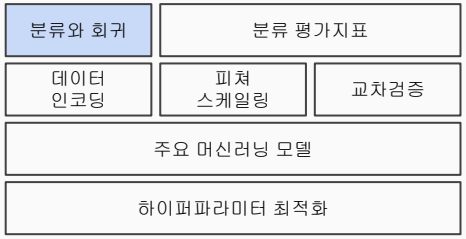
# 5장. 다시 살펴보는 머신러닝 주요 개념

* 머신러닝-딥러닝 문제해결 전략 (GOLDEN RABBIT)

# Index


분류와 회귀
분류 평가지표
⇒ 문제 유형과 평가지표
데이터 인코딩
피쳐 스케일링
교차검증
⇒ 데이터
주요 머신러닝 모델
⇒ 모델
하이퍼파라미터 최적화
⇒ 모델의 하이퍼파라미터

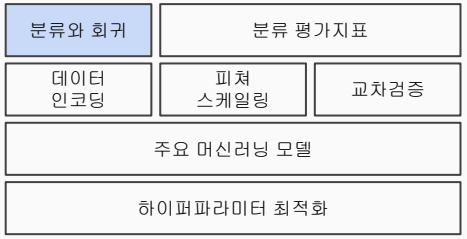
# 5.1 분류와 회귀




- 캐글 경진대회 대부분은 분류나 회귀 문제
  - 예측하려는 Target 값이 **범주형**이면 → 분류문제
  - 예측하려는 Target 값이 **수치형**이면 → 회귀문제
- 분류(Classification)
  - 예측하려는 범주형 데이터는 객관식 문제처럼 선택지가 있는 값 (유한한 선택지)
  - 예. 개/고양이를 구분, 스팸 메일/일반 메일 구분, 질병 검사 결과 양성/음성 구분
- 회귀(Regression)
  - 독립변수(X)와 종속변수(Y) 간 관계를 모델링하는 방법으로 종속변수는 수치형 데이터
  - 독립변수(Feature)와 종속변수(Target) 관계를 나타내는 최적의 회귀계수( $\theta$ ) 를 추정
  - 최적 회귀계수 구하려면 오차(예측값과 실제값의 차이)를 최소화해야함.

$$Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

# 5.1 분류와 회귀

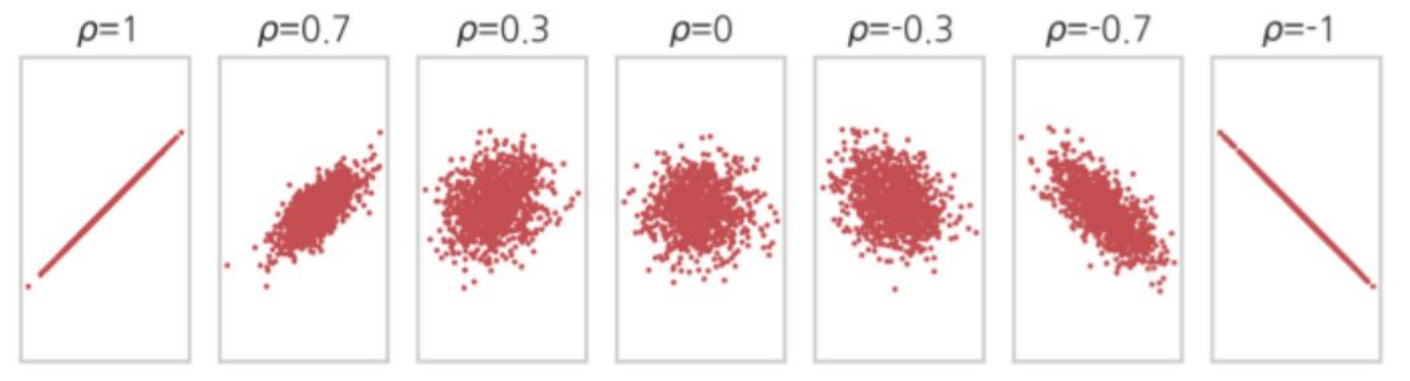



- 회귀 평가지표
  - MAE(Mean Absolute Error) $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$
    - 실제값과 예측값의 차이를 절대값으로 변환해 평균
  - MSE(Mean Squared Error) $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$
    - 실제값과 예측값의 차이를 제곱해 평균
  - RMSE(Root Mean Squared Error) $RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$
    - MSE값은 오류의 제곱을 구하므로 실제 오류 평균보다 커져 루트를 사용
  - MSLE(Mean Squared Log Error) $L(y,\hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(log(y_i + 1) - log(\hat{y}_i + 1))^2$
    - MSE에 log 사용, 일부 큰 오류값으로 인해 전체 오류가 커지는 것을 방지
  - RMSLE(Root Mean Squared Log Error) $\mathrm{RMSLE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$
    - RMSE에 log 사용, 일부 큰 오류값으로 인해 전체 오류가 커지는 것을 방지
  - $R^2$
    - $R^2$ = 예측값 Variance / 실제값 Variance
    - 실제값의 분산 대비 예측값의 분산 비율, 1에 가까울 수록 좋음

# 5.1 분류와 회귀



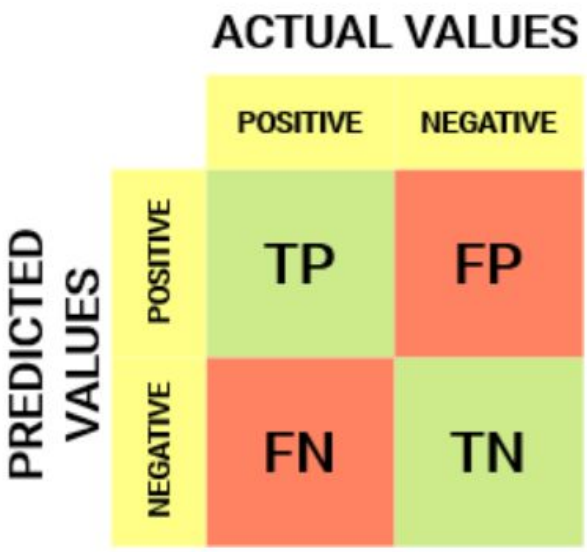
- 상관계수
  - 두 변수 사이의 상관관계(Correlation)를 수치로 나타낸 값



# 5.2 분류 평가지표



- 오차행렬 (Confusion Matrix)
  - 실제값과 예측값이 어떻게 매칭되는 지 보여주는 표
  - True Positive : 양성이라고해서 맞춤
  - True Negative : 음성이라고해서 맞춤
  - False Positive : 양성이라고해서 틀림
  - False Negative : 음성이라고해서 틀림
- 정확도 (Accuracy) $$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
  - 실제값과 예측값이 얼마나 일치하는지 비율로 나타냄
  - 분류 평가지표로 적절하지 않음 (예. 비 예측)
- 정밀도 (Precision) $$Precision = \frac{TP}{TP + FP}$$
  - 양성이라고 말한 것 중에 실제 양성의 비율



# 5.2 분류 평가지표

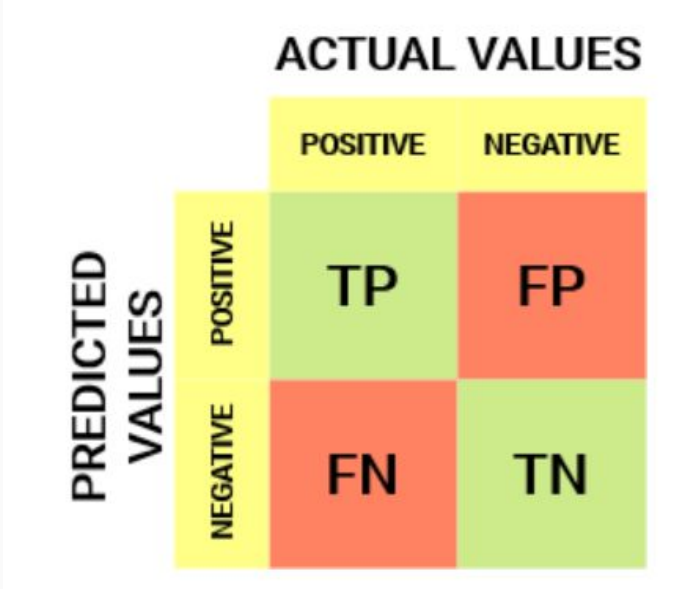
- 재현율 (Recall)

$$Recall = \frac{TP}{TP + FN}$$

  - 실제가 양성인 것 중 양성이라고 잘 예측한 값의 비율
- F1 점수 (F1-score)
  - 정밀도와 재현율의 조화평균

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

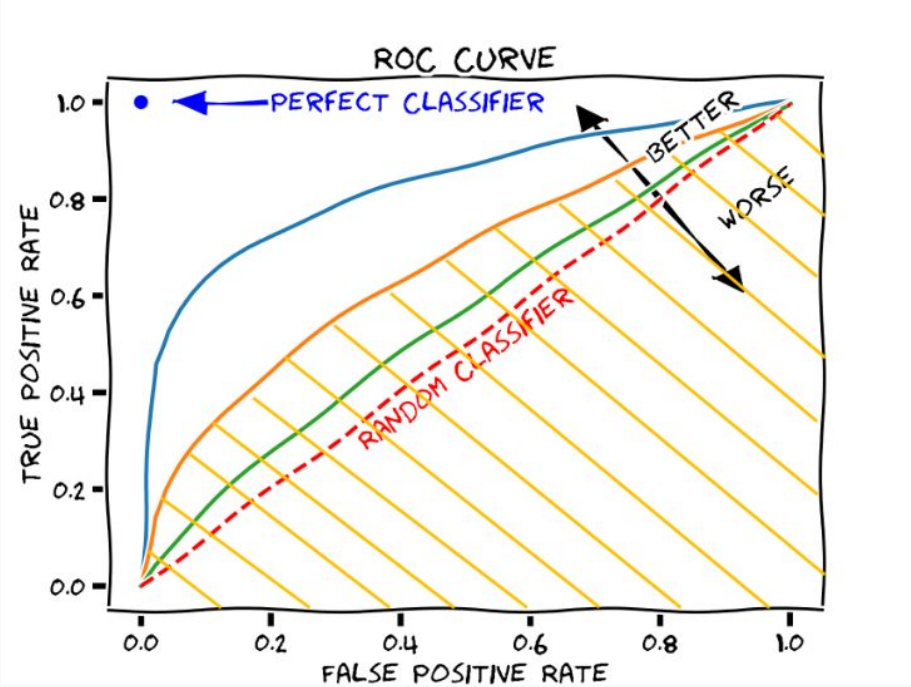




# 5.2 분류 평가지표



- ROC곡선과 AUC



# 5.3 데이터 인코딩

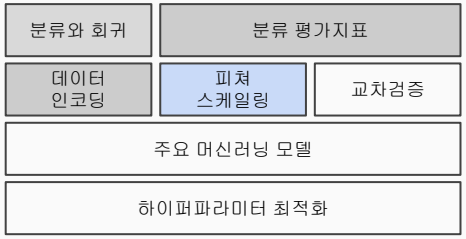

- 데이터 인코딩
  - 머신러닝은 문자 데이터를 인식하지 못함 → 숫자 데이터로 변환
- 레이블 인코딩
  - 범주형 데이터를 숫자로 일대일 매핑
  - 단점 : 머신러닝은 숫자가 가까운 데이터를 비슷한 데이터로 판단함 → 성능 문제 가능함.

```
encoder LabelEncoder()
encoder 결과 [0 1 4 5 3 3 2 2]
**************************************************
decoder 결과 ['TV' '냉장고' '전자렌지' '컴퓨터' '선풍기' '선풍기' '믹서' '믹서']
```

- 원-핫 인코딩
  - 피처 수만큼 열 추가 후 각 고유값에 해당하는 열을 1로 표시
  - 단점 : 피처 수가 많아지면 메모리 사용량 증가 → 모델 훈련 속도 느려짐

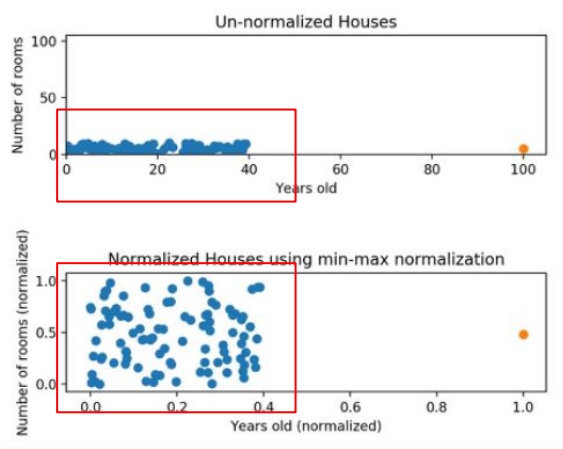
```
type <class 'numpy.ndarray'>
[[0]
 [1]
 [4]
 [5]
 [3]
 [3]
 [2]
 [2]]
(8, 1)
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0.]]
(8, 6)
```

# 5.4 피처 스케일링




- 피처 스케일링 (Feature Scaling)
  - 각 피처 값의 범위가 다르면 모델 훈련이 제대로 안될 수 있음(편향)
    → 같은 범위로 조정할 필요 있음.
- 정규화(Normalization)
  - min-max 정규화
    - 피처의 min, max를 이용
    - [0, 1]로 조정

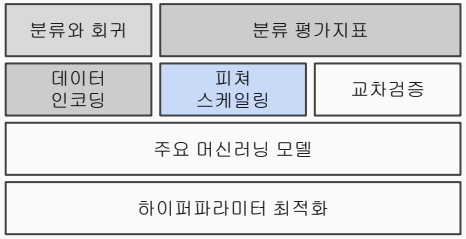
$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$



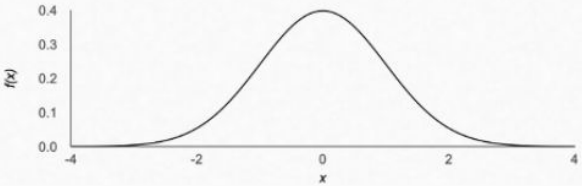

| | 키 | 몸무게 |
|---|---|---|
| 광일 | 1.7 | 75 |
| 혜성 | 1.5 | 55 |
| 덕수 | 1.8 | 60 |

```
[[0.66666667 1.        ]
 [0.         0.        ]
 [1.         0.25      ]]
```

# 5.4 피처 스케일링




- 표준화(Standardization)
  - 평균 0, 분산 1이 되도록 피처값을 조정 (범위 없음)
  - 머신러닝(SVM, Linear Regression 등)은 데이터가 가우시안 분포를 가지고 있다고 가정하고 구현 → 표준화가 모델의 예측 성능 향상에 중요




$$x_{new} = \frac{x - \mu}{\sigma}$$

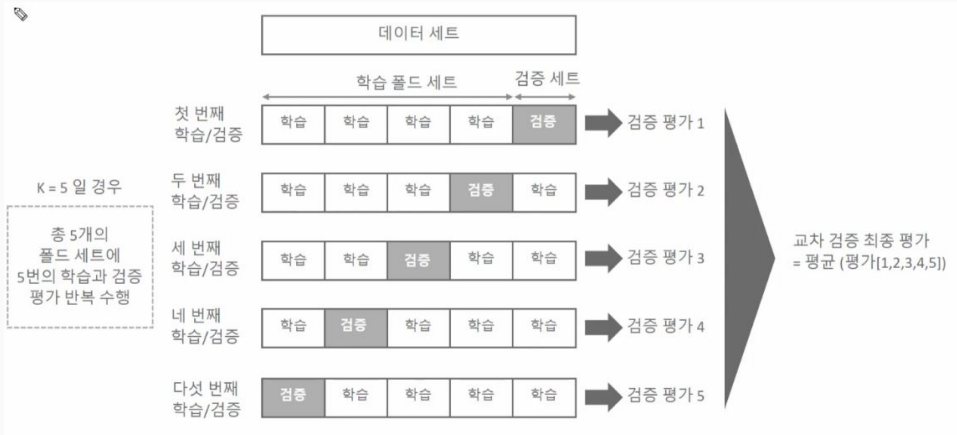
| | 키 | 몸무게 |
|---|---|---|
| 광일 | 1.7 | 75 |
| 혜성 | 1.5 | 55 |
| 덕수 | 1.8 | 60 |

```
[[ 0.26726124  1.37281295]
 [-1.33630621 -0.98058068]
 [ 1.06904497 -0.39223227]]
```

# 5.5 교차검증


- 학습만 하고 검증을 안하면
  - 과대적합
  - 사전에 모델 성능 확인 어려움
    - (학습:검증)으로 데이터를 나누면 손해 ??

→ 해결: 교차 검증(Cross Validation)



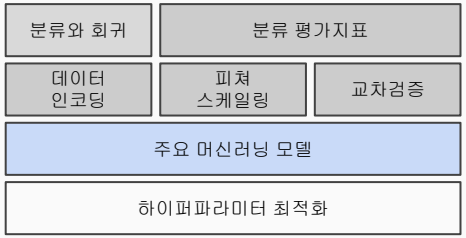
# 5.5 교차검증


- 층화 K 폴드 교차 검증(Stratified K-Fold CV)
  - Target 값이 불균형일 때 균등한 분포가 되도록 폴드를 나눔.
  - 회귀 문제는 연속된 값이라서 사용 어려움.

```
Fold 1 검증 데이터 타깃 값:
['스팸' '스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 2 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 3 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 4 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 5 검증 데이터 타깃 값:
['일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']
```

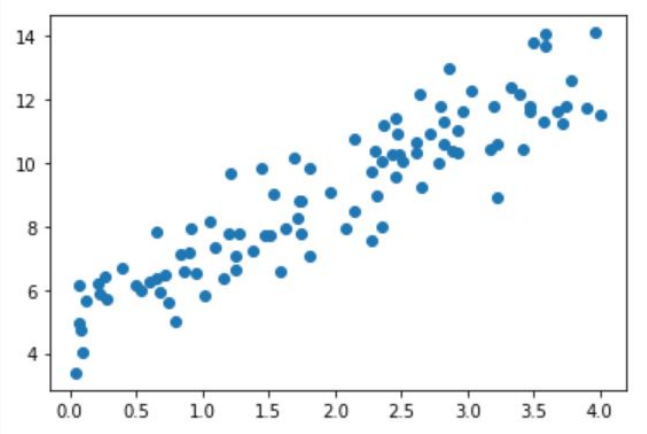
```
Fold 1 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 2 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 3 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 4 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']

Fold 5 검증 데이터 타깃 값:
['스팸' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반' '일반']
```

# 5.6 주요 머신러닝 모델



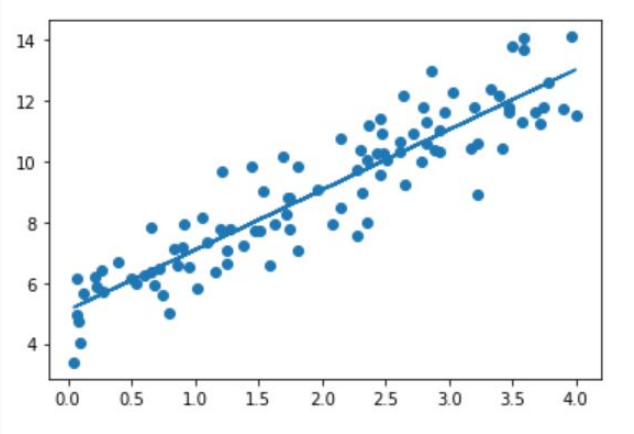

- 선형 회귀 (Linear Regression)
  - 학습 데이터에 잘 맞는 회귀 계수를 찾기

```
w0 = 5 # y절편
w1 = 2 # 회귀 계수
noise = np.random.randn(100, 1) # 노이즈
```

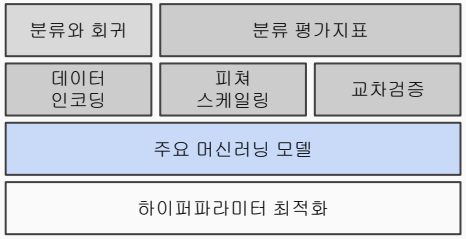



```
print('y절편(w0):', linear_reg_model.intercept_)
print('회귀계수(w1):', linear_reg_model.coef_)
```

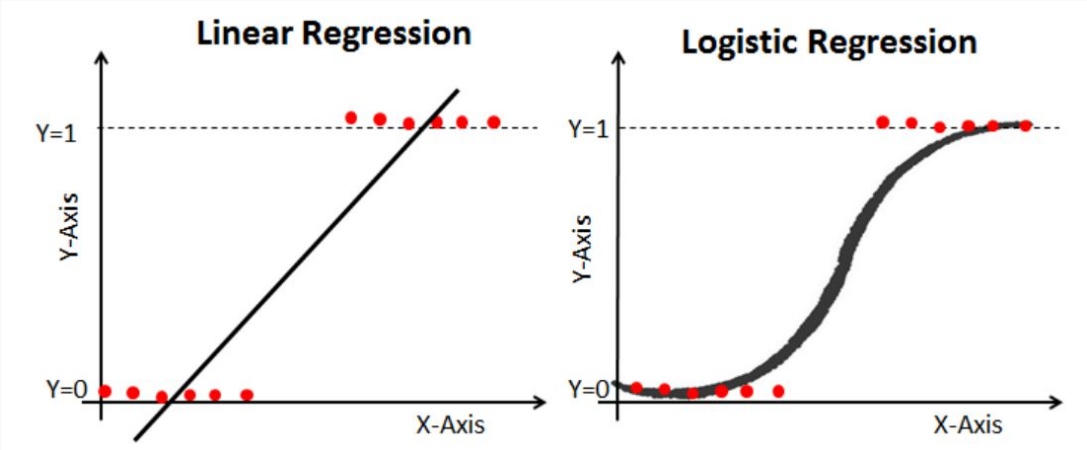
```
y절편(w0): [5.09772262]
회귀계수(w1): [[1.9808382]]
```



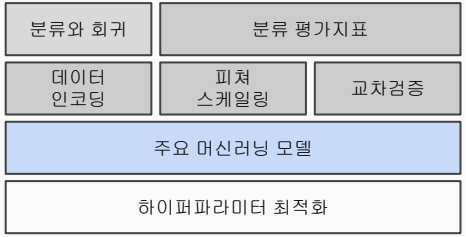
# 5.6 주요 머신러닝 모델



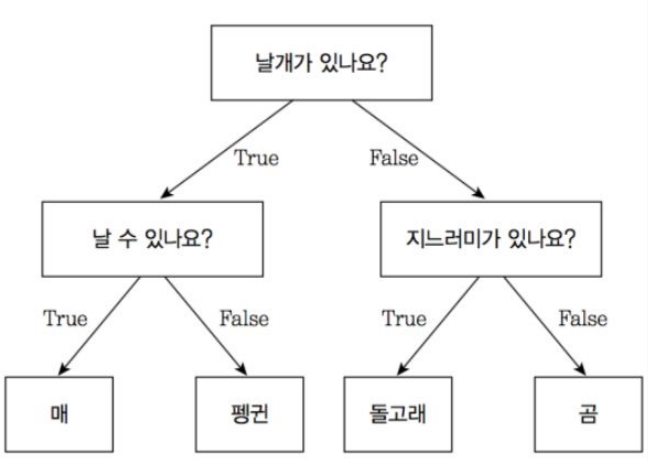

- 로지스틱 회귀 (Logistic Regression)
  - 선형 회귀 방식을 응용한 분류 모델
  - 시그모이드 함수를 활용해 Target값에 포함될 확률을 예측

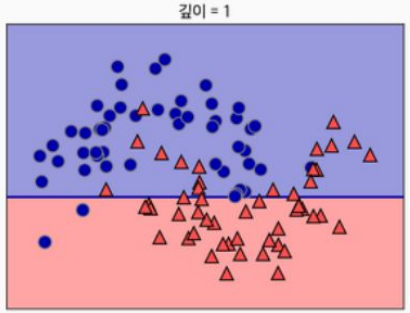


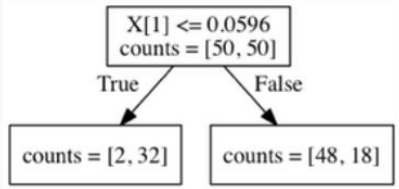
# 5.6 주요 머신러닝 모델

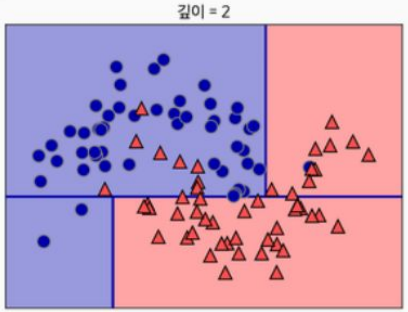


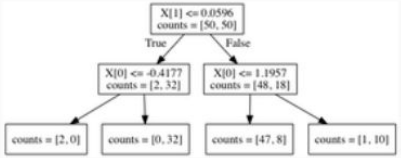

- 결정 트리 (Decision Tree)
  - 질문에 따라 노드를 구분하는 모델
  - 분류, 회귀 문제 모두 가능



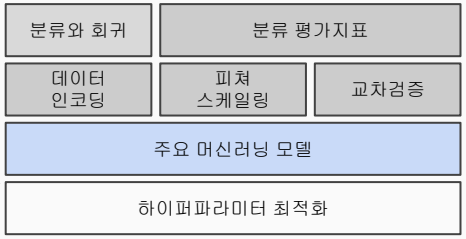



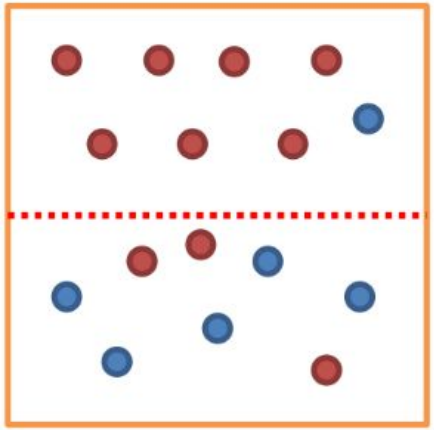



# 5.6 주요 머신러닝 모델




- 결정 트리 (Decision Tree)
  - 불순도(Impurity): 해당 범주 안에 서로 다른 데이터가 섞여 있는 정도
  - 엔트로피(Entropy): 불확실한 정도. 데이터 비율이 비등하면 1
  - 결정 트리는 엔트로피를 최소화하는 방향으로 노드를 분할



$$\text{Entropy} = -\sum_i (p_i) \log_2 (p_i)$$

# 5.6 주요 머신러닝 모델

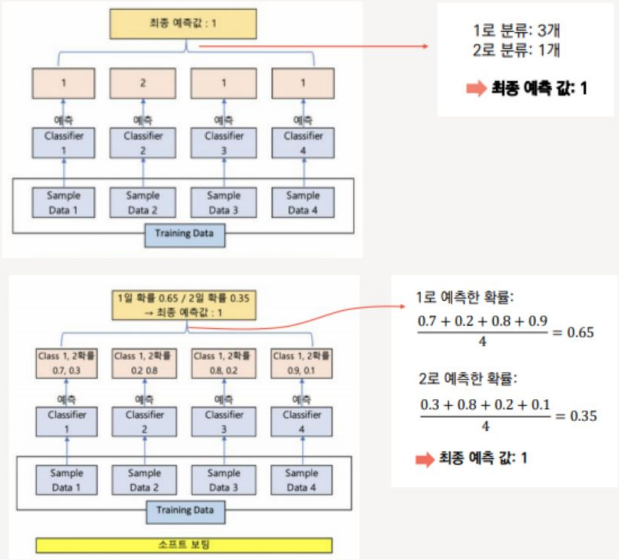


- 앙상블(Ensemble)
  - 다양한 모델이 내린 예측 결과를 결합하는 기법
  - 보팅(Voting)
    - 서로 다른 모델이 예측한 결과를 종합해 최종 결과를 결정 (하드보팅, 소프트보팅)

- Keyword: **집단지성**
- 여러 개의 다양한 모델을 만들고 합쳐서 에러를 줄이는 것!

<

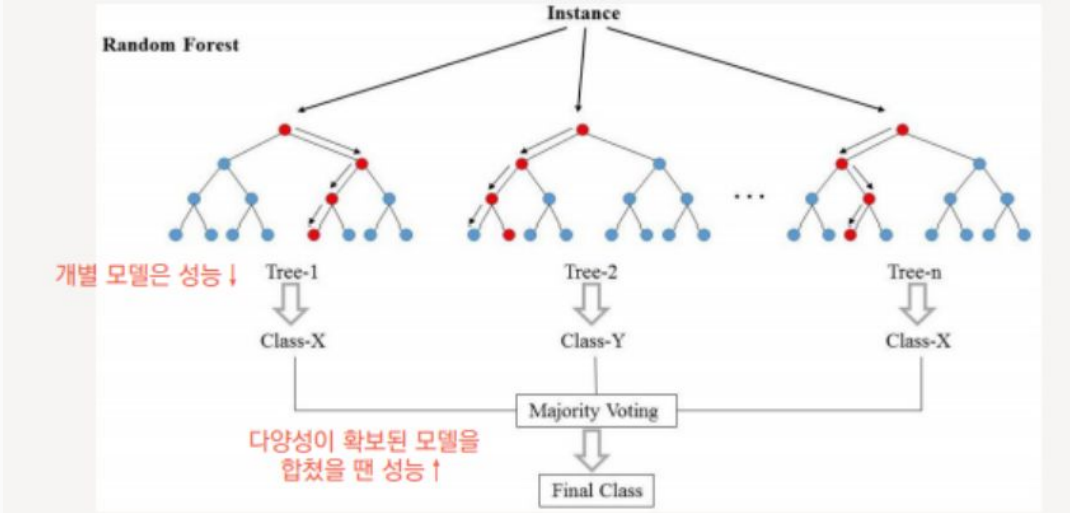
✓ 서울대 교수 한 명보다 다양한 학생들의 조합이 퀴즈를 더 잘 맞출 수 있다.



# 5.6 주요 머신러닝 모델



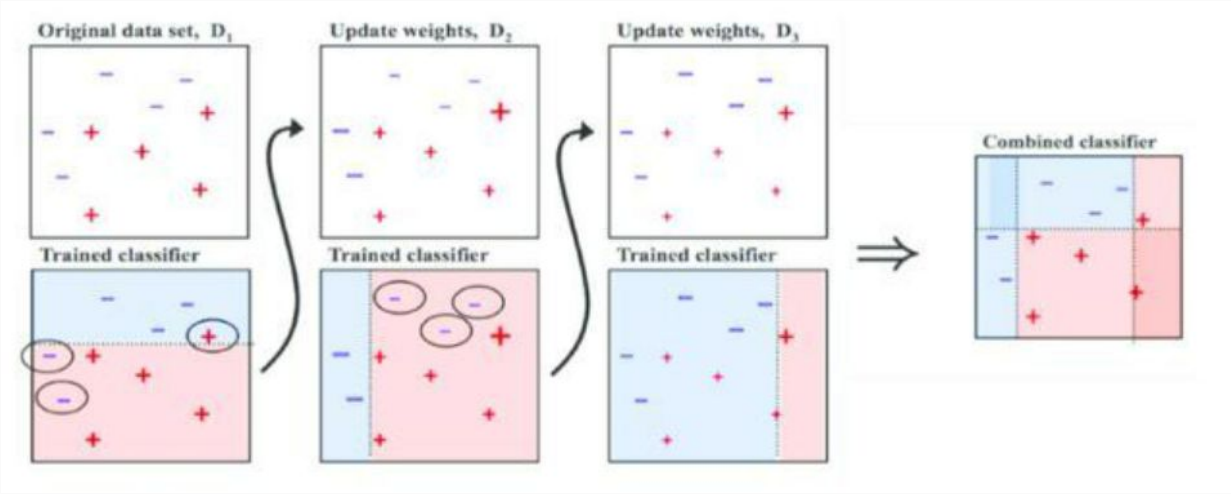
- 앙상블(Ensemble)
  - 배깅(Bagging)
    - 같은 모델에서 여러 개의 분류기를 만들어서 보팅
    - 랜덤 포레스트 (Random Forest)

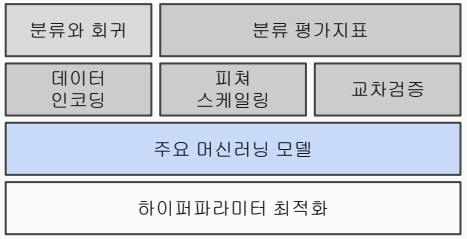
# 5.6 주요 머신러닝 모델



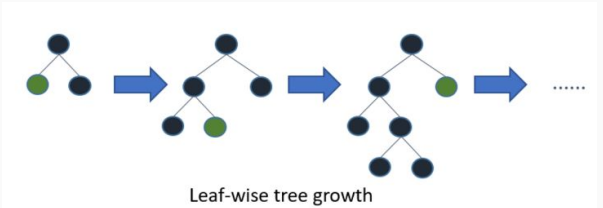
- 앙상블(Ensemble)
  - 부스팅(Boosting)
    - 여러개의 약한 모델을 순차적으로 학습/예측하며,
      잘못 예측된 데이터에 가중치부여를 하여 오류를 개선
    - XGBoost(extreame gradient boosting)



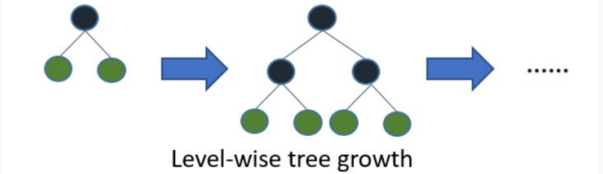
# 5.6 주요 머신러닝 모델




- 앙상블(Ensemble)
  - 부스팅(Boosting)
    - LightGBM: 트리의 균형을 맞추지 않고 최대 손실 값을 갖는 리프 노드를 지속적으로 분할하면서 깊고 비대칭적인 트리를 생성.
      → 속도가 빠름.
      → 불균형 트리로 인해 과적합 우려




*Light GBM 작동 방식*




*다른 Boosting 알고리즘 작동 방식*

# 5.7 하이퍼파라미터 최적화

- 최적 성능을 낼 수 있는 하이퍼파라미터를 찾아야함!
- 최적화 방법
  - 그리드서치(Grid Search)
    - 하이퍼파라미터를 모두 순회하면서 가장 좋은 성능을 내는 값을 찾음→ 오래걸림
  - 랜덤서치(Random Search)
    - 무작위로 찾아내 사용빈도가 떨어짐?
    - 특정 논문에서 랜덤서치가 더 성능이 좋다고?*
  - 베이지안 최적화(Bayesian Optimization)
    - 사전 정보를 바탕으로 확률적으로 추정하는 기법.
    - 평가지표 함수를 통해 최적 하이퍼파라미터를 찾는다.
    - https://data-scientist-brian-kim.tistory.com/88

*https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf