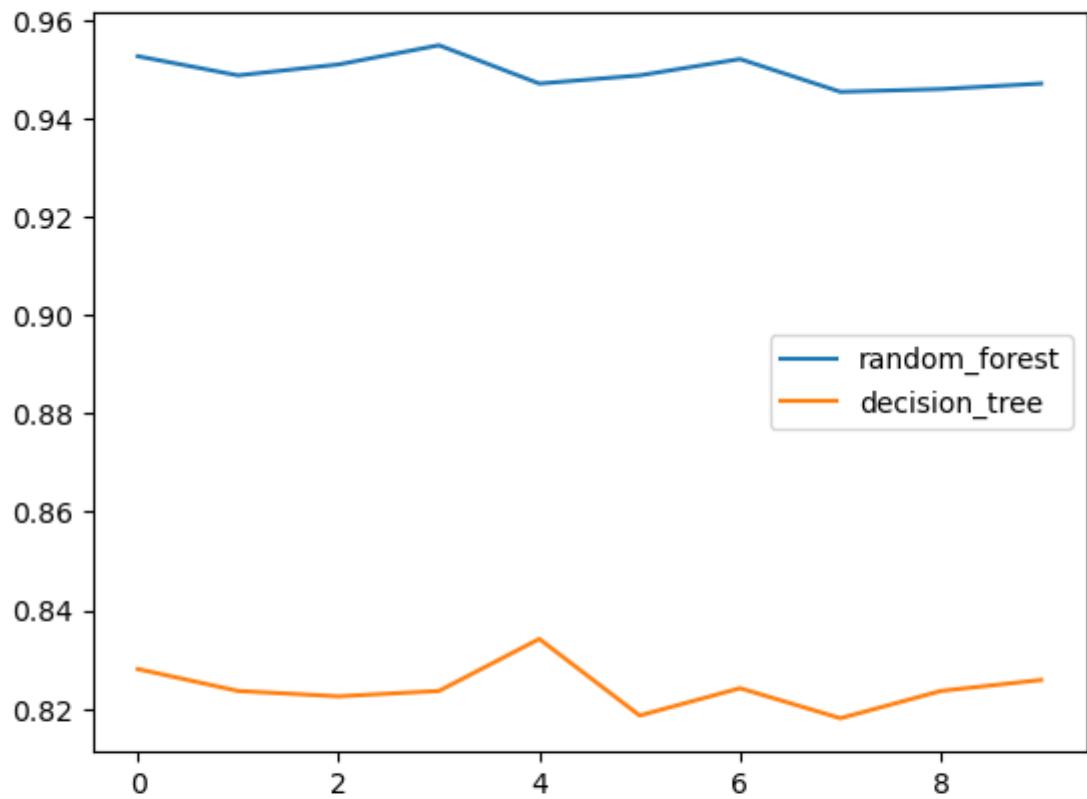
- 라인 차트를 통해 랜덤포레스트가 의사결정트리보다 매번 정확도가 높은 것을 확인할 수 있습니다.

```
cv_list = [
    ['random_forest', rf_cv_scores],
    ['decision_tree', dt_cv_scores]
]

df = pd.DataFrame.from_dict(dict(cv_list))
```

```
] df.plot()
```

```
] <AxesSubplot:>
```



```
] # 의사결정트리 정확도  
np.mean(dt_cv_scores)
```

```
] 0.8241843575418993
```

```
] # 랜덤포레스트 정확도  
np.mean(rf_cv_scores)
```

```
] 0.9494096834264433
```

보팅 양상률

- 단일 모델을 양상을하여 더 나은 예측을 하는 양상을 모델을 만들어 보겠습니다.

```
9]: from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

import numpy as np
np.random.seed(5)

0]: # 손글씨 데이터 로드

mnist = datasets.load_digits()
features, labels = mnist.data, mnist.target

X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2)
```

단일 모델 정확도 측정

- 의사결정트리, knn, svm 모델의 정확도를 측정.

```
31]: dtree = DecisionTreeClassifier(criterion='gini', max_depth=8, max_features=32, random_state=35)
dtree = dtree.fit(X_train, y_train)
dtree_predicted = dtree.predict(X_test)

32]: knn = KNeighborsClassifier(n_neighbors=299).fit(X_train, y_train) # n_neighbors값은 gridsearchcv로 찾아야 함.
knn_predicted = knn.predict(X_test)

33]: svm = SVC(C=0.1, gamma=0.003, probability=True, random_state=35).fit(X_train, y_train) # 전처리로 적합한 값을 찾아야 함.
svm_predicted = svm.predict(X_test)

34]: print('[accuracy]')
print('dtree : ', accuracy_score(y_test, dtree_predicted))
print('knn : ', accuracy_score(y_test, knn_predicted))
print('svm : ', accuracy_score(y_test, svm_predicted))

[accuracy]
dtree : 0.8277777777777777
knn : 0.8944444444444445
svm : 0.8916666666666667

35]: # 직접 소프트보팅을 구현하실 때는 predict_proba() 함수를 사용하여 투표를 수행 시 확률 분류값을 확률을 사용하시면 됩니다.

svm_proba = svm.predict_proba(X_test)
print(svm_proba[0:2]) # 예측값을 2번 블.

[[0.00129293 0.00434548 0.00446459 0.00248053 0.00336731 0.93848315
 0.00127073 0.00448341 0.02477231 0.01503956]
 [0.00189233 0.00631647 0.92958418 0.00345522 0.0030971 0.00828454
 0.00186869 0.0048112 0.03092707 0.0097632]]
```

하드 보팅

- 하드 보팅은 일반적인 투표와 같이, 각각의 분류기의 예측값들을 모아, 가장 많은 득표를 받은 예측값으로 최종 결론을 내는 방식입니다.

```
36]: voting_clf = VotingClassifier(estimators=[('decision_tree', dtree), ('knn', knn), ('svm', svm)],
                                    weights=[1,1,1], voting='hard').fit(X_train, y_train)
# estimators - 평가 모델 # weights - 가중치
hard_voting_predicted = voting_clf.predict(X_test)
accuracy_score(y_test, hard_voting_predicted) # 0.9222222222222223 # 여러 모델을 가지고 데이터를 내니까 좋아지기도 하더라.

36]: 0.9222222222222223
```

소프트 보팅

- 소프트 보팅은 각각의 분류 모델의 predict_proba를 활용하여, 모든 분류값들의 확률들을 더해서, 가장 높은 점수를 획득한 분류값으로 최종 결론을 내는 방식입니다.

```
37]: voting_clf = VotingClassifier(estimators=[('decision_tree', dtree), ('knn', knn), ('svm', svm)],
                                    weights=[1,1,1], voting='soft').fit(X_train, y_train)
# estimators - 평가 모델 # weights - 가중치
soft_voting_predicted = voting_clf.predict(X_test)
accuracy_score(y_test, soft_voting_predicted)

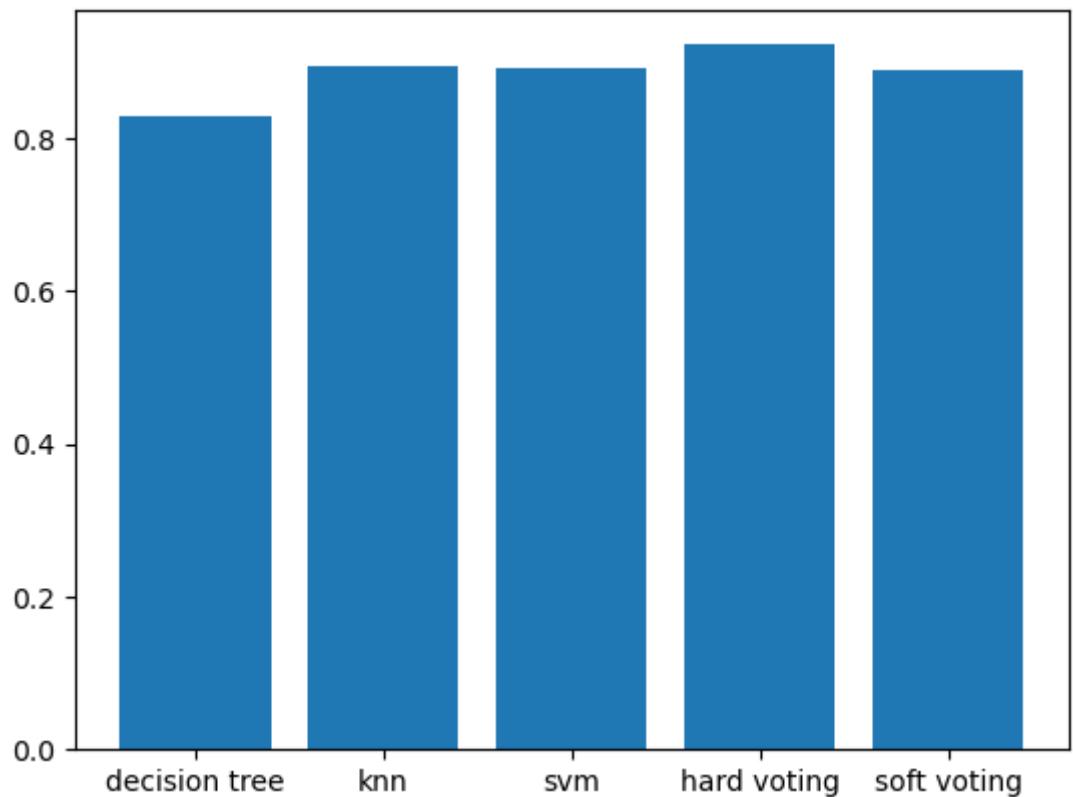
37]: 0.8888888888888888
```

정확도 비교 시각화

```
: import matplotlib.pyplot as plt

x = np.arange(5)
plt.bar(x, height = [accuracy_score(y_test, dtree_predicted),
                      accuracy_score(y_test, knn_predicted),
                      accuracy_score(y_test, svm_predicted),
                      accuracy_score(y_test, hard_voting_predicted),
                      accuracy_score(y_test, soft_voting_predicted)])
plt.xticks(x, ['decision tree', 'knn', 'svm', 'hard voting', 'soft voting'])

: ([<matplotlib.axis.XTick at 0x1f19337f6c8>,
  <matplotlib.axis.XTick at 0x1f19337f088>,
  <matplotlib.axis.XTick at 0x1f1931ba3c8>,
  <matplotlib.axis.XTick at 0x1f193825e88>,
  <matplotlib.axis.XTick at 0x1f193843948>],
 [Text(0, 0, 'decision tree'),
  Text(1, 0, 'knn'),
  Text(2, 0, 'svm'),
  Text(3, 0, 'hard voting'),
  Text(4, 0, 'soft voting')])
```



단일 모델 정확도 측정

- 의사결정트리, knn, svm 모델의 정확도를 측정.

Aggregating

- 여러 분류 모델이 예측한 값들을 조합해서 하나의 결론을 도출하는 과정.
 - 결론은 투표를 통해 결정.
-
- 하드 보팅(Hard Voting)
 - 소프트 보팅(Soft Voting)

probability

미국식[| pra:bə | bɪləti] ⇔ 영국식[| probə | bɪləti] ⇔

- 1 개연성 (=likelihood)
- 2 개연성 있는 일[것]
- 3 확률

[영어사전 결과 더보기](#)

분석이 잘 나왔다고 바로 보고를 하면 그게 기준이 되어서 다음 프로젝트에서 힘들어진다.

random_state

안녕하십니까,

scikit learn에서 사용하는 random_state 인자는 수행시마다 동일한 결과를 얻기 위해 적용합니다.

train_test_split(..., test_size=0.2) 과 같은 함수는 80% train, 20% test 데이터 세트를 추출합니다. 하지만 추출된 데이터는 수행을 할때마다 다를수 있습니다. random하게 80%, 20%를 추출하기 때문입니다.

가령 1~100까지 일련번호로 된 100개의 데이터를 train_test_split(..., test_size=0.2)로 수행하면 해당 함수를 첫번째 수행할 때는 1~80 번이 train, 81~100번이 test가 될 수 있지만, 다시 수행하면 이번에 21~100번이 train, 1~20번이 test가 될 수 있습니다. 80%, 20%로 나누는건 동일하지만 함수를 수행 시마다 추출한 레코드들을 달라질수 있습니다. 내부적으로 80%, 20%로 나눌때 random 함수를 적용합니다.

random_state=1 이라고 하면 바로 이 random 함수의 seed 값을 고정시키기 때문에 여러번 수행하더라도 같은 레코드를 추출합니다. random 함수의 seed값을 random_state라고 생각하시면 됩니다.

제가 강의에 사용된 train/test 데이터세트를 여러분도 동일하게 사용할 수 있도록 random_state를 고정값으로 할당했습니다. 그렇지 않으면 제가 설명드리는 데이터 세트와 여러분이 수행하는 데이터 세트는 80%, 20%는 맞더라도 서로 다른 레코드로 추출되기 때문입니다.

random_state를 어떤 값으로 하셔도 상관없습니다. 이는 random값을 고정하는 역할만 수행합니다.

감사합니다.

부스팅

부스팅은 동일한 알고리즘의 분류기를 순차적으로 학습해서 여러 개의 분류기를 만든 후, 테스트할 때 가중 투표를 통해 예측값을 결정.
순차적 학습과 가중 투표가 부스팅의 가장 큰 특징.

부스팅 – 순차적 학습

- 가령, 인물 사진 안에 있는 인물을 보고, 남자 또는 여자로 분류하는 의사결정 트리를 부스팅할 경우, 먼저 첫 번째 의사결정 트리를 학습함.
 - 테스트 결과, 남자 분류가 미흡할 경우, 남자 학습 데이터를 보강한 후 두 번째 의사결정 트리를 학습함.
 - 그리고 두 번째 의사결정 트리의 테스트 결과에 따라 학습 데이터를 보강해서 세 번째 의사결정 트리를 학습함.
 - 이처럼 부스팅은 순차적으로 학습 데이터를 보강하며 동일한 알고리즘의 분류기를 여러 개 만드는 과정을 가짐.
-

한 가지 분류 모델을 여러 개 만들어서 서로 다른 학습 데이터로 학습시킨 후(부트스트랩), 동일한 테스트 데이터에 대한 서로 다른 예측값들을 투표를 통해(aggregating) 가장 높은 예측값으로 최종 결론을 내리는 양상을 기법.

부스팅 – 가중 투표

- 배깅은 마치 우리가 선거를 하듯, 동일한 한 표식 부여되는 반면 부스팅은 가중 투표가 진행됨.
- 가중 투표는 투표자의 능력치에 따라 한 표의 가치가 다른 투표임.
- 팀회식 장소를 결정하는데, 사원 5명이 클럽에 가자고 했지만 부장님 3명이 삼겹살을 먹으러 가자고 했을 때, 삼겹살로 회식이 결정되는 것과 같음.
- 3가지 분류기의 검증 결과, 정확도가 다음과 같다고 가정하면

	분류기1	분류기2	분류기3
정확도	0.4	0.5	0.95

- 세 분류기의 정확도

	분류기1	분류기2	분류기3
분류값	남자	남자	여자

- 세 분류기의 분류값(하드보팅)

- 이 경우 가중 투표는 다음과 같이 진행.

- 하드 보팅

- ✓ 남자 : 0.4 + 0.5, 여자 : 0.95

- ✓ 따라서 두 분류기가 남자라고 예측했음에도 가중 투표 결과, 사진의 인물을 여자로 분류.

- 소프트 보팅

- ✓ 남자: $0.4 \times 0.7 + 0.5 \times 0.8 + 0.95 \times 0.1 = 0.775$

- ✓ 여자: $0.4 \times 0.3 + 0.5 \times 0.2 + 0.95 \times 0.9 = 1.075$.

- ✓ 소프트 보팅의 결과, 여자로 최종 결론 도출.

	분류기1	분류기2	분류기3
분류값	남자: 0.7 여자: 0.3	남자: 0.8 여자: 0.2	남자: 0.1 여자: 0.9

- 세 분류기의 분류값(소프트보팅)

머신러닝

k-최근접 이웃(k-Nearest Neighbor, kNN)

서포트 벡터머신(Support Vector Machine, SVM)

의사결정 트리(Decision Tree)

나이브 베이즈(Naïve Bayes)

앙상블(Ensemble)

군집화(Clustering)

주성분 분석(Principal Component Analysis, PCA)

딥러닝

- 선형 회귀(Linear Regression)
- 로지스틱 회귀(Logistic Regression)
- 딥러닝(Deep Neural Network, DNN)

선형 회귀(Linear Regression)

로지스틱 회귀(Logistic Regression)

딥러닝(Deep Neural Network, DNN)