D&A

ML Session 5차시

Data Preprocessing.

2022 / 10 / 04

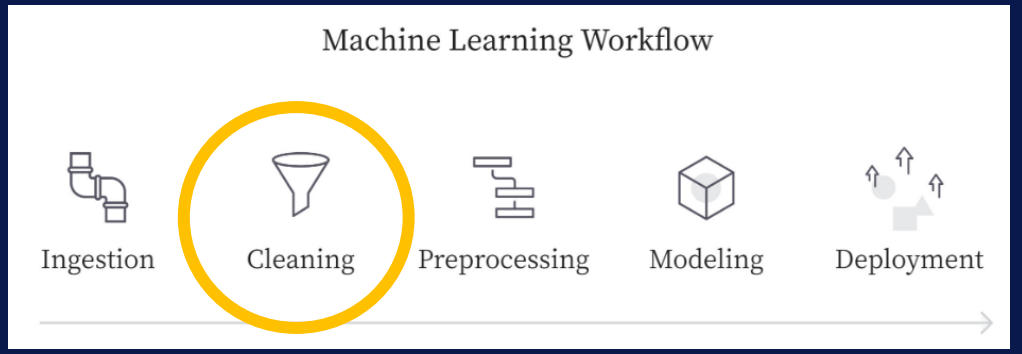
D&A 부학회장 김정하

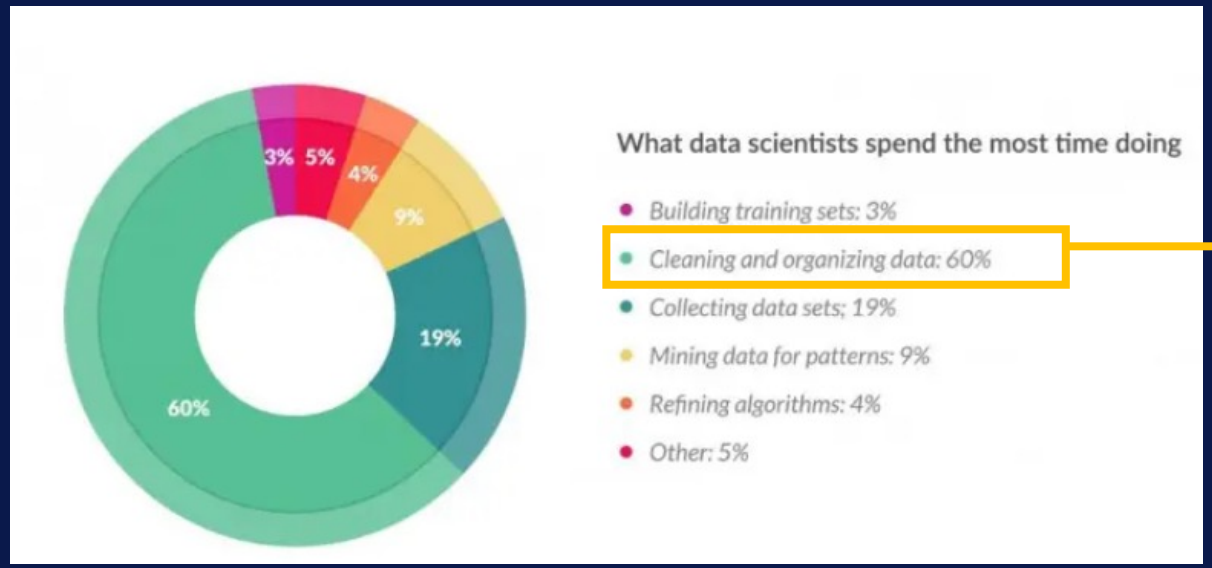
CONTENTS.

01. Data Cleansing

데이터 전처리란?

데이터 분석 전, 데이터를 분석에 적합한 형태로 처리하는 과정
머신러닝 알고리즘만큼 중요
동일한 머신러닝 기법을 적용해도 전처리에 따라 다른 결과가 나옴





Forbes에서 인용한 CrowdFlower의 설문 결과에 따르면,
데이터 분석가는 업무 시간 중 60%정도를
데이터 전처리 과정에 사용

01. Data Cleansing

Domain Understanding & Data Collection | **Data Preprocessing** | Modeling & Ensemble | Evaluation | Deployment

- Data Cleansing
- Data Intgration
- Feature Construction / Extraction
- Feature Selection
- Scaling
- Sampling

01. Data Cleansing

결측값 처리

결측값이 있는 상태로 모델을 만들 경우, 변수간의 관계가 왜곡될 수 있음
또한 사이킷런 패키지는 결측값이 있는 경우 사용할 수 없음

결측값이 발생하는 유형에 따라, 해당 Feature의 특성에 따라,
결측값을 올바르게 처리해야함

01. Data Cleansing

결측값 처리 – 삭제

주로 결측값이 무작위로 발생한 경우에 사용
무작위로 발생한 것이 아닌데 관측치를 삭제한 데이터를 사용할 경우, 왜곡된 모델이 생성될 수 있음

전체 삭제 – 결측값이 발생한 모든 관측치를 삭제
→ 간편한 반면, 관측치가 줄어들어 모델의 유효성이 낮아짐
부분 삭제 – 모델에 포함시킬 변수들 중 결측값이 발생한 모든 관측치 삭제
→ 모델에 따라 변수가 제각각 다르기 때문에 관리 Cost가 늘어남

01. Data Cleansing

결측값 처리 – 대체

한 가지 값으로 대체하는 경우

평균 : Column 내 값들의 평균으로 결측치 대체
연속형 변수만 사용 가능

중앙값 : Column 내 값들의 중앙값으로 결측치 대체
연속형 변수만 사용 가능

최빈값 : Column 내 값들 중 가장 많이 나온 값으로 결측치 대체
연속형, 범주형 모두 사용 가능

01. Data Cleansing

결측값 처리 – 대체

여러 가지 값으로 대체하는 경우

결측치가 아닌 데이터들을 train으로 두고 model을 돌려 값을 예측

KNN Imputation : KDTree를 구성한 후 최근접 이웃을 계산해
k-NN을 찾은 후 가중 평균 취함

MICE : Multiveriate Imputation by Chained Equations
연쇄 방정식을 이용한 대치
누락된 데이터를 여러 번 채우는 방식으로 작동

01. Data Cleansing

결측값 처리 – 가이드

적절성 확인

- 하나라도 결측이 있는 변수를 제외한 dataset 생성
- Imputation한 dataset 2개 정도 생성
- 위에서 생성한 dataset들이 서로 일관성이 있음을 보여줌
  → 결측치 대치 방법이 sensitive하지 않다는 것을 제시

가이드 라인

10% 미만 : 삭제 or 대치

10 ~ 20% : Hot deck, regression, model based imputation

20 ~ 50% : regression, model based imputation

50% 이상 : 해당 변수 제거

01. Data Cleansing

이상치 처리

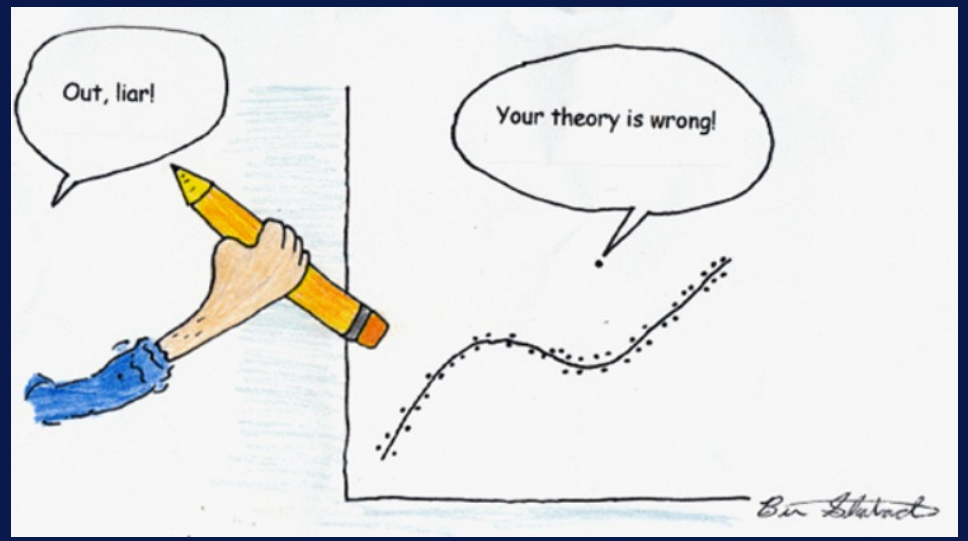
값의 범위가 일반적인 범위를 벗어나 특별한 값을 갖는 것
회귀모형의 경우, 이상치 값에 민감하게 반응

이상치 확인

- 시각화 : Boxplot, Histogram, Scatter plot
- 두 변수 간 회귀 모형에서,
  Residual, Studentized residual (or standardized residual),
  leverage, Cook's D 값 확인

이상치의 범위

- 표준점수로 변환
- IQR 방식
- DBScan
- 도메인 지식 이용이나 Bining 처리 방식



01. Data Cleansing

이상치 처리

- 표준점수로 변환
  표준정규분포로 변환 후 -3 이하 및 3 이상 값들을 이상치로 판단 후 제거 하거나 대체

- IQR 방식
  1사분위수보다 낮은 IQR의 1.5배를 벗어나는 포인트 or
  3사분위수보다 높은 IQR의 1.5배를 벗어나는 포인트
  이상치로 처리 (제거 or 대체)

01. Data Cleansing

Scaling이란?

변수의 단위를 변경하고 싶거나,
변수의 분포가 편향되어 있을 경우,
변수 간의 관계가 잘 드러나지 않는 경우

위와 같은 경우에 Scaling 수행

Scaling 방법

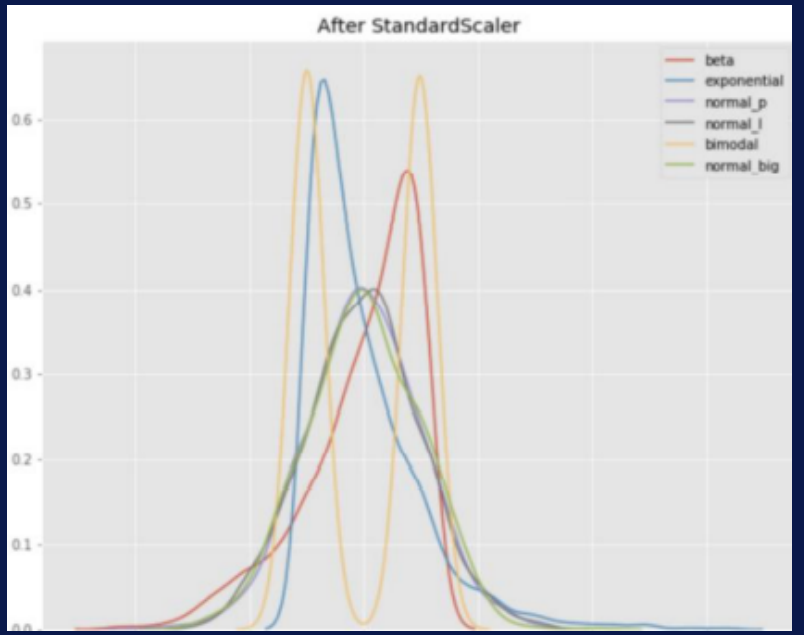
- Standard Scaler
- MinMax Scaler
- Robust Scaler
- Normalizer

01. Data Cleansing

Standard Scaler

개별 feature에서 평균값을 빼고, 분산을 나누어 평균은 0, 분산은 1로 변환 - Standardization
가우시안 정규 분포를 갖도록 변환하는 것은 몇몇 알고리즘에서 매우 중요
Ex) SVM, Linear Regression, Logistic Regression, Deep Learning
각 feature들 사이의 상대적 거리를 왜곡시킬 수 있다는 단점

$$Y = \frac{(X - X_{mean})}{\sigma_Y}$$



01. Data Cleansing

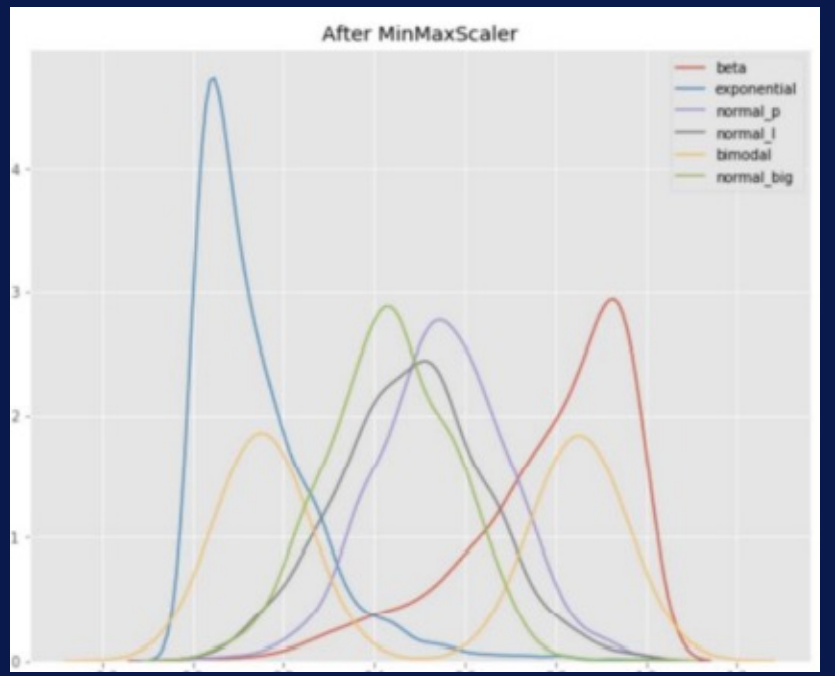
MinMax Scaler

개별 feature의 크기를 모두 똑같은 단위(0에서 1 사이)로 변경하는 것 – Normalization

본래 데이터의 정보를 변형시키지 않는다는 장점

이상치에 영향을 많이 받는다는 단점

$$Y = \frac{(X - X_{min})}{(X_{max} - X_{min})}$$

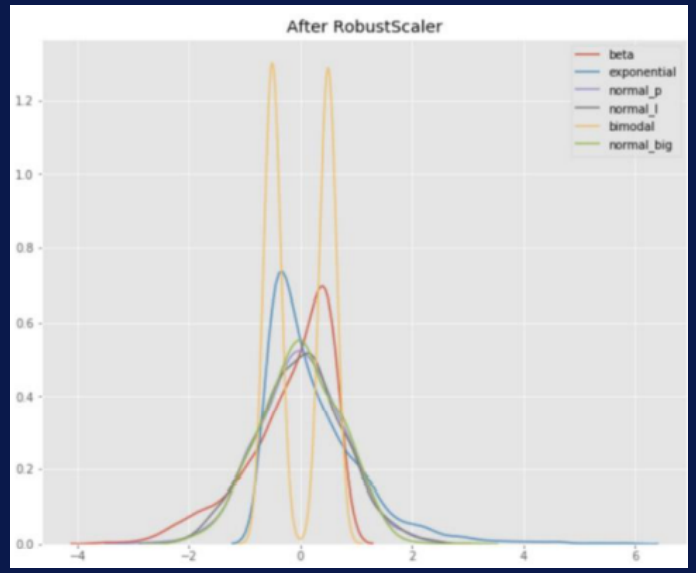


01. Data Cleansing

Robust Scaler

개별 feature 값에서 median을 빼고 IQR 범위로 나눈 것
각 feature의 범위는 Min-Max 보다는 큼
상대적으로 이상치의 효과를 줄이기에 적합

$$Y = \frac{(X - X_{median})}{(X_{IQR,75\%} - X_{IQR,25\%})}$$

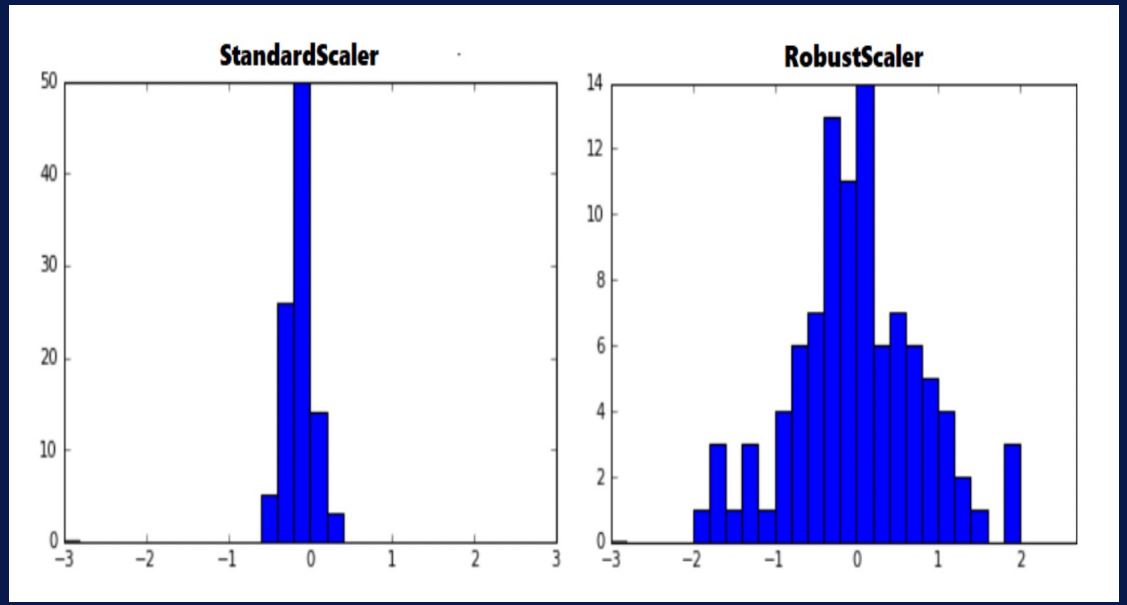


01. Data Cleansing

Robust Scaler

개별 feature 값에서 median을 빼고 IQR 범위로 나눈 것
각 feature의 범위는 Min-Max 보다는 큼
상대적으로 이상치의 효과를 줄이기에 적합

$$Y = \frac{(X - X_{median})}{(X_{IQR,75\%} - X_{IQR,25\%})}$$



01. Data Cleansing

Normalizer

선형대수에서의 정규화 개념이 차용되어 일반적 정규화와는 약간의 차이 존재
각 feature의 열(column)값이 아닌 행(row)값에 적용되는 scaler
대부분의 경우 위에 이전에 언급된 것들이 효율적임

$$Y = \frac{(X_i)}{\sqrt{(\sum_{j=1}^{N} X_j^2)}}$$

01. Data Cleansing

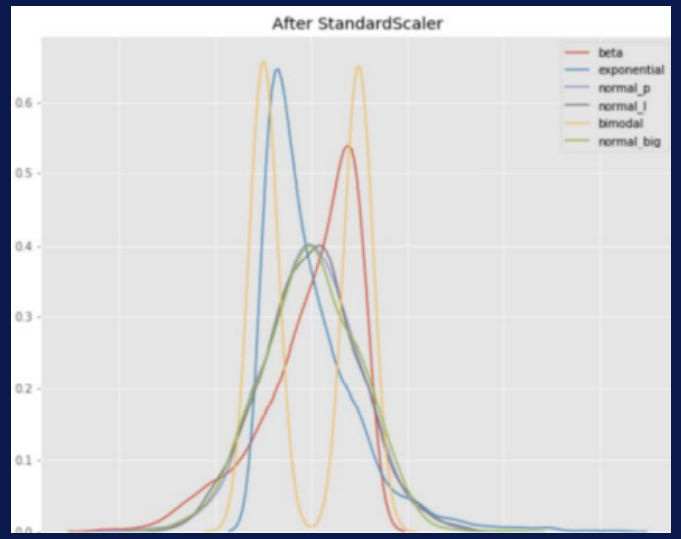
Scaler

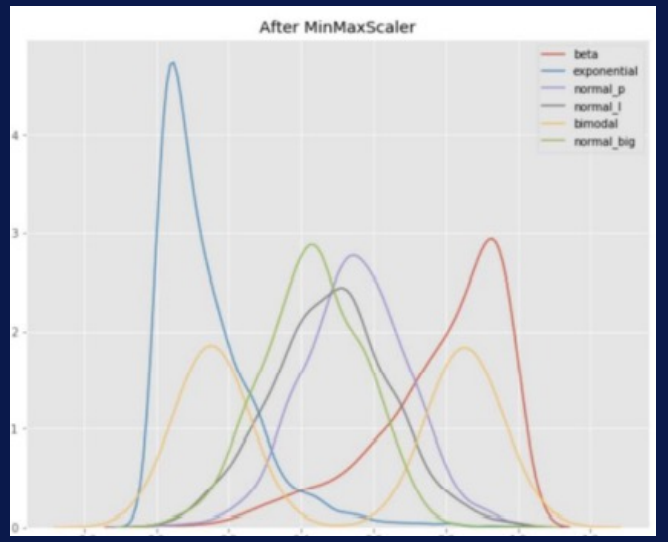
MinMax – 데이터의 왜곡이 없이 순수하게 분포를 비교하고자 할 때

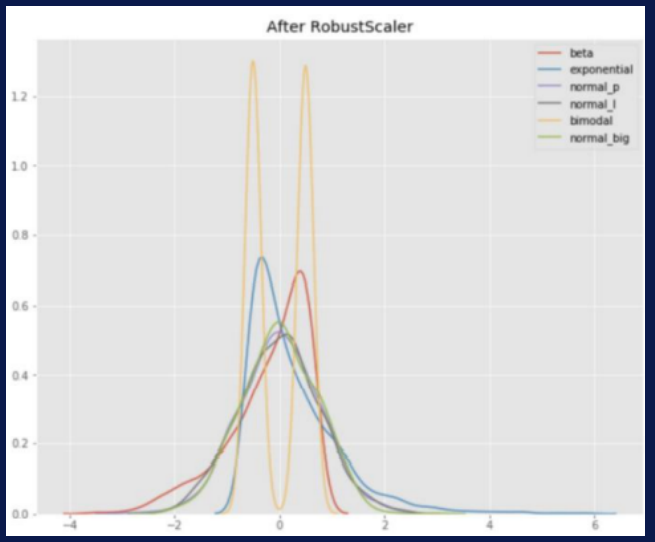
Robust – 이상치가 존재하고 그 영향을 줄이고 싶을 때

Standard – 모든 데이터의 분포를 정규분포로 보고싶을 때

(다양한 Scaler를 섞어서 시행하는 것도 고려해보면 좋음)







01. Data Cleansing

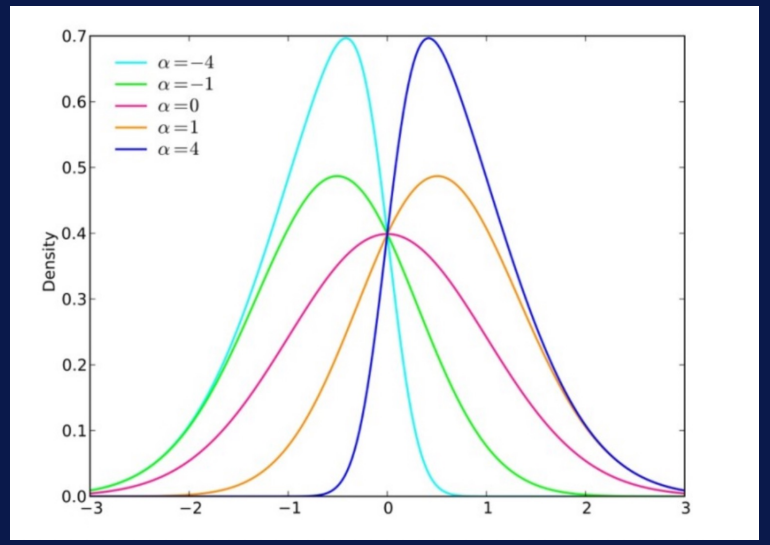
Transformation

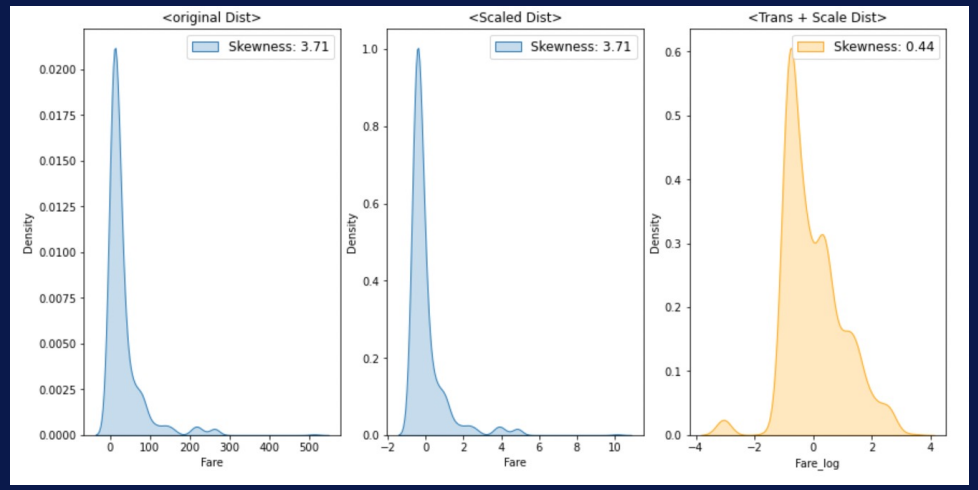
Skewness (왜도) :

분포의 정규분포에 비해서 얼마나 비대칭인지 나타내는 척도

왜도 값이 양의 값 – 정규 분포보다 오른쪽에 위치

왜도 값이 음의 값 – 정규 분포보다 왼쪽에 위치





Transformation 종류 :

np.log – 로그 변환

np.exp – 지수 변환

np.sqrt – 루트 변환

(Transformation과 scaling을 적절하게 섞어 사용하자!)

01. Data Cleansing

Encoding

Encoding = 코드화 = 암호화

Encoding in 컴퓨터 : 컴퓨터는 문자를 이해하지 못함

데이터를 약속된 규칙에 따라 컴퓨터가 이해할 수 있는 0과 1로 변환

Encoding in 분석 : 주로 범주형 변수를 수치형 변수로 변환

- Label Encoding
- One-hot Encoding
- Target Encoding

01. Data Cleansing

Encoding

- Label Encoding

  카테고리 Feature를 숫자로 변환 (ex. Apple → 1 , Chicken → 2 ..)

  모델은 숫자를 기반으로 연산 → 서열 변수가 아닌 경우 치명적

- One-hot Encoding

  Feature의 고유값에 해당하는 Column에만 1, 나머지는 0으로 표현

  차원의 저주에 걸릴 수 있음 (sparse하기 때문)

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

01. Data Cleansing

Encoding

- Target Encoding (Mean Encoding)

  Label Encoding과 유사하지만, Target값과 Encoding 값이 연관이 있다는 점에서 차이가 있음

  각 카테고리의 값을 학습 데이터의 Target값의 평균값으로 설정하는 방법

```
Sex
female    0.742038
male      0.188908
Name: Survived, dtype: float64
```

	Sex	Sex_mean
0	male	0.188908
1	female	0.742038
2	female	0.742038
3	female	0.742038
4	male	0.188908

01. Data Cleansing

Encoding

- Target Encoding (Mean Encoding)

장점

- 카테고리의 개수가 많을수록, Label Encoding은 Label 수가 계속 늘어남
  Target Encoding은 보다 적은 split이 생기고 학습이 더욱 빠르게 이루어짐
- Encoding된 Label값이 Target과 관련된 의미를 가짐 → less bias

단점

- Overfitting 가능성 높음
- 구현과 검증이 까다로움

02. Feature Extraction

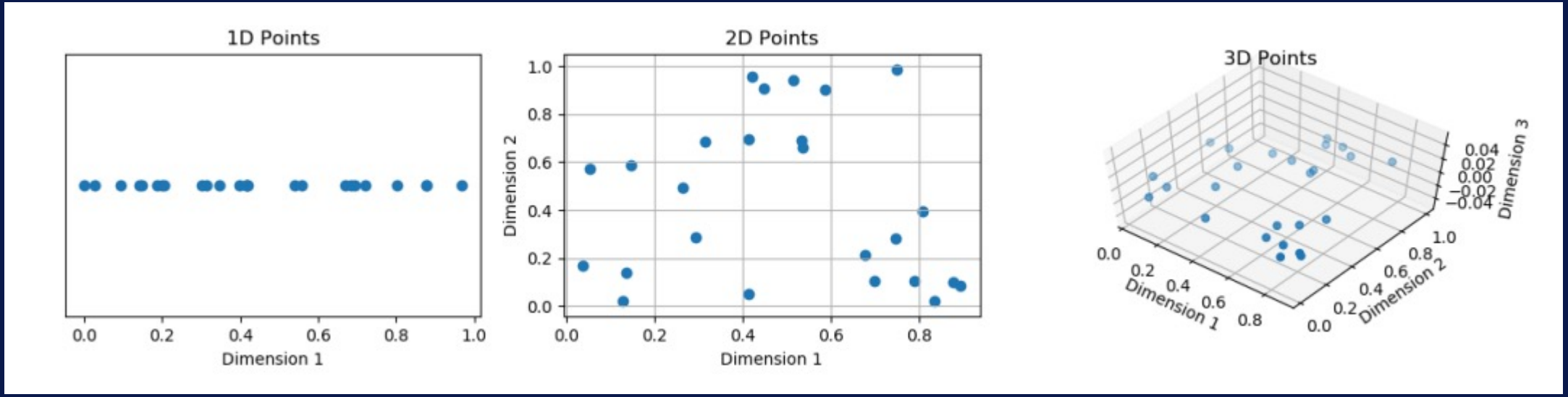
차원의 저주는?

데이터의 양은 동일한데, 데이터의 차원이 커지면 데이터의 밀도가 떨어짐

→ 원하는 정보를 찾는 데에 Computing Cost가 많이 소요

→ 데이터의 차원을 낮춰서 학습을 진행

(데이터의 차원을 낮추는 방법 : Feature Extraction, Feature Selection)



02. Feature Extraction

모든 Feature가 중요한 것은 아님
따라서 중요한 Feature를 선택 or 기존 Feature의 특징 추출 등 차원을 축소하여 사용

Feature Extraction

기존 Feature에 기반하여 새로운 Feature 생성
Ex. PCA

Feature Selection

기존 Feature들의 부분 집합으로
일부 중요한 Feature들만 선택적으로 사용

02. Feature Extraction

PCA란?

주성분 분석이라고도 말함

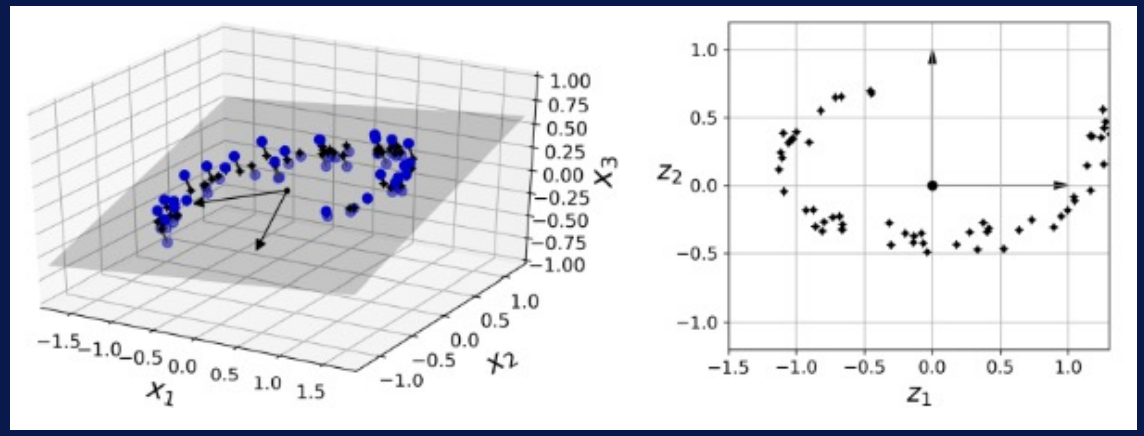
고차원의 데이터를 저차원의 데이터로 축소시키는 방법 중 하나

훈련 데이터에 가장 가까운 초평면(hyperplane)을 정의한 다음, 그 평면에 투영하는 기법

분산이 최대로 보존되는 축을 선택하는 것이 정보가 가장 적게 손실되므로 중요함

장점

1. 시각화
   3차원이 넘어간 시각화는 우리 눈으로 볼 수 없기 때문에
   PCA를 통해 차원을 축소하여 시각화 → 데이터 패턴을 쉽게 인지 가능
2. 노이즈 제거 – 쓸모없는 Feature를 제거함으로써 노이즈 제거 가능
3. 메모리 절약
4. 퍼포먼스 향상 – 불필요한 Feature를 제거해 모델 성능 향상에 기여



03. Feature Selection

Feature Selection이란?

기존 Feature에서 원하는 Feature만 선택하는 방법 (변경 X)

장점
- 사용자가 해석하기 쉽게 모델을 단순화
- 훈련 시간 축소
- 차원의 저주 방지
- 일반화

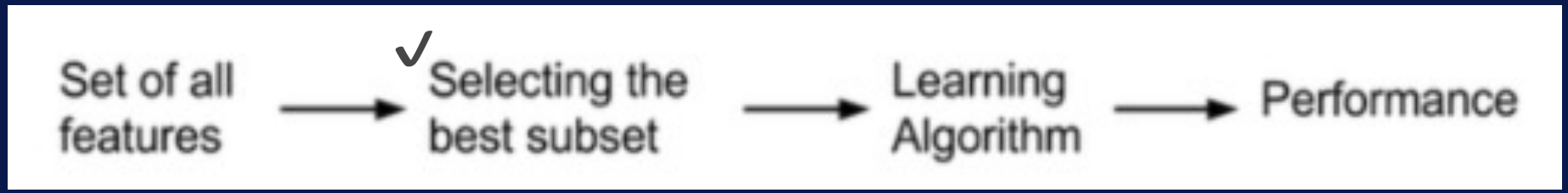
Filter Method, Wrapper Method, Embedded Method 가 있음

03. Feature Selection

Filter Method

통계적 측정 방법을 사용하여 Feature간의 상관관계를 알아낸 뒤
높은 상관계수(영향력)을 사지는 Feature를 사용하는 방법
다만, 상관관계가 높은 Feature가 반드시 모델에 적합한 Feature라고 할 수는 없음

- Information gain
- Chi-square test
- Correlation coefficient
- Variance threshold

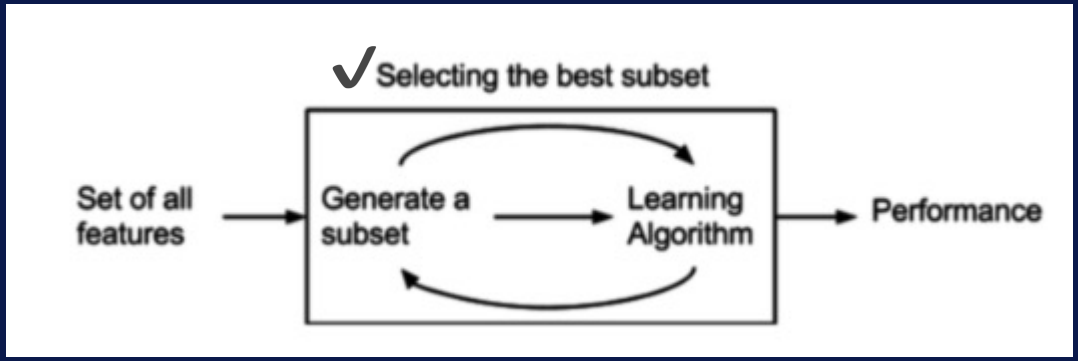


03. Feature Selection

Wrapper Method

예측 정확도 측면에서 가장 좋은 성능을 보이는 Feature subset을 뽑아내는 방법
테스트를 진행할 hold-out set 필요
여러 번 모델 학습을 진행하기 때문에 시간과 비용이 매우 높게 발생
하지만, Best Feature Subset을 찾을 수 있음

- Forward Selection (전진 선택)
- Backward Selection (후방 제거)
- Stepwise Selection (단계별 선택)

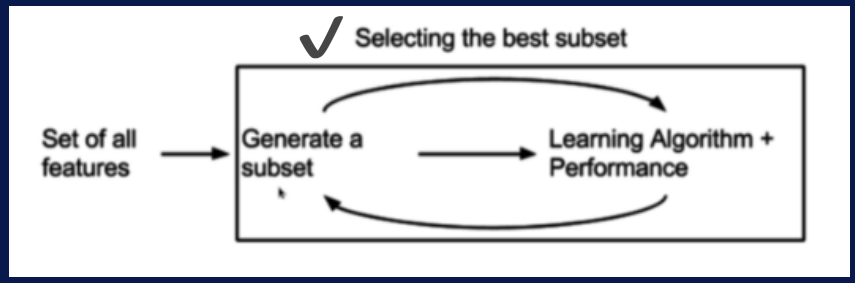


03. Feature Selection

Embedded Method

앞의 두 방식의 장점을 결합한 방법
각각의 Feature를 직접 학습하여 모델의 정확도에 기여하는 Feature 선택
계수가 0이 아닌 Feature가 선택되어, 더 낮은 복잡성으로 모델을 훈련하며 학습 절차를 최적화

- LASSO : L1-norm을 통해 제약을 주는 방법
- Ridge : L2-norm을 통해 제약을 주는 방법
- Elastic Net : 위 둘을 선형결합한 방법
- Select From Model : 트리 기반 알고리즘에서 Feature를 뽑아오는 방법
  (ex. RandomForest, LightGBM 등)



04. Hyperparameter Optimization

하이퍼 파라미터란?

모델링 과정에서 사용자가 직접 설정하는 값

Ex. Learning rate, SVM의 C or sigma, KNN의 K 등

최적의 값이 정해져 있는 것이 아님

→ 휴리스틱한 방법이나 경험에 의해 결정하는 경우가 많음

파라미터란?

모델 내부에서 결정되는 변수

그 값은 데이터로부터 결정

Ex. 평균, 표준편차 등

04. Hyperparameter Optimization

하이퍼 파라미터 튜닝

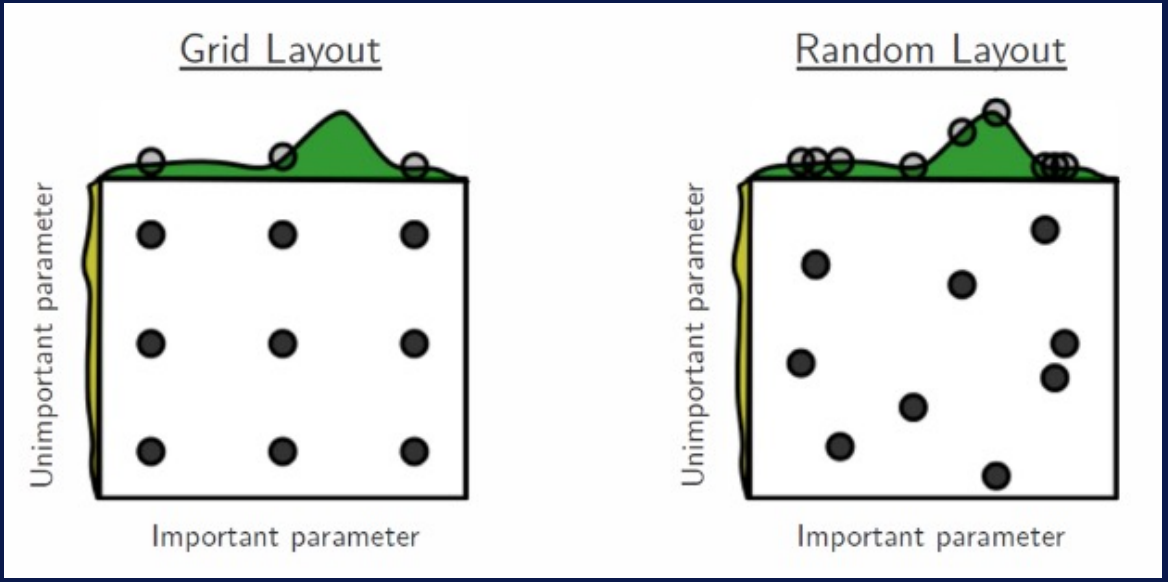
하이퍼 파라미터는 모델 학습 전에 값을 설정해야 함
하이퍼 파라미터 값 변경을 통해 모델의 성능을 끌어올릴 수 있음
데이터마다 최적의 하이퍼 파라미터 값이 다름
→ 해당 데이터 및 모델의 최적화된 하이퍼 파라미터를 찾는 과정이 필요

대표적인 하이퍼 파라미터 튜닝 방법 - Grid Search
Random Search
Bayesian Optimization

04. Hyperparameter Optimization

Grid Search

탐색의 대상이 되는 특정 구간 내의 후보 하이퍼 파라미터 값들을 일정한 간격을 두고 선정
후보 하이퍼 파라미터들의 모든 조합을 시행
이들 각각에 대하여 성능 결과를 측정한 후 가장 높은 성능을 발휘한 값을 최적값으로 선정



04. Hyperparameter Optimization

Grid Search

탐색의 대상이 되는 특정 구간 내의 후보 하이퍼 파라미터 값들을 일정한 간격을 두고 선정
후보 하이퍼 파라미터들의 모든 조합을 시행
이들 각각에 대하여 성능 결과를 측정한 후 가장 높은 성능을 발휘한 값을 최적값으로 선정

estimator	Classifier, Regressor, Pipeline 등이 사용
param_grid	Key : 리스트 형태의 딕셔너리로 주어짐 Estimator의 튜닝을 위해 파라미터명과 사용될 여러 파라미터 값 지정
scoring	예측 성능을 측정할 평가 방법 지정 정확도를 측정하는 accuracy등 성능 평가 지표 함수 지정
cv	교차 검증 폴드 수
refit	Default는 True GridSearchCV를 사용하여 찾은 최적을 하이퍼 파라미터 값을 Estimatior 객체에 적용하여 다시 학습할지를 묻는 부분
n_jobs	코드 실행할 때 몇 개의 코어를 쓸 것인지 지정 (-1 : 컴퓨터 코어 전부 사용)
verbose	실행 결과 출력에 대한 설정 (0 : 출력X, 1: 마지막 결과만 출력, 2 : 매 시도마다 출력)

04. Hyperparameter Optimization

Grid Search

장점

- 직관적이며 하이퍼 파라미터 탐색 공간이 좁은 경우 효과적임

단점

- 지정한 하이퍼 파라미터 후보군의 개수만큼 비례하여 탐색 시간이 늘어남

  Ex. 5개의 하이퍼 파라미터에 대해 각각 5개의 후보 값을 설정

  5^5 = 3125번의 시도

→ 처음에는 간격을 넓게 설정하여 적은 수의 그리드로 시작

→ 이후 결과에 따라 세분화하여 검색

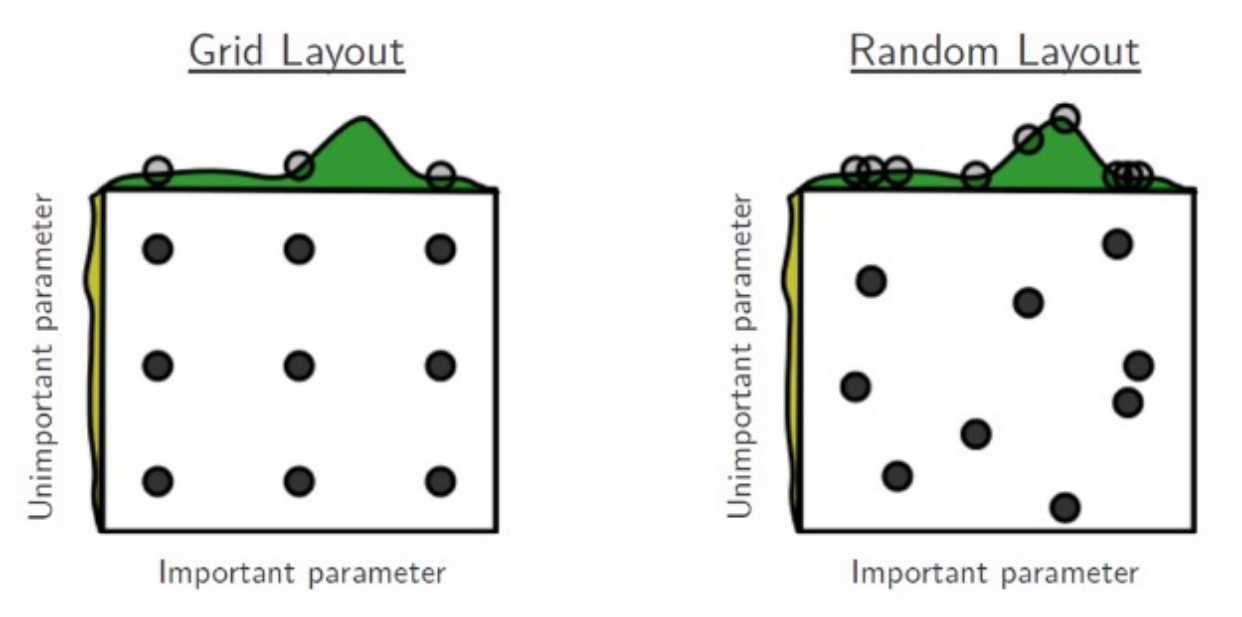
04. Hyperparameter Optimization

Random Search

하이퍼 파라미터로 적용해볼 값들을 미리 정하여 범위 내 무작위 값 추출

Grid Search보다 훨씬 다양한 조합들을 시험

n_iter 값만큼 반복 무작위 추출



04. Hyperparameter Optimization

Random Search

하이퍼 파라미터로 적용해볼 값들을 미리 정하여 범위 내 무작위 값 추출
Grid Search보다 훨씬 다양한 조합들을 시험
n_iter 값만큼 반복 무작위 추출

estimator	Classifier, Regressor, Pipeline 등이 사용
param_distributions	딕셔너리 형태로 변수 명과 해당 변수의 범위를 랜덤 값으로 선언하는 매개변수
n_iter	몇 번 반복하여 수행할 것인지
cv	교차 검증 폴드 수
scoring	예측 성능을 측정할 평가 방법 지정 정확도를 측정하는 accuracy등 성능 평가 지표 함수 지정

04. Hyperparameter Optimization

Random Search

장점

- 랜덤하게 숫자를 넣을 수도 있고, 정해진 간격 사이에 위치한 값들에 대해서도 탐색 가능
- Grid Search보다 최적 하이퍼 파라미터를 먼저 찾을 수 있음
  (시간 대비 성능이 좋음)
- 하이퍼 파라미터 탐색 공간이 클 경우 사용하면 좋음

→ Random Search 먼저 진행하고, Grid Search 하는 것을 추천

04. Hyperparameter Optimization

Bayesian Optimization

베이지안 최적화는 미지의 목적함수를 최대화 or 최소화하는 최적해를 찾는 기법

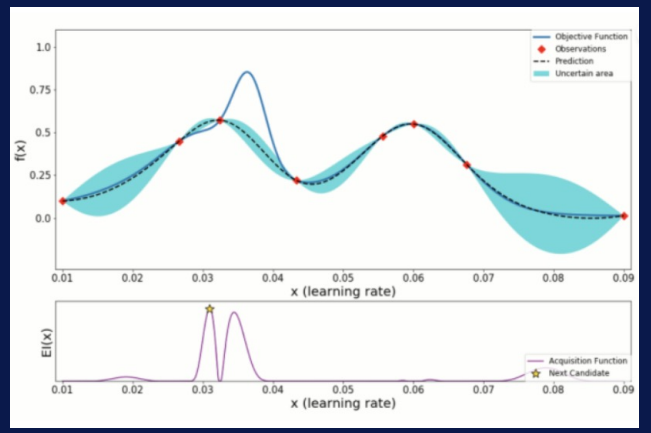
Surrogate model, Acquisition function으로 구성

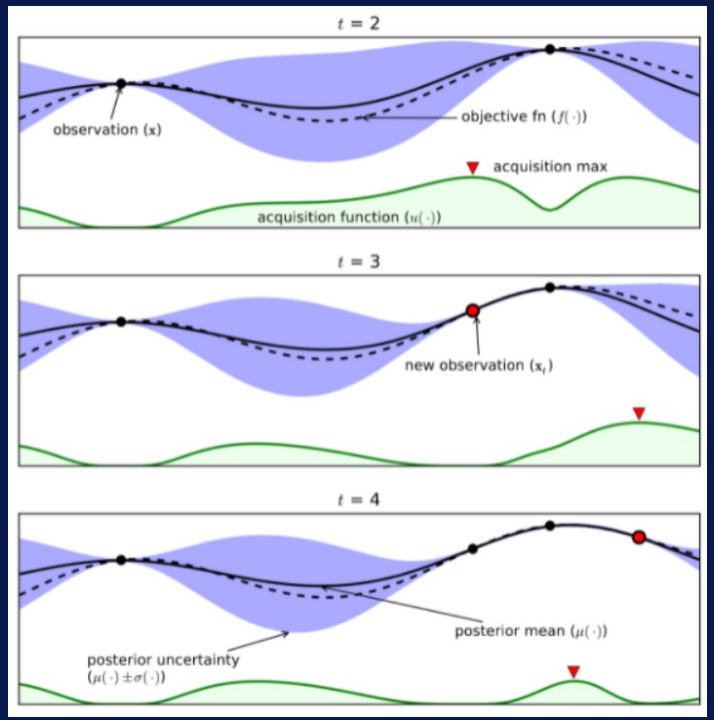
- Surrogate model : 현재까지 조사된 입력값-함수값 쌍들에 대해 목적 함수에 대한 확률적 추정을 수행하는 모델
- Acquisition function : Surrogate model을 활용해 다음 입력값 후보를 추천해주는 함수

앞에서 배운 Grid Search, Random Search와는 다르게 사전 분포를 활용

입력값-함수값의 조합을 기반으로 Surrogate model을 만들고 순차적으로 업데이트

예측값과 실제 함수값이 거의 유사해지는 순간이 오면 그 지점 중 최적값을 argmin(x)로 선택





D&A

ML Session 5차시 Data Preprocessing

Thank You.

2022 / 10 / 04

D&A 부학회장 김정하