

AI 프로그래밍 #01

AI 프로그래밍 - KNN

Fly-AI in No.4

2023.12.27
대전대학교 PhD. 조 영복
ybcho@dju.ac.kr

머신러닝 데이터 전처리

❖ 데이터 전처리

- 레이블 인코딩
- 원핫인코딩

머신러닝 데이터 전처리

❖ 레이블 인코딩

```
from sklearn.preprocessing import LabelEncoder
```

```
items = ["tv", "냉장고", "컴퓨터", "전자레인지", "믹서", "선풍기", "믹서"]
```

```
encoder = LabelEncoder()
```

```
encoder.fit(items)
```

```
labels = encoder.transform(items)
```

```
print('labels:', labels)
```

```
print('encoder.classes:', encoder.classes_)
```

```
print('encoder.inverse_transform :', encoder.inverse_transform([4, 5, 3, 2, 4, 2, 0, 1, 1]))
```

```
labels: [0 1 5 4 2 3 2]
```

```
encoder.classes: ['tv' '냉장고' '믹서' '선풍기' '전자레인지' '컴퓨터']
```

```
encoder.inverse_transform : ['전자레인지' '컴퓨터' '선풍기' '믹서' '전자레인지' '믹서' 'tv' '냉장고' '냉장고']
```

머신러닝 데이터 전처리

❖ 원핫 인코딩

- 하나만 1로 남기고 나머진 0으로변경
- get_dummies()
 - ▶ 전체 범주형 변수에 대한 더미변수 만들기

```
import pandas as pd
df=pd.DataFrame( { 'items':items } )
print(df)
```

```
print(pd.get_dummies(df))
```

	items
0	tv
1	냉장고
2	컴퓨터
3	전자레인지
4	믹서
5	선풍기
6	믹서

	items_tv	items_냉장고	items_믹서	items_선풍기	items_전자레인지	items_컴퓨터
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	0	0	0	1
3	0	0	0	0	1	0
4	0	0	1	0	0	0
5	0	0	0	1	0	0
6	0	0	1	0	0	0

데이터 스케일링

- breast_cancer()
 - StandardScaler()
 - MinMaxScaler()
 - RobustScaler()
 - Normalizer()

SVM

❖ 데이터 스케일링

- 데이터 전처리
- 피처(feature)들마다 데이터 값의 범위가 다르기 때문에 범위 차이가 클경우 데이터를 갖고 모델링할때 0으로 수렴하거나 무한으로 발산
- 모든 피처들의 데이터 분포나 범위를 동일하게 조정
 - ▶ StandardScaler()
 - ▶ MinMaxScaler()
 - ▶ MaxAbsScaler()
 - ▶ RobustScaler()
 - ▶ Normalizer()

SVM-유방암 데이터 셋 실습

```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
```

```
cancer=load_breast_cancer()
```

```
cancer_df = pd.DataFrame(data=cancer.data, columns=cancer.feature_names)
cancer_df['target'] = cancer.target
cancer_df.head()
```

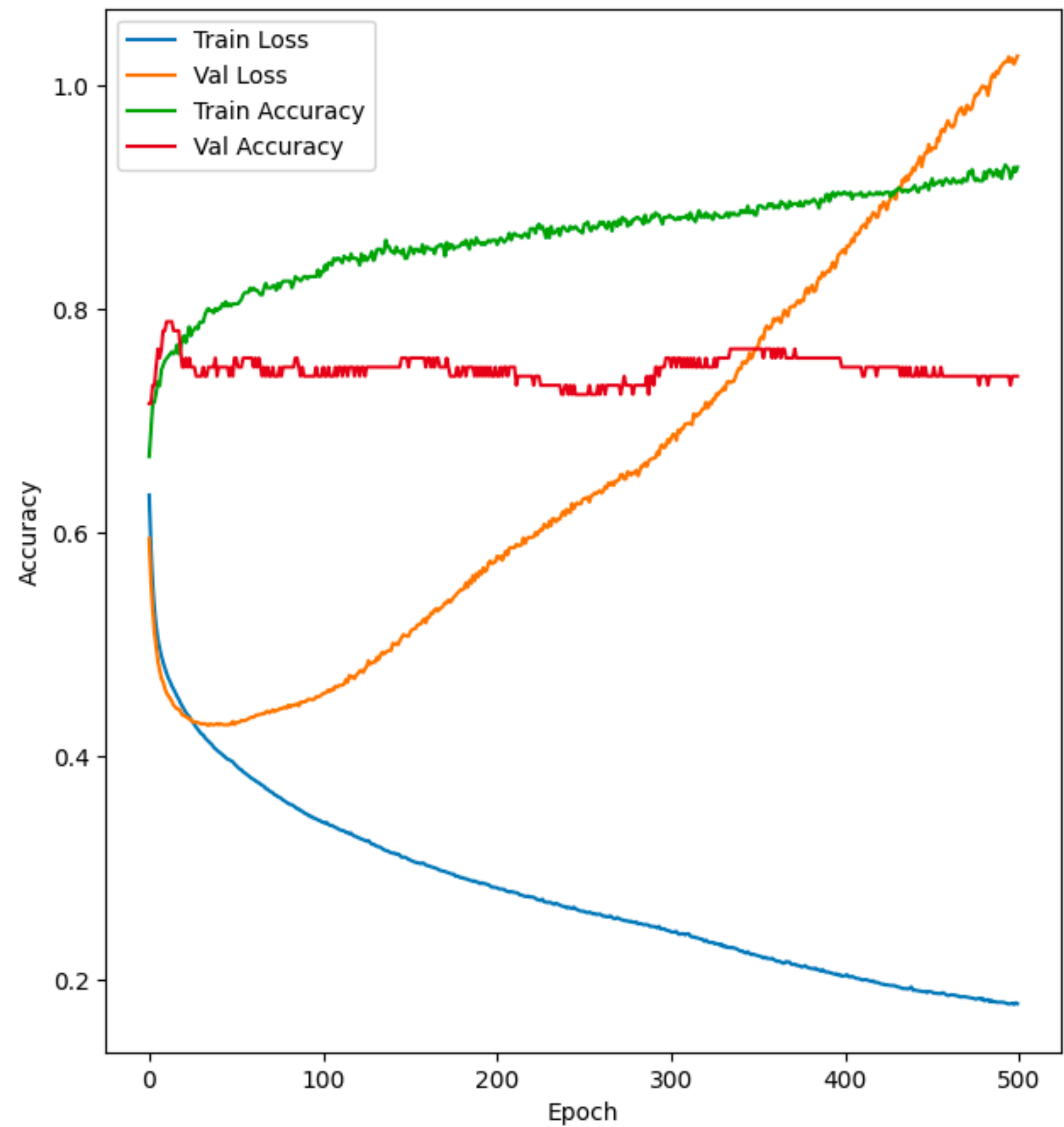
KNN

❖ Pandas

- Pandas

- ▶ 데이터 처리와 분석을 위한 라이브러리
- ▶ 행과 열로 이루어진 데이터 객체를 만들어 다룰 수 있음
- ▶ 대용량의 데이터들을 처리하는데 매우 편리
- ▶ pandas 자료구조
 - Series : 1차원
 - DataFrame : 2차원
 - Panel : 3차원
 - Pandas 로딩

KNN-피마인디언 당뇨병 데이터



금융상품 갱신 여부 예측하는 인공지능망 구성하기

라이브러리 호출

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
```

데이터 확인

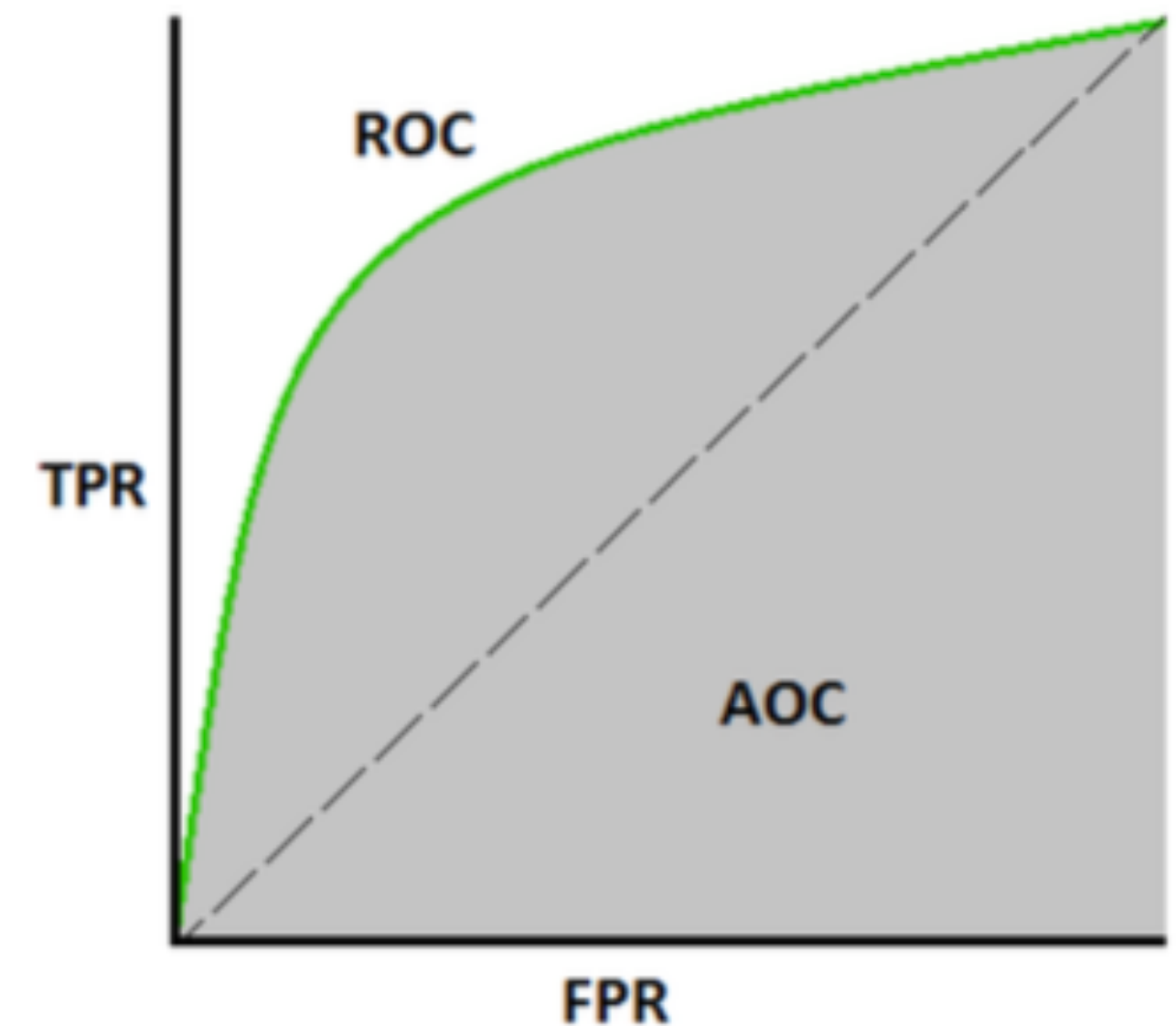
```
df = pd.read_csv(' ./fly-AI/Churn_Modelling.csv')
df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112357.68
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113920.34
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93000.67
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	101684.42

Confusion Matrix

❖ 예측값

- 정확도 (accuracy)
 - ▶ 전체의 경우 중에 맞은 경우에 대한 확률, $(TN+TP) / (\text{전체})$
- 정밀도 (precision)
 - ▶ 맞다고 예측한 애들 중에 실제로 맞은것의 비율, $TP / (FP+TP)$
- 재현율 (recall)
 - ▶ 실제로 맞은 사람들 중에 맞았다고 예측한 애들의 비율, $TP / (FN+TP)$
- F1 스코어
 - ▶ 정밀도와 재현율은 서로 트레이드오프 관계, 둘 다 높은 점수를 맞아야 좋은 모델이라 할 수 있음
 - ▶ F1 스코어는 정밀도와 재현율의 조화 평균, $2 / [(1/\text{recall}) + (1/\text{precision})]$
- ROC, AUC
 - ▶ 맞출 때마다 TPR(수직) 방향으로 한 칸씩 올라갑니다. 그리고 틀리면 FPR(수평) 방향으로 이동



Confusion Matrix

❖ 예측값

- 정확도 (accuracy)
 - `from sklearn.metrics import accuracy_score`
 - `accuracy_score(정답, 예측)`
- 정밀도 (precision)
 - `from sklearn.metrics import precision_score`
 - `precision_score(정답, 예측)`
- 재현율 (recall)
 - `from sklearn.metrics import recall_score`
 - `recall_score(정답, 예측)`
- F1 스코어
 - `from sklearn.metrics import f1_score`
 - `f1_score(정답, 예측)`
- ROC, AUC
 - `from sklearn.metrics import roc_auc_score`
 - `roc_auc_score(정답, 예측)`

Confusion Matrix

❖ 데이터 스케일링 라이브러리

– `from` sklearn.preprocessing `import` StandardScaler, MinMaxScaler, MaxAbsScaler, RobustScaler, Normalizer

▶ StandardScaler()

- 평균이 0이고 분산이 1인 정규 분포로 만드는 것
- 표준편차란 평균으로부터 얼마나 떨어져있는지를 구한 것

▶ MinMaxScaler()

- 최대값은 1 최소값은 0으로 최소-최대 정규화 Min-Max Normalization
- 이상치에 취약.

▶ MaxAbsScaler()

- MaxAbsScaler는 MinMaxScaler와 비슷
- 방법의 이름에서도 알 수 있듯이 모든 피쳐들의 절댓값이 0과 1 사이에 놓이도록 만들어줌.
- 즉, 0을 기준으로 절댓값이 가장 큰 수가 1또는 -1의 값을 가짐.
- 마찬가지로, 이상치의 영향을 크게 받기 때문에 이상치가 존재할 경우 이 방법은 적절하지 않음.

Confusion Matrix

❖ 데이터 스케일링 라이브러리

- `from sklearn.preprocessing import StandardScaler, MinMaxScaler, MaxAbsScaler, RobustScaler, Normalizer`
 - ▶ RobustScaler()
 - StandardScaler는 평균과 분산을 사용했지만 RobustScaler는 중간값(median)과 사분위값(quartile)을 사용
 - 이상치의 영향을 최소화
 - 표준화 후 데이터가 더 넓게 분포
 - ▶ Normalizer()
 - 열(Columns)을 대상으로 함
 - 한 행의 모든 피쳐들 사이의 유클리드 거리가 1이 되도록 데이터값 생성
 - 좀 더 빠르게 학습할 수 있고 과대적합 확률을 낮출 수 있다

결정트리(Decision Tree)

❖ Decision Tree

- 의사결정 규칙을 나무 구조로 나타내어 전체 자료를 몇 개의 작은 집단으로 나누어서 분석하는 기법
- SVM 처럼 결정트리(decision tree)는 분류 및 회귀가 가능한 머신러닝 알고리즘
- 매우 복잡한 데이터셋도 학습할 수 있는 강력한 알고리즘
- 최근에 많이 사용하는 랜덤 포레스트의 기본 구성 요소
 - ▶ Root Node : 깊이가 0인 꼭대기 노드
 - ▶ Leaf Node : 자식 노드가 없는 마지막 노드
 - ▶ Gini Impurity : 한 노드의 모든 샘플이 같은 클래스에 속해있으면, 해당 노드는 순수(gini=0)
- **결정 트리의 장점** : 스케일이나 평균을 원점에 맞추는 것과 같은 데이터 전처리가 거의 필요하지 않음.
- **사이킷런**: 이진 트리(자식 노드의 수가 2개 이하)만 만드는 CART 알고리즘을 사용.
- **결정 트리**: 위와 같이 매우 직관적이고 이해하기 쉽고, 해석력이 아주 좋다. ('화이트 박스')

결정트리(Decision Tree)

❖ 규제 매개변수

- 결정 트리의 경우 훈련 데이터에 대한 가정을 보통 두지 않음(선형모델은 데이터가 선형일거라 가정)
- 훈련 데이터에 대한 일반적인 가정을 두지 않는다면, 모델 자체가 훈련 데이터와 아주 가깝게 만들려고 해서 과적합이 발생할 수 있음
- 결정 트리 모델의 경우에는 일반적인 선형 모델과 같이 파라미터 모델이 아니다
- 결정 트리 모델은 훈련되기 전에 파라미터의 수를 지정할 수 없는 비파라미터 모델(non-parametric model).
- 비파라미터 모델은 훈련 데이터에 맞춰지기 때문에 모델 구조가 고정되지 않고 자유롭다.
- 파라미터 모델은 미리 파라미터를 정할 수 있기 때문에 제한적이지만 과대적합의 위험을 조절할 수 있음.
- 결정 트리는 보통 최대 깊이(max_depth)로 모델의 과적합을 규제할 수 있음.
- max_depth가 낮아지면 과대적합의 위험이 감소

결정트리(Decision Tree)

❖ 결정 트리의 규제 매개변수 종류

- **max_depth** : 트리 최대 깊이 : max_depth 감소 → 모델 규제 증가 → **과적합 감소**
- **min_samples_split** : 분할되기 위해 노드가 가져야 하는 최소 샘플 수
 - min_samples_split 증가 → 모델 규제 증가 → 과적합 감소
- **min_samples_leaf** : leaf node가 가지고 있어야할 최소 샘플 수
 - min_samples_leaf 증가 → 모델 규제 증가 → 과적합 감소
- **min_weight_fraction_leaf** : min_samples_leaf와 비슷, 가중치가 부여된 전체 샘플 수에서의 비
 - min_weight_fraction_leaf 증가 → 모델 규제 증가 → 과적합 감소
- **max_leaf_nodes** : leaf node의 총 최대 개수
 - max_leaf_nodes 감소 → 모델 규제 증가 → 과적합 감소
- **max_features** : 최상의 분할을 찾을 때 고려할 기능의 수
 - max_features 감소 → 모델 규제 증가 → 과적합 감소

앙상블 보팅

❖ 앙상블 학습

- 보팅, 배깅, 부스팅
- 보팅은 기본적으로 여러 개의 분류기를 사용하여 각각의 분류기 결과를 투표하여 예측
- 보팅(Voting)
 - ▶ 서로 다른 알고리즘을 가진 분류기의 조합, 다른 알고리즘을 사용하여 분류기를 조합
 - ▶ 하드 보팅과 소프트 보팅
 - ▶ 하드 보팅이란, 각각 분류기의 결과값 중 가장 많은 걸 따른다.
 - ▶ 소프트 보팅이란, 분류기의 확률을 더하고 각각 평균을 내서 확률이 제일 높은 값으로 결과값을 선정

앙상블 보팅

❖ 앙상블 학습

- 배깅(Bagging)

- ▶ 보팅과 다르게 서로 같은 알고리즘의 분류기 조합
- ▶ 분류기 1의 데이터는 [0,0,3,4,5,5] 분류기 2의 데이터는 [0,1,2,3,4,5] 분류기 3의 데이터는 [0,1,1,2,4,5]
이런 식으로 개별 데이터의 중첩을 허락