04. 인공지능 분석

소프트웨어융합대학
인공지능학부
이수미

목차

01

인공지능을 활용한 데이터 분석

01. 인공지능을 활용한 데이터 분석

I. 데이터의 특성과 인공지능 학습

- 인공지능으로 데이터를 분석하려면 수집한 데이터 양이 충분해야 하며, 알고리즘에 입력할 데이터를 적절한 방식으로 표현해야 함.
- 이때 수집된 데이터를 적절한 방식으로 표현하였는지가 인공지능 분석의 성패를 좌우하는데, 수집된 데이터를 가장 잘 설명할 수 있게 뽑은 항목을 특성(Feature)이라고 부름.

I. 데이터의 특성과 인공지능 학습

- 데이터는 특성으로 표현되며 특성으로 설명된 데이터를 인공지능 학습에 활용.
- 원시 데이터(Raw data) : 어떤 주제로 수집한 데이터 원본
- 학습 데이터(Training data): 문제에 맞게 데이터 특성을 정의하고 추출하여 재구성한 데이터
- 데이터 가공: 원시 데이터를 학습 데이터로 만드는 과정.

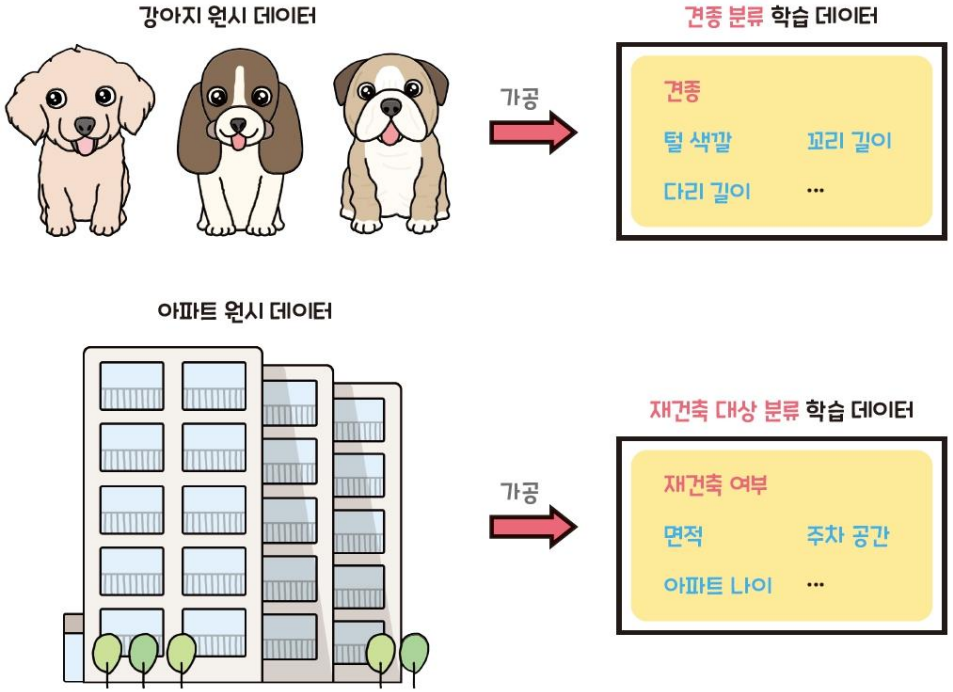


그림 11-1 분류 문제의 원시 데이터와 학습 데이터

I. 데이터의 특성과 인공지능 학습

- 인공지능 학습
  - 인공지능은 특성을 바탕으로 학습하여 모형(Model, 모델)을 생성. 학습(Learning)이란 수학적 알고리즘(틀)에 특성(재료)를 충분히 입력하여 일정한 모양의 모형을 만드는 과정.
  - [그림 11-2]는 이미지를 보고 고양이를 인식하는 인공지능이 학습하는 과정.

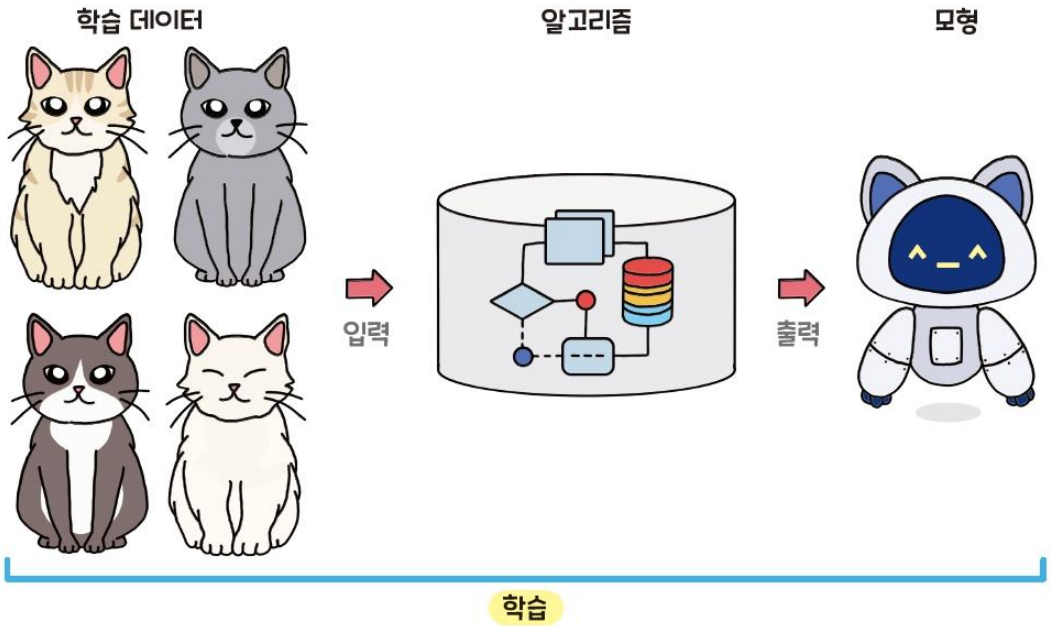


그림 11-2 인공지능 학습 과정

II. 분류 문제와 예측 문제

– 분류(Classification) 문제는 지도학습으로 해결할 수 있는 대표적인 문제.

그림 11-3 분류 문제 해결을 위한 인공지능 학습 과정

– [그림 11-3]은 강아지와 고양이를 분류하는 모형을 생성하는 과정.

– 학습할 때 데이터에서 어떤 특성을 선택하는지가 매우 중요.

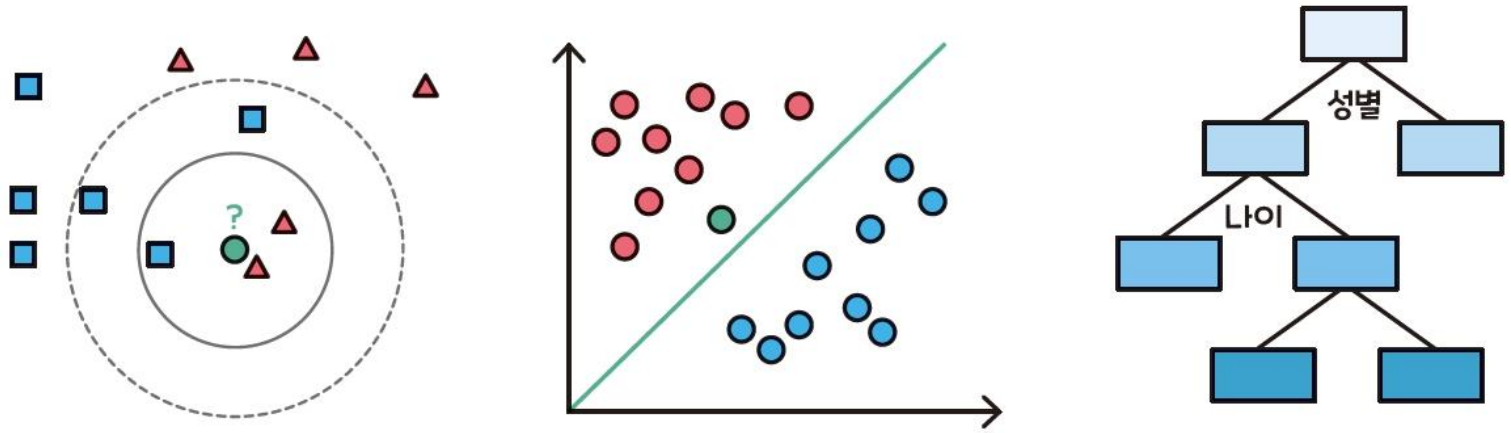
II. 분류 문제와 예측 문제

– 대표적인 분류 알고리즘

» K-NN 알고리즘: 데이터로부터 거리가 가까운 데이터 k개의 레이블을 참조하여 많은 수에 해당하는 클래스로 분류.

» SVM 알고리즘: 두 클래스를 구분하는 가상의 결정경계면(Hyperplane, 초평면)을 계산하여 클래스를 분류.

» 의사결정 트리: 스무고개를 하듯이 데이터를 나무 형태로 분류해 내려감.



(a) K-NN (b) SVM (C) 의사결정 트리

그림 11-4 지도학습 기반 분류 알고리즘

II. 분류 문제와 예측 문제

✓ **하나 더 알기**: 더 많은 분류 알고리즘

이외에도 의사결정 트리를 다중으로 엮어 만든 랜덤 포레스트(Random Forest), 베이즈 확률이론을 활용한 나이브 베이즈(Naïve-Bayes) 알고리즘 등이 널리 사용되고 있음.

또한 딥러닝에 속하는 DBM(Deep Boltzman Machine), DBN(Deep Belief Network), RNN, CNN 등 심층신경망 알고리즘이 분류 문제에 사용됨.

II. 분류 문제와 예측 문제

- 연속적인 값 예측을 위한 인공지능
  - 예측(Prediction) 문제는 분류 문제와 함께 지도학습으로 해결할 수 있는 대표적인 문제.
  - 예측 문제 해결에는 주로 회귀(Regression)를 활용하며 예측 모형은 가격이나 확률과 같이 연속적인 값(Continuous value)을 예측.
  - 예측 알고리즘은 선형 회귀와 로지스틱 회귀 외에도 리지(Ridge) 회귀, 라쏘(Lasso) 회귀, 다항(Polynomial) 회귀 등이 있음.

II. 분류 문제와 예측 문제

- 연속적인 값 예측을 위한 인공지능

그림 11-5 예측 문제 해결을 위한 인공지능 학습 과정

– [그림 11-5]는 고양이 데이터로부터 나이가 몇 개월인지 예측하는 모형을 생성하는 과정.

– 적절한 특성을 추출하여 모형을 학습시키면 매우 높은 확률로 새로운 고양이 데이터로부터 나이를 예측할 수 있음.

02

인공지능과 분류

02. 인공지능과 분류

I. 와인 경작자 분류

- 분류 모형을 생성하고 와인을 분류해보기.
- wine.DESCR를 출력하면 데이터에 대한 설명을 확인할 수 있음.

[코드 11-1] 와인 데이터 확인

```
from sklearn.datasets import load_wine
wine = load_wine( )
print(wine.DESCR)
```

```
.. _wine_dataset:
Wine recognition dataset
------------------------

**Data Set Characteristics:**

     :Number of Instances: 178
     :Number of Attributes: 13 numeric, predictive attributes and the class
     :Attribute Information:
            - Alcohol
            - Malic acid
            (중략)
            - OD280/OD315 of diluted wines
            - Proline
```

I. 와인 경작자 분류

[코드 11-2] 실행결과(계속)

```
- class:
        - class_0
        - class_1
        - class_2
 :Summary Statistics:

 ============================== ====== ===== ======= =====
                           Min   Max    Mean    SD
 ============================== ====== ===== ======= =====
 Alcohol:                  11.0  14.8    13.0  0.8
 Malic Acid:               0.74  5.80    2.34  1.12
 Ash:                      1.36  3.23    2.36  0.27
 Alcalinity of Ash:        10.6  30.0    19.5  3.3
 Magnesium:                70.0  162.0   99.7  14.3
 Total Phenols:                      0.98  3.88    2.29  0.63
 Flavanoids:               0.34  5.08    2.03  1.00
 Nonflavanoid Phenols:     0.13  0.66    0.36  0.12
 Proanthocyanins:          0.41  3.58    1.59  0.57
 Colour Intensity:         1.3   13.0    5.1    2.3
 Hue:                      0.48  1.71    0.96  0.23
 OD280/OD315 of diluted wines:   1.27  4.00    2.61  0.71
 Proline:                  278   1680    746    315
 ============================== ====== ===== ======= =====
```

I. 와인 경작자 분류

– 이 데이터를 인공지능에 학습시켜 와인의 경작자를 분류하는 모형을 생성합니다. 클래스가 3개이므로 이 모형은 다중분류(Multi-class) 문제를 해결.

[코드 11-2] 원시 데이터에서 학습 데이터로 변환

```
import pandas as pd
import numpy as np

wine_feature = wine.data
wine_label = wine.target

df_wine = pd.DataFrame(data=wine_feature, columns=[wine.feature_names])
df_wine['label'] = wine_label
df_wine
```

– wine.data는 사이킷런의 와인 데이터에서 특성 넘파이 배열을 반환함. wine.target은 사이킷런의 와인 데이터에서 레이블 넘파이 배열을 반환함.

– 가져온 특성 배열을 데이터프레임 df_wine으로 변환하고 label 열을 추가.

I. 와인 경작자 분류

[코드 11-2] 실행결과

	alcohol	malic_acid	ash	alcalinity_of_ash	magnes	r_intensity	hue	od280/od315_of_diluted_wines	proline	label
0	14.23	1.71	2.43	15.6	12	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	10	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	10	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	1	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	1	4.32	1.04	2.93	735.0	0
...	...	...	...	...		...	...	...	...	...
173	13.71	5.65	2.45	20.5		7.70	0.64	1.74	740.0	2
174	13.40	3.91	2.48	23.0	10	7.30	0.70	1.56	750.0	2
175	13.27	4.28	2.26	20.0	12	10.20	0.59	1.56	835.0	2
176	13.17	2.59	2.37	20.0	12	9.30	0.60	1.62	840.0	2
177	14.13	4.10	2.74	24.5		9.20	0.61	1.60	560.0	2

178 rows × 14 columns

I. 와인 경작자 분류

– 데이터셋을 학습과 테스트로 분할.
테스트 데이터는 학습 데이터에서 사용하지 않은 것으로 구성.

[코드 11-3] 테스트 데이터 분할

```
from sklearn.model_selection import train_test_split

df_wine = df_wine.astype({'label':'int'})
train, test = train_test_split(df_wine, test_size=0.3, random_state=0,
stratify=df_wine['label’])

train_X = train[train.columns[:13]]
train_Y = train[train.columns[13:]]

test_X = test[test.columns[:13]]
test_Y = test[test.columns[13:]]
```

– 데이터 프레임 df_wine을 입력으로 하여 데이터를 분할, 마지막 stratify 인자는 학습 데이터와 테스트 데이터에 각 레이블이 적절한 비율로 포함되도록 데이터를 구성.

– train과 test 데이터셋을 다시 특성과 레이블로 분리.

I. 와인 경작자 분류

[코드 11-4] K-NN 분류

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

#학습하기
model = KNeighborsClassifier()
model.fit(train_X, train_Y)

#테스트와 평가하기
pred_knn = model.predict(test_X)
print('KNN 알고리즘 분류 정확도:', metrics.accuracy_score(pred_knn, test_Y))
```

```
KNN 알고리즘 분류 정확도: 0.7222222222222222
```

- 모형이 K-NN 알고리즘으로 분류하도록 설정하고 모형이 학습할 데이터와 레이블을 설정하여 학습을 수행.
- 레이블이 없는 테스트 데이터셋을 분류하고, 결과를 pred_knn에 저장.
- 분류 결과인 pred_knn과 실제 레이블인 test_Y를 비교하여 출력.

I. 와인 경작자 분류

- 인공지능 모형의 학습에 앞서 데이터 전처리가 필수.
- 불필요하거나 학습에 방해되는 특성을 정리하는데 상관분석 기법이 활용됨.

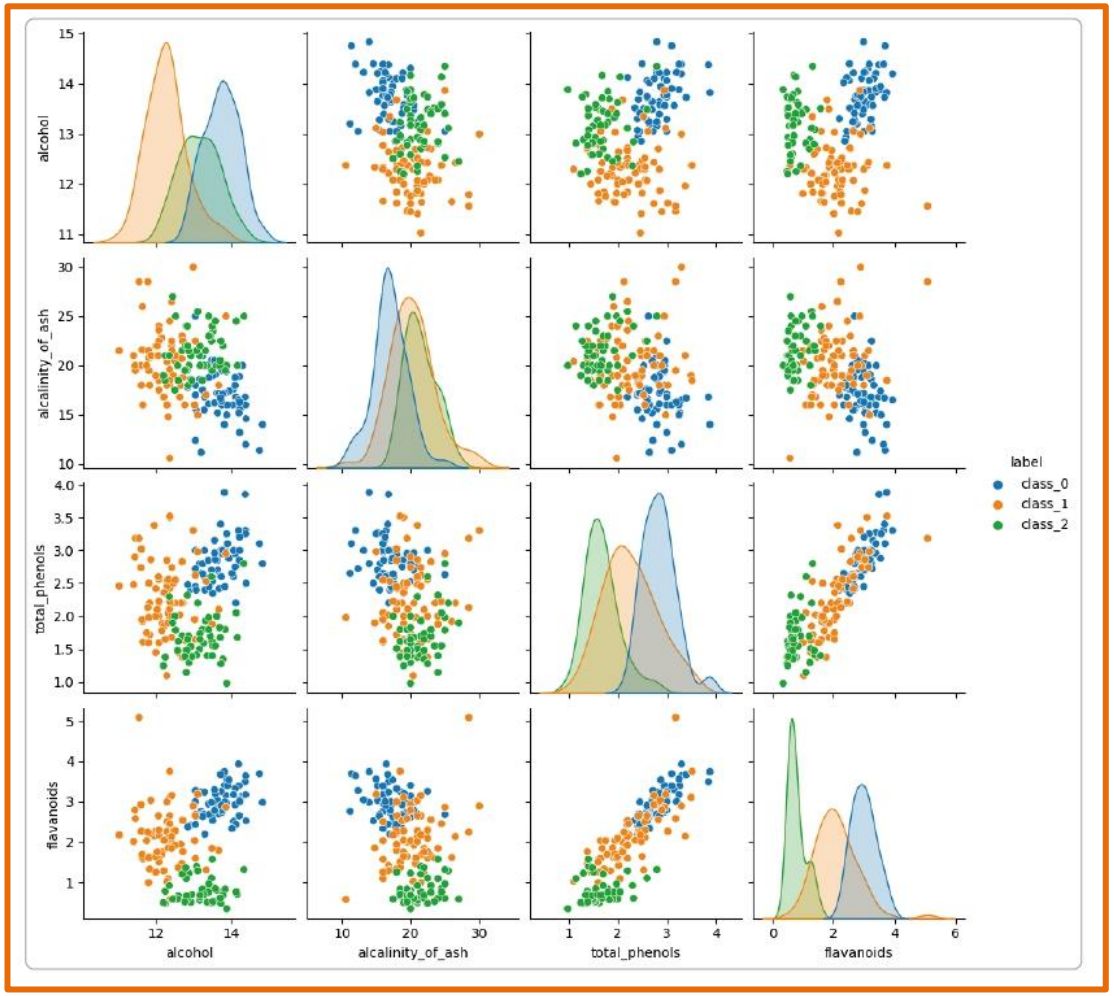
[코드 11-5] 특성 간 상관관계와 레이블 분포를 시각화

```
import seaborn as sns

df_analysis = pd.DataFrame(wine.data, columns=wine.feature_names)
wine_class = pd.Series(wine.target, dtype='category')
wine_class = wine_class.cat.rename_categories(wine.target_names)
df_analysis['label'] = wine_class

sns.pairplot(vars=['alcohol', 'alcalinity_of_ash', 'total_phenols', 'flavanoids'],\
         hue='label', data=df_analysis)
```

- 데이터프레임 df_analysis를 정의하고 데이터프레임 wine_class에 레이블 데이터를 할당.
- 특성 'alcohol', 'alcalinity_of_ash', 'total_phenols', 'flavanoids'을 선택하여 특성 간 상관관계와 특성별 레이블 분포를 확인.

I. 와인 경작자 분류

[코드 11-5] 실행결과



02. 인공지능과 분류

I. 와인 경작자 분류

– 중요도가 낮은 alcalinity_of_ash, total_phenols, proanthocyanins 열을 제거하고 학습을 진행.

[코드 11-6] 랜덤 포레스트 분류

```
from sklearn.ensemble import RandomForestClassifier

df_wine_proc = df_wine.drop(labels=['alcalinity_of_ash', 'total_phenols', 'proanthocyanins'], axis=1)
df_wine_proc = df_wine_proc.astype({'label':'int'})

train, test = train_test_split(df_wine_proc, test_size=0.3, random_state=0,\
                        stratify=df_wine_proc['label'])

train_X=train[train.columns[:10]]
train_Y=train[train.columns[10:]]
test_X=test[test.columns[:10]]
test_Y=test[test.columns[10:]]

model = RandomForestClassifier(n_estimators=3)
model.fit(train_X, train_Y)
pred_RF = model.predict(test_X)
print('랜덤 포레스트 알고리즘 분류 정확도:', metrics.accuracy_score(pred_RF, test_Y))
```

I. 와인 경작자 분류

[코드 11-6] 실행결과

```
랜덤 포레스트 알고리즘 분류 정확도: 0.9629629629629629
```

– 분류 정확도는 96.2%로 [코드 11-4]의 K-NN 알고리즘을 이용한 분류 결과보다 성능이 높음.

II. 화이트와인과 레드와인 분류

- 와인 품질 데이터로 화이트와인과 레드와인을 구분하는 이진분류를 실습.
- 연구용 공개 데이터셋 집합인 UCI 보관소의 와인 데이터를 사용.
- URL https://archive.ics.uci.edu/dataset/186/wine+quality를 입력하여 다운로드 페이지에 접속.

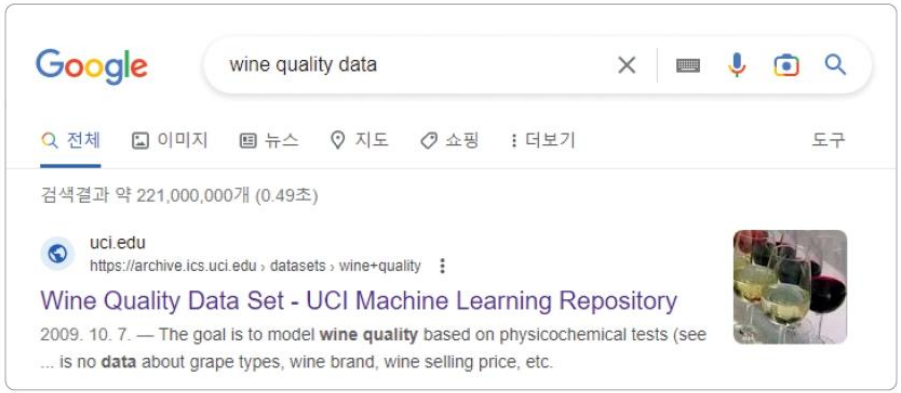


그림 11-6 wine quality data 검색 화면

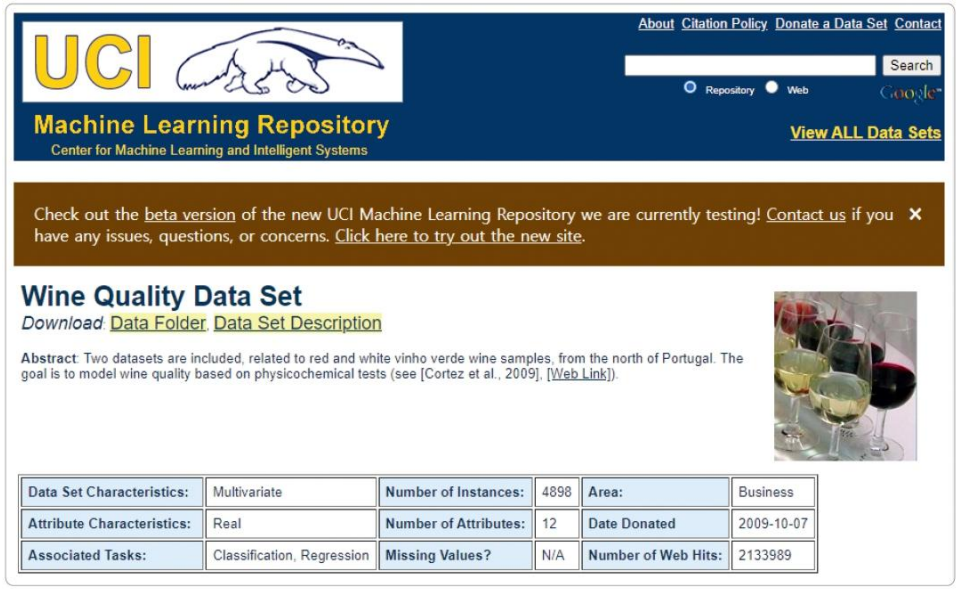


그림 11-7 UCI 기계학습 보관소의 와인 품질 데이터 페이지

II. 화이트와인과 레드와인 분류

– UCI의 파일 링크 주소를 복사하고 구글 Colab에서 직접 접근.

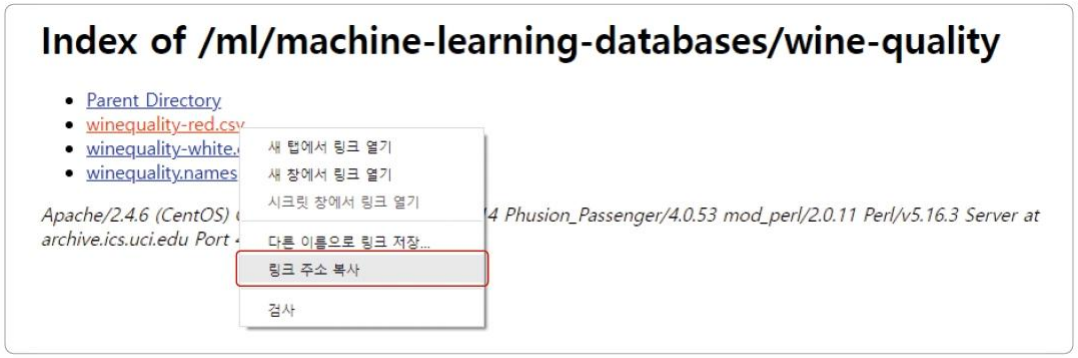


그림 11-8 데이터셋 링크 복사

II. 화이트와인과 레드와인 분류

[코드 11-7] 데이터프레임 생성

```
import numpy as np
import pandas as pd

df_white = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases\
                    /wine-quality/winequality-white.csv', sep=';')
df_red = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases\
                    /wine-quality/winequality-red.csv', sep=';')
df_white
```

	fixed acidity	volatile acidity	citric acid	resi	otal sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36		170.0	1.00100	3.00	0.45	8.8	6
1	6.3	0.30	0.34		132.0	0.99400	3.30	0.49	9.5	6
2	8.1	0.28	0.40		97.0	0.99510	3.26	0.44	10.1	6
3	7.2	0.23	0.32		186.0	0.99560	3.19	0.40	9.9	6
4	7.2	0.23	0.32		186.0	0.99560	3.19	0.40	9.9	6
...	...	...	...		...	...	...	...	...	...
4893	6.2	0.21	0.29		92.0	0.99114	3.27	0.50	11.2	6
4894	6.6	0.32	0.36		168.0	0.99490	3.15	0.46	9.6	5
4895	6.5	0.24	0.19		111.0	0.99254	2.99	0.46	9.4	6
4896	5.5	0.29	0.30		110.0	0.98869	3.34	0.38	12.8	7
4897	6.0	0.21	0.38		98.0	0.98941	3.26	0.32	11.8	6

4898 rows × 12 columns

II. 화이트와인과 레드와인 분류

	fixed acidity	volatile acidity	citric acid	resi	otal sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00		34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00		67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04		54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56		60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00		34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...		...	...	...	...	...	...
1594	6.2	0.600	0.08		44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10		51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13		40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12		44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47		42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

- 화이트와인 데이터는 4,898행, 레드와인 데이터는 4,898행.
- 화이트와 레드와인을 구분하는 이진분류 학습을 위해 데이터프레임에 레이블 열을 추가.

[코드 11-8] 레이블 열 추가

```
df_white['class'] = 1
df_red['class'] = 0

df_white
```

II. 화이트와인과 레드와인 분류

[코드 11-8] 실행결과

	fixed acidity	volatile acidity	citric acid	resi	lfur dioxide	density	pH	sulphates	alcohol	quality	class
0	7.0	0.27	0.36		170.0	1.00100	3.00	0.45	8.8	6	1
1	6.3	0.30	0.34		132.0	0.99400	3.30	0.49	9.5	6	1
2	8.1	0.28	0.40		97.0	0.99510	3.26	0.44	10.1	6	1
3	7.2	0.23	0.32		186.0	0.99560	3.19	0.40	9.9	6	1
4	7.2	0.23	0.32		186.0	0.99560	3.19	0.40	9.9	6	1
...	...	...	...		...	...	...	...	...	...	...
4893	6.2	0.21	0.29		92.0	0.99114	3.27	0.50	11.2	6	1
4894	6.6	0.32	0.36		168.0	0.99490	3.15	0.46	9.6	5	1
4895	6.5	0.24	0.19		111.0	0.99254	2.99	0.46	9.4	6	1
4896	5.5	0.29	0.30		110.0	0.98869	3.34	0.38	12.8	7	1
4897	6.0	0.21	0.38		98.0	0.98941	3.26	0.32	11.8	6	1

4898 rows × 13 columns

– 데이터프레임 df_white에 class 열을 추가하고 값으로 1을 입력. 데이터프레임 df_red에 class 열을 추가하고 값으로 0을 입력.

II. 화이트와인과 레드와인 분류

– 두 데이터프레임을 병합하여 하나의 데이터셋으로 만듦.

[코드 11-9] 데이터셋 병합

```
df_wine_category = pd.concat([df_white, df_red])
```

– 병합한 결과 6,497행 13열인 데이터프레임 df_wine_category를 얻음.

II. 화이트와인과 레드와인 분류

– df_wine_category를 학습 데이터와 테스트 데이터로 분할.

[코드 11-10] 학습 데이터 분할

```
from sklearn.model_selection import train_test_split

df_wine_category = df_wine_category.astype({'class':'int'})
train, test = train_test_split(df_wine_category, test_size=0.3, random_state=0,\
                    stratify=df_wine_category['class’])

train_X = train[train.columns[:12]]
train_Y = train[train.columns[12:]]
test_X = test[test.columns[:12]]
test_Y = test[test.columns[12:]]

train_X
```

II. 화이트와인과 레드와인 분류

[코드 11-10] 실행결과

	fixed acidity	volatile acidity	citric acid	res	l sulfur dioxide	density	pH	sulphates	alcohol	quality
2126	6.5	0.43	0.28		174.0	0.99860	3.31	0.55	9.3	5
871	7.7	0.34	0.58		151.0	0.99780	3.06	0.49	8.6	5
3213	7.5	0.18	0.45		158.0	0.99270	3.01	0.38	10.6	6
725	6.4	0.39	0.21		136.0	0.99225	3.15	0.46	10.2	5
4557	6.1	0.37	0.46		210.0	0.99700	3.17	0.59	9.7	6
...	...	...	...		...	...	...	...	...	...
3754	5.4	0.46	0.15		130.0	0.98953	3.39	0.77	13.4	8
2032	6.5	0.25	0.20		101.0	0.99160	3.24	0.54	10.8	6
3239	6.6	0.34	0.24		99.0	0.99031	3.10	0.40	12.3	7
1411	6.4	0.47	0.40		19.0	0.99630	3.56	0.73	10.6	6
3864	6.6	0.39	0.22		98.0	0.99018	3.25	0.53	13.0	7

4547 rows × 12 columns

- 분류 문제이므로 class 열의 레이블 값을 범주형으로 변환.
- train_test_split() 함수로 학습 데이터와 테스트 데이터를 7:3 비율로 분할, stratify 인자로 레이블인 class의 분포에 맞추어 데이터를 분할하도록 명령.
- 특성 열과 레이블 열을 따로 변수에 저장.

II. 화이트와인과 레드와인 분류

- 학습준비가 끝났음
- 이번에는 이진분류 성능이 좋다고 알려진 SVM 알고리즘으로 분류.
- 사이킷런에서 svm 패키지를 불러오고, 모형의 성능을 평가하는 데 사용할 metrics 패키지와 f1_score를 함께 불러오기.

[코드 11-11] 라이브러리

```
from sklearn import svm
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
```

II. 화이트와인과 레드와인 분류

– 데이터셋을 학습하고 이 모형이 얼마나 정확하게 분류하는지 평가.

[코드 11-12] SVM 이진분류

```
model = svm.SVC(kernel='linear', C=0.1, gamma=0.1)
model.fit(train_X, train_Y)
pred_SVM = model.predict(test_X)

print('SVM 알고리즘 분류 정확도:', metrics.accuracy_score(pred_SVM, test_Y))
print(classification_report(pred_SVM, test_Y))
```

```
SVM 알고리즘 분류 정확도: 0.9846153846153847
              precision recall f1-score support
           0       0.95   0.98     0.97     466
           1       0.99   0.99     0.99    1484
    accuracy                          0.98    1950
   macro avg       0.97   0.98     0.98    1950
weighted avg       0.98   0.98     0.98    1950
```

– 98.46%의 정확도로 화이트와인과 레드와인을 분류했음.

II. 화이트와인과 레드와인 분류

- classification_report() 함수는 다각도로 모형을 평가함.
- 정밀도(Precision)는 모형의 분류 결과 중 맞은 것의 비율이고, 재현율(Recall)은 실제로 레드와인인 데이터 중 모형이 레드와인으로 분류한 것의 비율.
- F1-score는 인공지능 모형 평가에 매우 자주 사용되는 지표로, 정밀도와 재현율의 조화평균.

✓**하나 더 알기**: F1-score의 장점

데이터셋에서 레이블의 분포가 일정하지 않을 때가 많은데, 이러한 경우에 정확도만으로 모형을 평가하면 통계상의 오류를 범할 수 있음. 따라서 각각의 클래스를 잘 맞추고 덜 틀릴 수 있는 방식으로 성능을 평가해야 함.

II. 화이트와인과 레드와인 분류

– 랜덤 포레스트 알고리즘으로 이진분류를 수행.

[코드 11-13] 랜덤 포레스트 이진분류

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=10)
model.fit(train_X, train_Y)
pred_RF = model.predict(test_X)

print('Random Forest 알고리즘 분류 정확도:', metrics.accuracy_score(pred_RF,test_Y))
print(classification_report(pred_RF,test_Y))
```

```
Random Forest 알고리즘 분류 정확도: 0.9917948717948718
              precision recall f1-score support
           0       0.97   0.99     0.98     472
           1       1.00   0.99     0.99    1478

    accuracy                         0.99    1950
   macro avg       0.99   0.99     0.99    1950
weighted avg       0.99   0.99     0.99    1950
```

– 랜덤 포레스트 모형은 SVM 모형보다 약간 향상된 99.2%의 정확도로 와인을 분류함.

03

인공지능과 예측

03. 인공지능과 예측

I. 와인 등급 예측

- 와인 품질 데이터의 11가지 특성으로 와인의 품질(Quality)를 예측하는 모형을 만들기.
- 예측 모형 학습에 앞서 예측 대상인 와인 등급의 분포를 확인.

[코드 11-14] 레이블 분포 확인

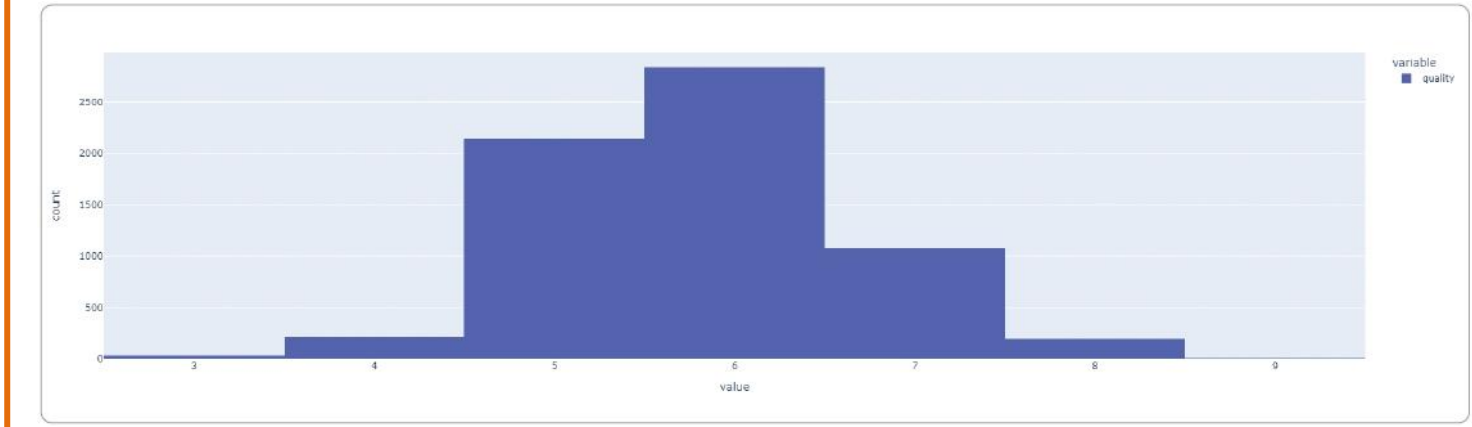
```
import numpy as np
import pandas as pd
import plotly.express as px

df_white = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases\
                  /wine-quality/winequality-white.csv', sep=';')
df_red = pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases\
                  wine-quality/winequality-red.csv', sep=';')
df_wine_predic = pd.concat([df_white, df_red])

px.histogram(df_wine_predic.quality)
```

- 판다스의 concat( ) 함수로 데이터프레임 df_white와 df_red를 하나로 병합.
- plotly.express 라이브러리를 이용하여 레이블인 quality 열을 히스토그램으로 시각화.

03. 인공지능과 예측

I. 와인 등급 예측

[코드 11-14] 실행결과



I. 와인 등급 예측

- 특성들의 상관관계 및 특성과 레이블의 상관관계를 파악하기.
- pairplot( ) 함수 이외에도 다양한 시각화 함수가 있는데,
  그 중 히트맵(Heatmap)은 다변량 데이터를 색상으로 나타내는 그래프

[코드 11-15] 데이터 특성 히트맵

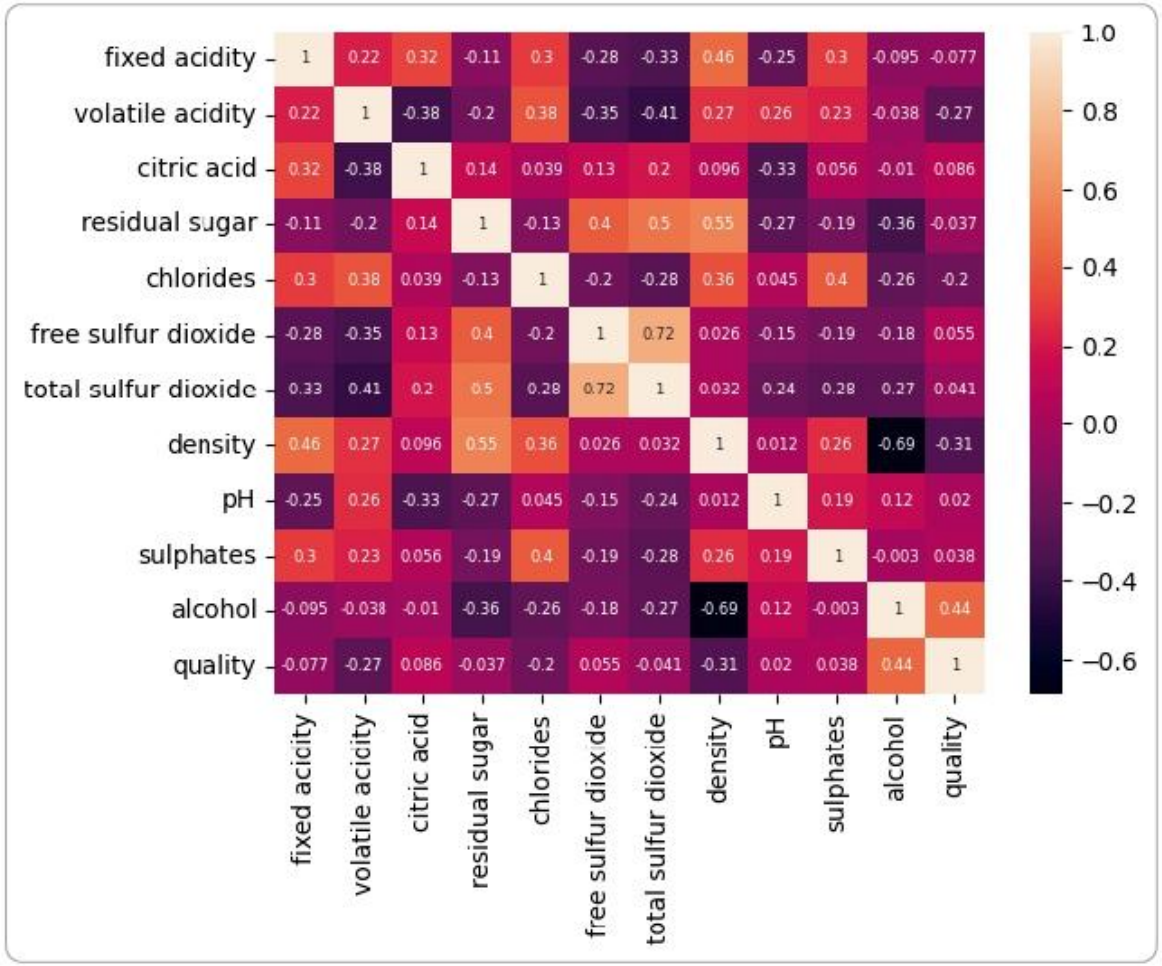
```
import matplotlib.pyplot as plt
import seaborn as sns

ax = sns.heatmap(df_wine_predic.corr( ), annot=True, annot_kws={'size':6})
plt.show( )
```

- 시본의 heatmap( ) 함수로 시각화.
- 먼저 corr( ) 함수로 데이터프레임 df_wine_predic의 특성 간 상관관계를 계산.

03. 인공지능과 예측

I. 와인 등급 예측

[코드 11-15] 실행결과



I. 와인 등급 예측

– 데이터를 학습 데이터와 테스트 데이터로 분할.

[코드 11-16] 테스트 데이터 분할

```
from sklearn.model_selection import train_test_split

df_wine_predic.columns = df_wine_predic.columns.str.replace(' ', '_')
train, test = train_test_split(df_wine_predic, test_size=0.3, random_state=0,
stratify=df_wine_predic['quality’])

train_X = train[train.columns[:11]]
train_Y = train[train.columns[11:]]
test_X = test[test.columns[:11]]
test_Y = test[test.columns[11:]]
```

– str.replace( ) 함수로 열 이름에 포함된 공백을 언더바(_)로 교체.

– 학습 데이터와 테스트 데이터를 7:3으로 분리하고
인덱스가 11인 quality 열을 분할하여 train_Y에 레이블로 저장.

I. 와인 등급 예측

– 사용할 특성이 11개이므로 다중 선형 회귀분석 알고리즘을 사용.

[코드 11-17] 다중 선형 회귀모형 학습

```
from statsmodels.formula.api import ols, glm
reg_form = 'quality ~ fixed_acidity + volatile_acidity + citric_acid\
 + residual_sugar + chlorides + free_sulfur_dioxide\
 + total_sulfur_dioxide + density + pH + sulphates + alcohol'
reg_result = ols(reg_form, data=train).fit( )
```

– OLS(최소자승법) 함수로 회귀 예측을 수행. 종속변수 quality와 독립변수 11개로 모형을 작성.

I. 와인 등급 예측

- 생성된 모형에 테스트 데이터를 입력하고 정확도를 계산.
- 학습된 회귀 모형 reg_result에 predict( ) 함수를 사용하여 test 데이터의 와인 등급 점수를 예측.

[코드 11-18] 다중 선형 회귀모형 예측

```
reg_predict = reg_result.predict(test)
print(reg_predict)
reg_predict = reg_predict.astype('int')
```

```
1209 6.209276
3193 5.647934
1192 6.724634
3852 6.538731
2022 5.294939
 ...
305 5.485388
1963 5.194879
4305 6.015268
4798 5.933790
1384 4.731303
Length: 1950, dtype: float64
```

I. 와인 등급 예측

[코드 11-19] 예측 정확도

```
from sklearn import metrics
print('다중 선형회귀 알고리즘 예측 정확도:', metrics.accuracy_score(reg_predict,test_Y))
```

```
다중 선형회귀 알고리즘 예측 정확도: 0.4574358974358974
```

- 예측 정확도는 약 46%.
- 특성이 11개인 와인 데이터를 다중 회귀모형으로 학습하여 0에서 10 까지의 정수 등급을 예측했더니 분류 문제보다는 정확도가 낮게 나타남.
- 그 이유는 범주형과 연속형 값의 차이 때문.

I. 와인 등급 예측

[코드 11-20] 랜덤 포레스트 예측

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100)
model.fit(train_X, train_Y)
pred_RF = model.predict(test_X)
print('랜덤 포레스트 알고리즘 예측 정확도:', metrics.accuracy_score(pred_RF, test_Y))
```

```
랜덤 포레스트 알고리즘 예측 정확도: 0.6635897435897435
```

– 예측 정확도는 약 65.2%. 다중 선형회귀보다 약 20%p 높음

03. 인공지능과 예측

I. 와인 등급 예측

- 인공지능 모형은 특정 알고리즘의 고유한 수식으로 학습을 수행하는데 이때 인자 값을 설정하여 알고리즘이 동작하는 방식을 정의할 수 있음.
- 이러한 인자를 인공지능 모형의 하이퍼파라미터(Hyper parameter)라고 부름.
- 랜덤 포레스트는 트리 계열 알고리즘으로 의사결정 트리(Decision Tree)가 여러 개 있는 숲과 같은데, 이 알고리즘은 학습 데이터를 최대한 중복되지 않게 나눈 후 각각 트리를 생성함.
- 트리들은 서로 학습한 데이터가 다르기 때문에 트리 간 상관관계가 최소화되어 분류 성능이 극대화.
- 다만, 트리 수가 많더라도 데이터 양이 충분하지 않거나 특성 수가 부족하면 오히려 과적합(Overfitting)이 발생.

✓**하나 더 알기**: 하이퍼파라미터 튜닝

모형 성능을 최적화하기 위하여 하이퍼파라미터를 조정하여 모형 성능 확인을 반복하는 과정을 하이퍼파라미터 튜닝이라고 부름. 튜닝에는 여러 방법이 있지만 베이지안 최적화, 그리드 서치, 랜덤 서치가 널리 사용됨.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 아주 간단한 인공신경망으로 와인 등급 데이터를 학습하고 예측 정확도를 확인함.
- 먼저 인공신경망을 사용하는 데 필요한 라이브러리를 가져오기.

[코드 11-21] 케라스 라이브러리

```
from keras.models import Sequential
from keras.layers import Dense
```

- 시퀀셜(Sequential)은 [그림 11-11]과 같이 인공신경망에 연결 구조를 만들어서 모형을 생성할 수 있도록 틀을 정의하고 덴스(Dense)는 뉴런으로 층(Layer)을 만듦.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 인공신경망에는 입력층, 은닉층, 출력층이 있음.
- 입력층(Input layer)에서는 학습 데이터를 입력받아 신경망에 전달.
- 은닉층(Hidden layer)에서는 뉴런 간에 데이터가 전파되어 학습이 진행됨.
- 출력층(Output layer)에서는 레이블 값을 출력.

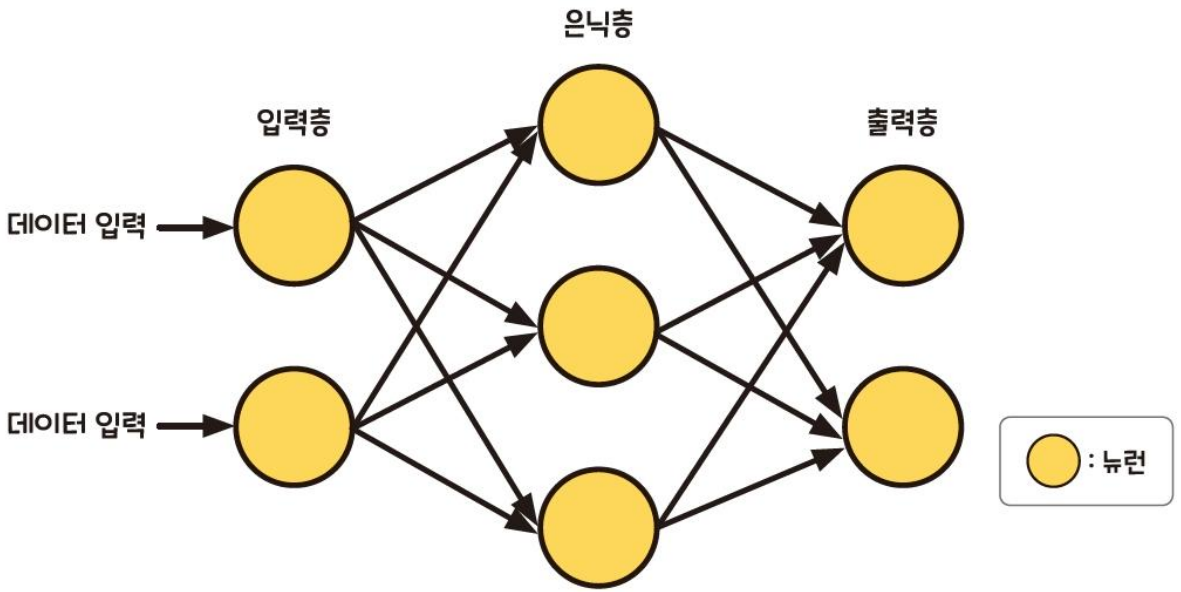


그림 11-11 인공신경망

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

[코드 11-22] 인공신경망 틀 정의와 입력층 생성

```
model = Sequential( )
model.add(Dense(64,input_dim=11, activation='relu'))
```

- 모형을 할당할 변수 model을 정의하고 Sequential( ) 함수를 불러와 층을 연속적으로 연결.
- 데이터를 입력받는 입력층을 설계하고 정의. 층을 정의하고 연결할 때 층의 종류와 상관없이 모두 add( ) 함수를 사용하고, Dense( ) 함수로 층 하나를 정의하여 add( ) 함수로 연결.
- 첫 번째 층은 입력층이므로 입력 데이터의 특성을 고려해야함.

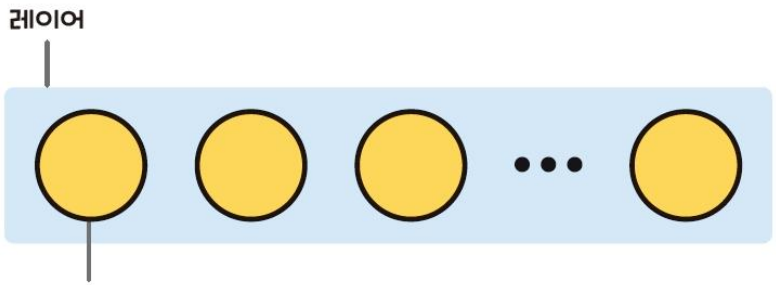


그림 11-12 입력층

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 이어서 은닉층을 생성하고 연결.

[코드 11-23] 인공신경망의 은닉층 설정

```
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
```

- 데이터 특성 정보는 입력층에서 전달했기 때문에 다시 입력하지 않음.
- 뉴런 수는 입력층보다 작게, 출력할 클래스 수보다는 크게 설정.
- 두 번째 은닉층은 뉴런을 16개로 설정하고, 세 번째 은닉층은 뉴런을 8개로 설정.
- 마지막으로 레이블을 판별할 출력층을 생성.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

[코드 11-24] 인공신경망의 출력층 설정

```
model.add(Dense(7, activation = 'softmax'))
model.summary()
```

- softmax는 logistic 함수와 마찬가지로 값에 따라 클래스를 구분하는 함수.
- summary( ) 함수로 생성된 모형의 구조를 출력하여 확인.
  입력층이 1개, 은닉층이 3개, 출력층이 1개로 총 다섯 층인 모형이 생성되었음.
- Output Shape의 숫자는 각 층의 뉴런 개수, Param은 하이퍼파라미터의 개수.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

[코드 11-24] 실행 결과

```
Model: "sequential"
_________________________________________________________________
Layer (type) Output Shape Param #
=================================================================
dense (Dense) (None, 64) 768

dense_1 (Dense) (None, 32) 2080

dense_2 (Dense) (None, 16) 528

dense_3 (Dense) (None, 8) 136

dense_4 (Dense) (None, 7) 63

=================================================================
Total params: 3,575
Trainable params: 3,575
Non-trainable params: 0
_________________________________________________________________
```

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 설계한 모형을 실제로 사용하려면 컴파일하여 컴퓨터에 실체화해야 함.

[코드 11-25] 컴파일

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- loss 인자는 반복 학습 중에 정답과 비교하여 얼마나 틀렸는지 측정하는 지표를 설정함. optimizer인자로 모형 최적화 방식을 설정하고, metrics 인자로 모형 성능을 측정하는 지표를 설정.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 모형 학습 준비가 끝났으나 데이터셋을 모형에 입력 가능한 형태로 가공하는 단계가 아직 남아 있음. 레이블 정보가 포함된 train_Y와 test_Y는 일부 가공해야 함.
- 현재 레이블은 3에서 9까지의 정수라서 인공신경망 모형은 출력에 사용할 뉴런을 7개 가지고 있는데, Softmax 함수로 각 등급을 예측했을 때 참(True, 1)일 확률을 계산하고 가장 높은 확률인 등급을 선택함.
- 그래서 각 등급을 1과 0만으로 다시 구성해야 함.
- 원핫 인코딩(One-hot encoding)은 범주형 데이터를 더미 변수로 변환하는 기법.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

– 원핫 인코딩으로 1차원의 레이블 데이터를 클래스마다 1과 0으로 이루어진 7차원 데이터로 변환하기.

[코드 11-26] 원핫 인코딩

```
from sklearn.preprocessing import LabelEncoder
#from keras.utils import np_utils
import np_utils
from tensorflow.keras.utils import to_categorical

#레이블을 학습 데이터 레이블 모형으로 변환하기
encoder = LabelEncoder( )
encoder.fit(train_Y)

onehot_train_Y = encoder.transform(train_Y)
train_f_Y = np_utils.to_categorical(onehot_train_Y)
onehot_test_Y = encoder.transform(test_Y)
test_f_Y = np_utils.to_categorical(onehot_test_Y)

print(test_f_Y)
```

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

[코드 11-26] 실행 결과

```
array([[0., 0., 0., ..., 1., 0., 0.],
 [0., 0., 1., ..., 0., 0., 0.],
 [0., 0., 0., ..., 1., 0., 0.],
 ...,
 [0., 0., 0., ..., 0., 0., 0.],
 [0., 0., 1., ..., 0., 0., 0.],
 [0., 0., 1., ..., 0., 0., 0.]], dtype=float32)
```

- 원핫 인코딩 수행 시 주의: 학습 데이터 레이블에는 있지만 테스트 데이터 레이블에는 포함되지 않은 레이블이 있을 수 있음.
- 원핫 인코딩에서는 모든 레이블 정보를 포함하는 데이터로 인코딩 모형을 먼저 설정하고 나머지 데이터를 변환해야 함.
- 여기서는 학습 데이터셋이 더 많은 데이터를 가지고 있으므로 encoder.fit(train_Y)와 같이 학습 데이터로 모형을 생성하고, 같은 모형으로 테스트 데이터까지 변환함.

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

- 학습을 수행하고 결과를 확인하기.
- 몇 가지 인자를 추가로 설정함.

[코드 11-27] 인공신경망 학습

```
model.fit(train_X, train_f_Y, epochs=500, batch_size=10)
```

```
Epoch 1/500
455/455 [==============================] - 2s 1ms/step - loss: 1.4932 - accuracy: 0.3495
Epoch 2/500
455/455 [==============================] - 1s 1ms/step - loss: 1.3099 - accuracy: 0.4240
Epoch 3/500
455/455 [==============================] - 1s 1ms/step - loss: 1.2881 - accuracy: 0.4363
Epoch 4/500
455/455 [==============================] - 1s 1ms/step - loss: 1.2798 - accuracy: 0.4366
...
Epoch 497/500
455/455 [==============================] - 1s 1ms/step - loss: 0.8607 - accuracy: 0.6202
Epoch 498/500
455/455 [==============================] - 1s 1ms/step - loss: 0.8682 - accuracy: 0.6156
Epoch 499/500
455/455 [==============================] - 1s 1ms/step - loss: 0.8671 - accuracy: 0.6136
Epoch 500/500
455/455 [==============================] - 1s 1ms/step - loss: 0.8688 - accuracy: 0.6134
```

II. 심화분석: 인공신경망을 활용한 와인 등급 예측

– 테스트 데이터로도 예측을 수행하여 정확도를 측정.

[코드 11-28] 예측 성능 평가

```
model.evaluate(test_X, test_f_Y)
```

```
61/61 [==============================] - 0s 2ms/step - loss: 1.2614 - accuracy: 0.5492
```

– 평가에는 evaluate( ) 함수 사용. 모형은 학습 완료 후 예측 정확도가 약 64%였으나 테스트 데이터로 검증한 결과 약 55%의 확률로 예측하는 것이 확인됨.

– 많은 시간을 들여 학습을 수행하였지만 랜덤 포레스트를 사용했을 때보다 예측 정확도가 낮음.

THANK YOU

Q&A?