Cross Validation

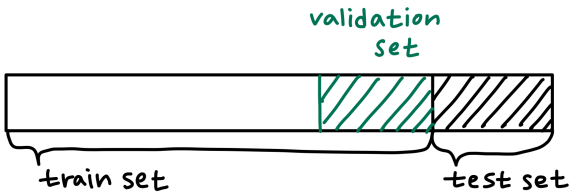
데이터를 train set과 test set으로 나눈 후, train set을 다시 train set과 validation set으로 나눔.

- train set: 모델을 학습시킴
- validation set: 모델 학습 과정에서 모델의 성능을 평가하여 최적의 hyper-parameter를 선택할 수 있도록 함
- test set: 모델의 성능을 최종 평가

  overfitting 및 underfitting을 방지할 수 있다는 장점이 있지만, 모델 훈련 및 검증 소요시간이 증가한다는 단점이 있음.

a. Hold-out CV

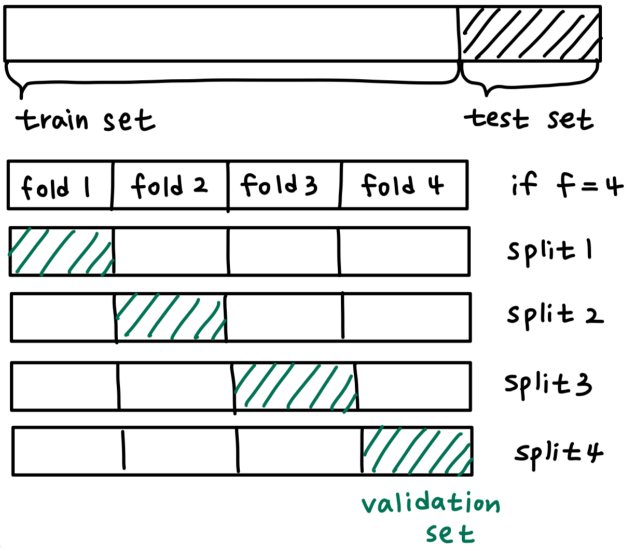
특정 비율로 1회 분할



b. K-fold CV

train set을 k개의 fold로 나누고, 그 중 1개의 fold를 선택해 validation set으로 활용

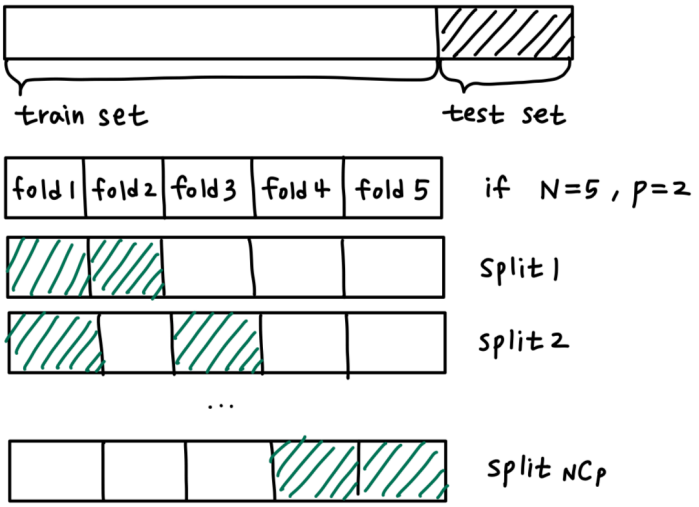
- k개의 성능 결과의 평균을 해당 모델의 성능으로 봄



c. Leave-p-Out CV

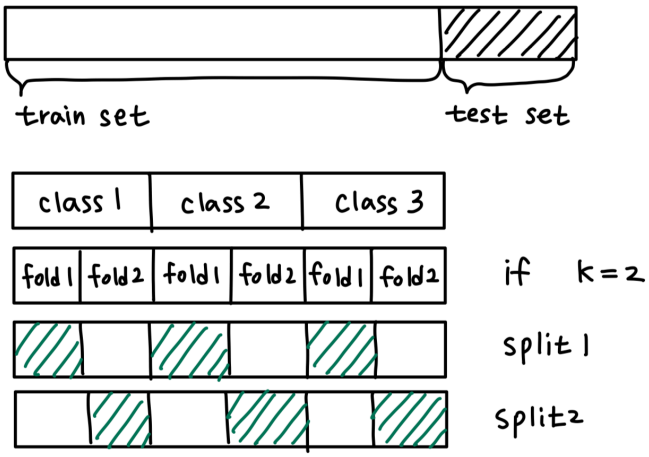
train set을 N개의 fold로 나누고, 그 중 p개의 fold를 선택해 validation set으로 활용

- p=1인 경우, Leav-One-Out CV라고 불리며, K-fold CV에서 K=N인 경우에 해당함
- NCp개의 성능 결과의 평균을 해당 모델의 성능으로 봄



d. Stratified K-fold CV

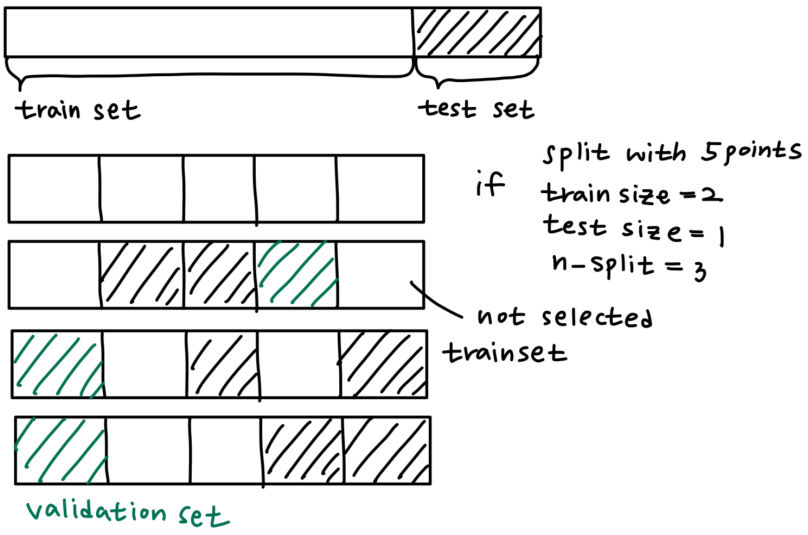
train set을 k개의 fold로 나누고, 그 중 1개의 fold를 선택해 validation set으로 활용. 단, 각 fold는 전체 data set의 클래스 분포와 유사한 클래스 분포를 갖도록 함



e. Shuffle Split CV

데이터를 무작위로 섞은 후, 일정 비율로 train set과 validation set을 여러 번 반복에서 나눔

- fold의 개수를 미리 설정할 필요 없음



Core Concepts in ML Models

- hypothesis function: 회귀 모형이 학습으로 찾은 함수
- cost function: 예측 결과의 정확도를 판단하는 함수
    - 주로 Mean Squared Error MSE를 사용
- overfitting을 방지하기 위한 방법
    a. reduce number of features
    b. Regularization
    λ값을 크게 하면 회귀 계수 w의 값을 작게 하도록 유도할 수 있음
        a. Lasso Regularization = L1 Regularization
        가중치의 절댓값의 합을 cost function에 추가함
        - $Cost\ Function = Loss\ Function + \lambda \sum_i |w_i|$

        b. Ridge Regularization = L2 Regularization
        가중치의 제곱의 합을 cost function에 추가함
        - $Cost\ Function = Loss\ Function + \lambda \sum_i w_i^2$

        c. Elastic Net Regularization
        L1과 L2를 결합
        - $Cost\ Function = Loss\ Function + \lambda_1 \sum_i |w_i| + \lambda_2 \sum_i w_i^2$
- sampling
    classification 문제에서 data set의 클래스가 불균형하다면, over-sampling 또는 under-sampling할 수 있음
    - over-sampling
        - minority class sample 수를 늘려서 데이터의 불균형을 해소함
        - random over-sampling: minority class의 데이터를 단순히 복제하는 방법
        - Synthetic Minority Over-sampling TEchnique SMOTE: minority class의 데이터를 기반으로 새로운 가상 데이터를 생성하는 방법
    - under-sampling
        - majority class sample 수를 줄여서 데이터의 불균형을 해소함
        - random under-sampling: majority class의 데이터를 무작위로 선택하여 삭제하는 방법
        - NearMiss: majority class의 데이터 중에서 minority class와 가까운(유사한) 데이터만 선택하는 방법

Regression Analysis

관찰된 연속형 변수들 사이 관계를 분석

a. Polynomial Regression
독립변수의 차수가 2차 이상인 회귀 모형

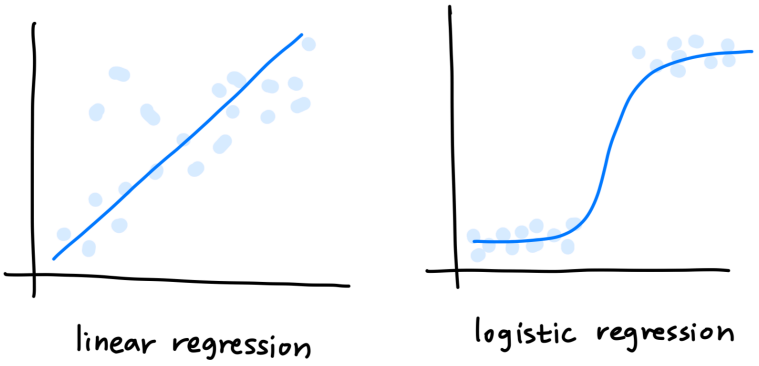
b. Multivariate Linear Regression

독립변수가 2개 이상인 회귀 모형

- 독립변수간 상관관계가 높아 발생하는 multicollinearity 처리가 필요
  multicollinearity는 variation inflation factor VIF로 확인 가능

Logistic Regression

Linear Regression의 classification 버전



- **sigmoid function = logistic function**
  - odds ratio
    - 어떤 임의의 사건 A가 발생하지 않을 확률 대비 발생할 확률의 비율
    - $odds = \frac{P(A)}{1-P(A)}$
  - logit function
    - odds ratio를 로그 변환한 것
    - $z = logit(odds) = log(\frac{P(A)}{1-P(A)})$
  - logistic function
    - logit function의 역함수
    - $g(z) = \frac{1}{1+e^{-z}}$

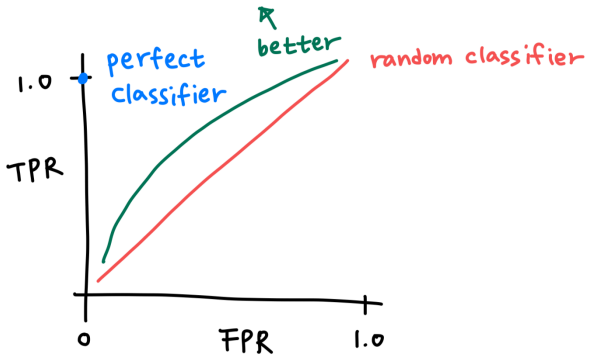
Gradient Descent

1. 임의의 점에서 시작
2. 현재 위치에서의 기울기를 계산
3. 경사의 반대 방향으로 이동
   - 경사가 양수라면 음수 방향으로, 경사가 음수라면 양수 방향으로 이동함
   - learning rate에 따라 한 번에 얼마나 이동할지 결정됨
     - learning rate가 너무 크다면 최적의 위치를 지나쳐버릴 수 있고, 너무 작다면 최적의 위치까지 도달하는 속도가 너무 느릴 수 있음
4. 반복

Receiver Operating Characteristic ROC

x축을 FPR, y축을 TPR로 하는 그래프

- false positive rate FPR: 실제 negative sample 중 positive로 예측한 비율
- true positive rate TPR: 실제 positive sample 중 positive로 예측한 비율



hyper-parameter

- parameter과의 차이점
  - parameter은 모델의 구성요소이자 데이터로부터 학습되는 것
    e.g. linear regression model y = ax + b에서 a와 b는 parameter
  - hyper-parameter은 모델 학습 과정에 반영되며, 학습 시작하기 전에 미리 결정해야 하는 것
    e.g. neural network model에서 learning rate, KNN에서 k값

a. Grid Search

모든 가능한 hyper-parameter 조합을 시도해보는 방법

- hyper-parameter 수와 값이 많아질수록 계산량이 기하급수적으로 증가해 비효율적이고 연산 비용이 크다는 단점이 있음

b. Random Search

무작위로 몇몇 조합을 선택해 평가하는 방법

- Grid Search보다 효율적이지만, 무작위 선택인 만큼, 최적의 조합이 선택되지 않을 수도 있고 반복할 때마다 다른 결과가 나올 수도 있음

c. Bayesian Optimization

1. 초기 탐색: 무작위로 몇몇 조합을 선택해 평가
2. 확률 모델 생성: 각 조합의 평가 결과를 바탕으로, hyper-parameter과 모델 성능 사이 관계를 학습하는 확률 모델(보통 gausian process)을 만듦
3. 최적의 다음 조합 예측: 2에서 만든 확률 모델을 활용해, 성능이 좋을 것 같은 조합을 우선적으로 선택
4. 반복: 새롭게 선택된 조합을 평가하고, 그 결과를 확률 모델에 반영

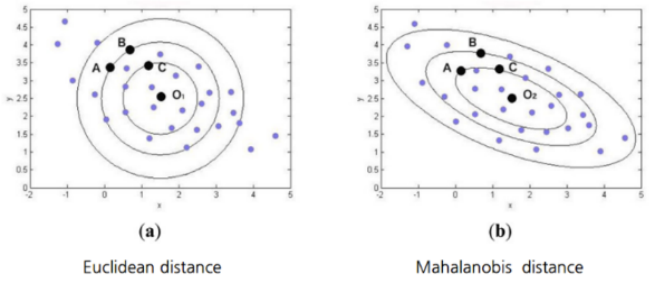
- 이해와 구현이 상대적으로 복잡하지만, 가장 효율적임

K-Nearest Neighbors

supervised learning, classification 및 regression algorithm.
새로운 데이터의 클래스나 값을 예측하는 데에 있어서 해당 데이터와 가장 가까운 거리에 있는 k개의 데이터를 활용하는 방법

- k
    - 너무 크면 data point 주변의 세부적인 특성을 파악하기 어려워져 underfitting
    - 너무 작으면 잡음 및 이상치를 모두 포함하게 되어 overfitting
    - CV를 통해 최적의 k를 선정하곤 함
- 예측하고자 하는 변수가
    - 범주형일 경우, k-nearest neighbors 중 가장 많이 나타나는 클래스로 추정. 이때 tie issue 방지하기 위해 k는 홀수로 설정하는 것이 좋을 것
    - 연속형일 경우, k-nearest neighbors의 대표값으로 추정.
- k-nearest neighbors 찾는 방법
    - brute force
        - 완전 탐색: 모든 data point 쌍에 대해 거리를 계산함
        - 거리 계산하는 방법
            - 범주형일 경우:
                - hamming distance $D_H = \sum_{j=1}^{J} I(x_j \neq y_j)$

                  e.g. 1011101과 1001001 사이 거리는 2
            - 연속형일 경우:
                - euclidian distance $D_E = \sqrt{\sum_{j=1}^{J} (x_j - y_j)^2}$
                - mahalanobis distance



(a) Euclidean distance (b) Mahalanobis distance

                - manhattan distance $D_M = \sum_{j=1}^{J} |x_j - y_j|^2$
    - k-dimensional tree
        - 이진 탐색: 각 노드는 데이터를 특정 축을 기준으로 분할

          e.g. 데이터를 첫 번째 노드는 x축을 기준으로, 두 번째 노드는 y축을 기준으로, 세 번째 노드는 z축을 기준으로 반으로 나누는 식
        - 각 노드의 분할 기준은 해당 축의 중간값을 활용
    - ball tree
        - k-d tree와 유사하나, 특정 축을 기준으로 분할하지 않고 각 data point의 중심점을 기준으로 구를 생성하여 데이터를 분할한다는 점이 다름

Feature Scaling

- **standardization**

각 feature를 평균이 0, 분산이 1이 되도록 변환 = (값 - 평균) / 표준편차

- **normalization**

  각 feature의 단위를 최소 0 ~ 최대 1로 변환

  a. min-max normalization

  $$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  b. z-score normalization

  $$X_{norm} = \frac{X-\mu}{\sigma}$$

  c. max abs normalization

  $$X_{norm} = \frac{X}{max(|X|)}$$

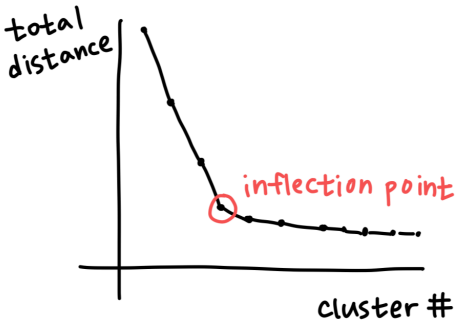
  d. robust scaler

  $$X_{norm} = \frac{X-median}{IQR}$$

- scaler 객체를 활용해 data의 scale 변환할 시 다음 세 method 사용함
  - fit() 데이터 변환을 위한 기준 정보 설정을 적용
  - transform() 설정된 정보를 이용해 데이터 변환
    - train set은 fit()과 transform() 모두 사용, test set은 transform()만 사용
  - fit_transform() 두 가지 한번에 적용

K-means Clustering

- 과정
  1. cluster 개수 k 설정, 초기 centroid 지정
  2. 각 개체에서 k개의 centroid까지의 거리 측정, 가장 가까운 centroid로 clustering
  3. 각 cluster의 평균 값으로 centroid 업데이트
  4. 1~3의 작업을 수렴할 때까지 반복
- 한계
  - cluster 개수를 미리 지정해야 함
  - cluster 모양을 원형으로 가정하기 때문에, 다양한 분포 형태를 띠는 데이터에는 적용하기 어려울 수 있음
- cluster 개수 k
  - elbow method
    - inertia가 급격히 떨어지는 때의 값을 k로 지정



  - silhouette coefficient

1. 특정 개체와 그 개체가 속한 cluster 내 데이터들의 평균 거리 a(i) 산출
2. 특정 개체와 그 개체가 속하지 않은 cluster 중 가장 가까운 cluster 내 데이터들의 평균 거리 b(i) 산출
3. silhouette coefficient $s(i) = \frac{b(i)-a(i)}{max[a(i), b(i)]}$

- silhouette coefficient가 1에 가까울수록 성공적인 clustering이라 평가할 수 있음

RFM

R, F, M feature로 고객을 이해 및 분석하자는 방법론

- R recency: 고객의 가장 최근 상품 구입일과 현재까지의 기간
- F frequency: 상품 구매 횟수
- M monetary: 고객의 총 주문 금액

Evaluation

- Accuaracy: 전체 샘플 중에서 정확하게 예측한 샘플의 비율
- Precision: 1이라고 예측한 샘플 중에서 진짜 1인 샘플의 비율
- F1-score: Accuracy와 Precision의 조화평균
- confusion matrix: 실제 클래스와 예측 클래스 간의 관계를 나타내는 행렬로, 다음과 같이 구성됨
    - TP true positive: 실제 클래스 1, 예측 클래스 1
    - FP false positive: 실제 클래스 0, 예측 클래스 1
    - FN false negative: 실제 클래스 1, 예측 클래스 0
    - TN true negative: 실제 클래스 0, 예측 클래스 0
- classification report: 각 클래스에 대한 Precision, Recall, F1-score을 포함하는 상세한 성능 평가 결과를 출력

Decision Tree

- recursive partitioning
    1. 초기화: 데이터를 하나의 집합으로 취급
    2. 분할 기준 선택: 특정 특성을 기준으로 데이터를 두 개의 하위 집합으로 나눔. 이때 분할 기준은 불순도를 최소화할 수 있는 방향으로 선택됨
    3. 반복: 각 하위 집합에 대해 다시 분할 기준을 찾고, 이를 반복적으로 수행
    4. 종료 조건: 데이터가 충분히 순수해지거나, 더 이상 분할할 수 없을 때 분할을 멈춤
- pruning

  생성된 모델의 복잡성을 줄이고, 과적합을 방지하기 위해 사용함
    - pre-pruning: 모델을 생성하는 과정에서 미리 가지치기를 수행하는 방법

      e.g. 트리의 최대 깊이, 각 노드에 남아있는 최소 데이터 수 등을 미리 정하여 모델의 성장을 제한함
    - post-pruning: 모델이 완전히 생성된 후, 불필요하게 복잡한 부분을 제거하는 방법

    - cost-complexity pruning: 가지치기 수행할 때 모델의 복잡도와 성능 사이의 균형을 찾기 위한 기법
- Decision Tree algorithm의 주요 변형
    a. **Classification and Regression Tree CART**
        - 분류와 회귀 모두에 사용할 수 있는 알고리즘
        - 범주형 변수와 수치형 변수 모두 처리할 수 있음
        - gini index를 사용해 데이터의 불순도를 측정
        - binary split: 각 노드는 두 개의 하위 노드로 나뉨
    b. **C4.5**
        - 분류와 회귀 모두에 사용할 수 있는 알고리즘
        - 범주형 변수와 수치형 변수 모두 처리할 수 있음
        - entropy 사용해 데이터의 불순도를 측정
        - multi-way split 허용: 하나의 노드는 여러 개의 하위 노드로 나뉠 수 있음
    c. **CHAID**
        - 분류에 사용할 수 있는 알고리즘
        - 범주형 변수만 처리할 수 있기 때문에, 수치형 변수를 사용할 경우 범주형으로 변환해야 함
        - Chi-square 통계량을 사용해 변수 간의 독립성을 검정하고, 이를 바탕으로 최적의 분할을 찾음
        - multi-way split 허용: 하나의 노드는 여러 개의 하위 노드로 나뉠 수 있음
- 불순도 측정하는 방법
  불순도는 주어진 데이터 집합에서 다양한 클래스가 얼마나 혼합되어 있는지 나타내는 지표
    a. **gini index**
       한 데이터 집합에서 두 개의 sample을 무작위로 뽑았을 때, 서로 다른 클래스에 속할 확률
        - $Gini = 1 - \sum_{i=1}^{n} p_i^2$
        - 0에서 0.5 사이의 값을 가지며, 0에 가까울수록 불순도가 낮음을 의미함
    b. **entropy**
        - $Entropy = -\sum_{i=1}^{n} p_i log_2(p_i)$
        - 값이 클수록 불순도가 높음을 의미함
    c. **chi-square** 통계량
        - 불순도를 직접적으로 나타내진 않지만, 각 변수가 얼마나 독립적인지 확인하는 것은 곧 불순도가 얼마나 높은지 확인하는 것과 같은 맥락

Random Forest

- Decision Tree와의 차이
    - Random Forest는 여러 개의 Decision Tree를 결합한 모델
- Ensemble Learning
    - Random Forest는 Ensemble Learning의 일종 (i.e. 여러 개의 약한 학습기를 결합해 더 강한 학습기를 만드는 방법)

a. Bagging

여러 개의 모델을 독립적으로 학습한 후, 이들의 예측 결과를 결합하여 최종 예측을 도출하는 방식

- Bootstarpping
    - 원본 data set에서 중복을 허용하여 sampling
    - 각 sample은 독립적으로 모델을 학습시킴
- Aggregating
    - 각 모델의 예측은 분류 문제의 경우 다수결, 회귀 문제의 경우 평균으로 결합함

b. Boosting

약한 학습기를 순차적으로 학습시키며, 이전 모델의 오류를 보완해 나가는 방식

- 첫 번째 모델이 학습된 후, 잘못 예측한 data point에 더 높은 가중치를 주고 다음 모델을 학습시킴
- overfitting의 위험이 있음

c. Stacking

서로 다른 종류의 모델의 예측 결과를 meta-model로 입력하여 최종 예측을 도출하는 방식