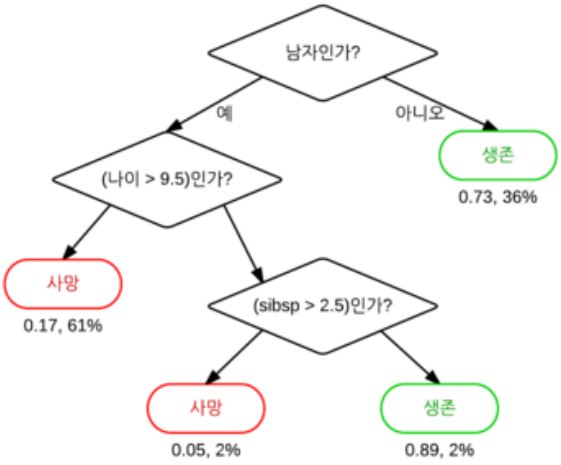
4장. 분류

💡 지도학습의 대표적인 유형인 분류는 학습 데이터로 주어진 데이터의 피처와 레이블값(결정 값, 클래스 값)을 머신러닝 알고리즘으로 학습해 모델을 생성하고, 이렇게 생성된 모델에 새로운 데이터 값이 주어졌을 때 미지의 레이블 값을 예측하는 것

2️⃣ 결정트리

결정트리는 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리(Tree) 기반의 분류 규칙을 만든다. 일반적으로 쉽게 표현하는 방법은 **if/else** 로 스무고개 !!

아래 그림에서 다이아몬드 모양은 규칙 노드를, 타원형은 리프 노드를 뜻한다. 데이터 세트에 피처가 있고 이러한 피처가 결합해 규칙 조건을 만들 때마다 규칙 노드가 만들어지지만, 많은 규칙이 있으면 분류를 결정하는 방식이 복잡해지고 이는 과적합으로 이어지기 쉽다. 즉 트리의 깊이(depth)가 깊어질수록 결정 트리의 예측 성능이 저하될 가능성이 높다.



결정 노드는 정보 균일도가 높은 데이터 세트를 먼저 선택할 수 있도록 규칙 조건을 만든다. 정보의 균일도를 측정하는 대표적인 방법은 정보 이득 지수와 지니 계수가 있다.

- 정보 이득은 엔트로피 개념을 기반으로 한다. 엔트로피는 주어진 데이터 집합의 혼잡도를 의미하는데, 서로 다른 값이 섞여 있으면 엔트로피가 높고, 같은 값이 섞여 있으면 엔트로피가 낮다. 정보 이득 지수는 *1 - 엔트로피 지수* 이다.
- 지니 계수는 경제학에서 불평등 지수를 나타낼 때 사용하는 계수로, 0이 가장 평등하고 1로 갈수록 불평등하다. 머신러닝에서는 지니 계수가 낮을수록 데이터 균일도가 높은 것으로 해석한다.

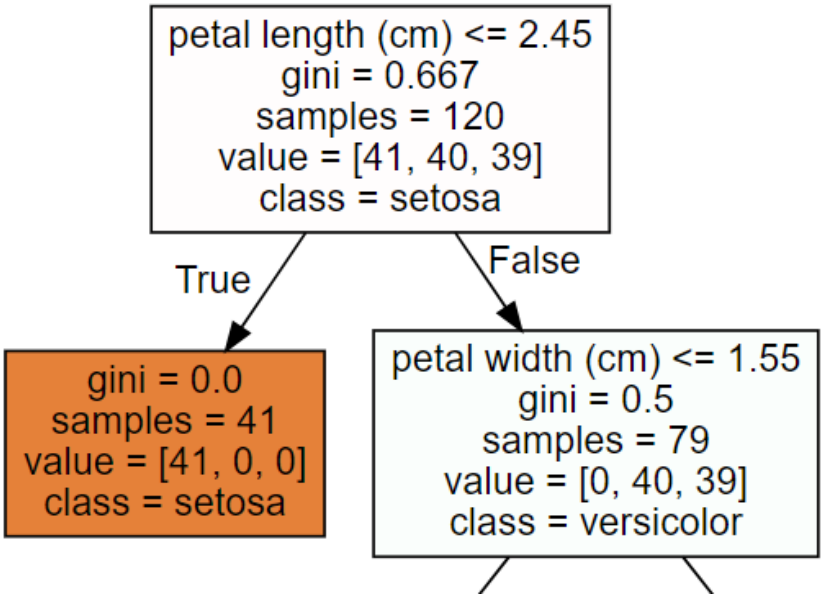
2-2. 결정 트리 파라미터

[사이킷런]

- DecisionTreeClassifier : 분류
  - `min_sample_split` : 노드 분할 위한 최소한의 샘플 데이터 수 (default = 2)
  - `min_sample_leaf` : 분할이 될 경우 왼/오 브랜치 노드에서 가져야할 최소 샘플 데이터 수
  - `max_features` : 최대 피처 개수 (default = None)
  - `max_depth` : 최대 깊이 규정
  - `max_leak_nodes` : 말단 노드 최대 개수
- DecisionTreeRegressor : 회귀
- 결정 트리 구현 : CART (Classification And Regression Tree) 알고리즘 기

2-3. 결정 트리 모델의 시각화

- Graphviz 패키지
  - `export_graphviz()`



- petal length(cm): 피처의 조건이 있는 것은 자식 노드를 만들기 위한 규칙 조건, 조건이 없으면 리프 노드
- gini: 다음의 value=[]로 주어진 데이터 분포에서의 지니 계수
- samples: 현 규칙에 해당하는 데이터 건수
- value: 클래스 값 기반의 데이터 건수, [41, 40, 39]는 Setosa 41개, Vesicolor 40개, Virginica 39개

- `feature_importances_` : 결정 트리 알고리즘이 어떻게 동작하는지 직관적 이해가능

2-4. 결정 트리 과적합(Overfitting)

- `make_classification()` : 분류를 위한 테스트용 데이터 쉽게 만들 수 있도록
- `visualize_boundary()` : 모델이 클래스 값을 예측하는 결정 기준을 색상과 경계로 나타내 모델이 어떻게 데이터 세트를 예측 분류하는지 !

```
import numpy as np

# Classifier의 Decision Boundary를 시각화 하는 함수
def visualize_boundary(model, X, y):
    fig,ax = plt.subplots()

    # 학습 데이타 scatter plot으로 나타내기
    ax.scatter(X[:, 0], X[:, 1], c=y, s=25, cmap='rainbow', edgecolor='k',
               clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim_start , xlim_end = ax.get_xlim()
    ylim_start , ylim_end = ax.get_ylim()

    # 호출 파라미터로 들어온 training 데이타로 model 학습 .
    model.fit(X, y)
    # meshgrid 형태인 모든 좌표값으로 예측 수행.
    xx, yy = np.meshgrid(np.linspace(xlim_start,xlim_end, num=200),np.linspace(ylim_start,ylim_end, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    # contourf() 를 이용하여 class boundary 를 visualization 수행.
    n_classes = len(np.unique(y))
    contours = ax.contourf(xx, yy, Z, alpha=0.3,
                           levels=np.arange(n_classes + 1) - 0.5,
                           cmap='rainbow', clim=(y.min(), y.max()),
                           zorder=1)
```

3️⃣ 앙상블 학습

3-1. 앙상블 학습 개요

앙상블은 여러 개의 분류기를 생성하고 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법이다.

[앙상블 학습의 유형]

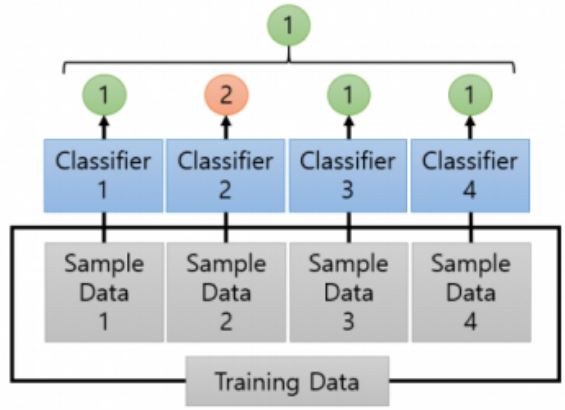
- **보팅(Voting)**
- **배깅(Bagging)**
- **부스팅(Boosting)**

보팅과 배깅은 여러 개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식인데, 차이점은 보팅은 서로 다른 알고리즘을 가진 분류기를 결합하는 것이고, 배깅은 각각의 분류기가 모두 같은 유형의 알고리즘 기반이지만, 데이터 샘플링을 서로 다르게 가져가면서 학습을 수행해 보팅을 수행한다. 대표적인 배깅은 랜덤 포레스트 알고리즘이다.

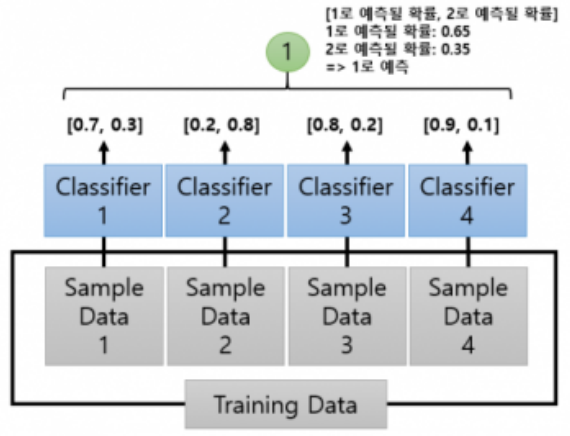
부스팅은 여러 개의 분류기가 순차적으로 학습을 수행하되, 예측이 틀린 데이터에 대해 올바르게 예측하도록 가중치를 부여하면서 학습과 예측을 진행하는 것이다.

3-1. 보팅 유형 - 하드 보팅과 소프트 보팅

일반적으호 하드 보팅보다는 소프트 보팅이 예측 성능이 좋음



Hard Voting



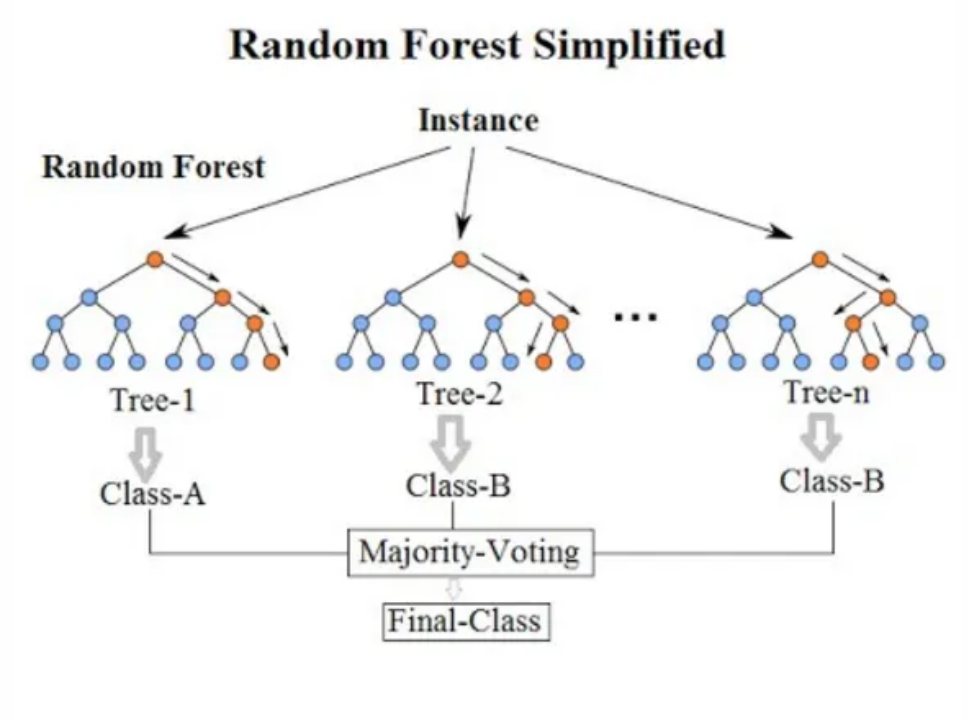
Soft Voting

- Hard Voting : 예측한 결괏값들중 다수의 분류기가 결정한 예측값을 최종 보팅 결괏값으로 선정 (다수결의 원칙과 유사)
- Soft Voting : 분류기들의 레이블 값 결정 확률을 모두 더해 이를 평균내서 확률이 가장 높은 레이블 값을 최종 보팅 결괏값을 선정

4️⃣ 랜덤 포레스트

4-1. 랜덤 포레스트의 개요 및 실습

랜덤 포레스트는 여러 개의 결정 트리 분류기가 전체 데이터에서 배깅 방식으로 각자의 데이터를 샘플링해 개별적으로 학습을 수행한 뒤 최종적으로 모든 분류기가 보팅을 통해 예측 결정을 한다.



- RandomForestClassifier 클래스 사용

랜덤 포레스트에서 전체 데이터에서 일부가 중첩되게 샘플링해서 데이터 세트를 만드는데 이를 **부트스트래핑 (bootstrapping)** 분할 방식이라 한다. 파라미터 *n_estimators*= 에 넣는 숫자만큼 데이터 세트를 만든다.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

X_train, X_test, y_train, y_test = get_human_dataset()

rf_clf = RandomForestClassifier(random_state=0, max_depth=8)
rf_clf.fit(X_train, y_train)
pred = rf_clf.predict(X_test)
accuracy = accuracy_score(y_test, pred)
print('랜덤 포레스트 정확도:{0:.4f}'.format(accuracy))
```

4-2. 랜덤 포레스트 하이퍼 파라미터 및 튜닝

- `n_estimators` : 랜덤 포레스트에서 결정 트리의 갯수를 지정(디폴트는 10개)
- `max_features` : 결정 트리에 사용된 max_features 파라미터와 같지만, RandomForestClassifier의 기본 max_features는 None이 아닌 auto, 즉 sqrt와 같다. 예를 들어 전체 피처가 16개라면 4개 참조
- `max_depth`나 `min_samples_leaf` 와 같이 결정 트리에서 과적합을 개선하기 위해 사용되는 파라미터가 랜덤 포레스트에서도 똑같이 적용될 수 있다.

5️⃣ GBM

5-1. GBM의 개요 및 실습

5-2. GBM 하이퍼 파라미터 소개

5-3. XGBoost 개요