Machine Learning Summary

1. 확률 통계

1) Sample Space and Events

S (Sample Space)	발생할 수 있는 모든 경우의 수의 집합
Event (사건)	Sample Space의 부분 집합
Event Space	Sample Space의 요소로 구성될 수 있는 모든 조합의 경우의 수 (2^S개로 구성됨) Event Space 내에 공집합과 전체 집합도 포함됨 / 공집합과 전체 집합도 사건 일부

2) 확률 (Probability)

- **불확실한 사건의 발생을 숫자로서 표현하는 것** (※ 불확실성이 있다는 것은 특정 상황을 확률로서 표현할 수 있음)

- **확률을 구하는 방법**
  - ① **Classical Approach의 확률** : 발생 가능한 사건이 각각 동일한 발생 가능성을 가진다고 가정함
    (※ 완벽한 동전 또는 주사위는 존재하지 않음 / 불확실성에 확실성을 부여하려하기에 모순적임)
  - ② **Relative Frequency Approach (Frequentist)의 확률** : 무수한 실험 후 확률 결정
    (※ 무한적인 실험을 요구하기에 비현실적이며, 무수한 실험의 결과가 수렴하지 않으면 문제가 발생함)
  - ③ **Subjective Approach (Bayesian)의 확률** : 상황 발생에 대한 믿음의 정도를 가지고 결정
    (※ 반복적인 실험에 대한 확률을 알 수 없을 경우 사용 / 주관적인 견해와 믿음이 반영됨)

- **확률로 인정받기 위한 조건 (Probability Measure)**
  - ① Event Space 내에서 각각의 Event에 대한 확률값은 0이상이어야함
  - ② 모든 경우의 수의 확률 합은 1이어야함
  - ③ 교집합이 없는 두 상황에 대한 확률값은 각각의 상황에 대한 확률값을 더한 것과 같음

- Classical Approach, Relative Frequency Approach, Subjective Approach 중 1개를 선택하여 특정 상황에 대한 확률값을 구하고, 확률 인정 조건을 통해 확률이라는 것을 증명/인정함

3) 조건부 확률 (Conditional Probability)

- **$P(F|H)$: H라는 사건이 발생할 때, F 사건도 같이 발생할 확률** = $P(F \cap H) / P(H)$

- **$P(F|H) = P_H(F)$** → H에 대한 Sample Space로 관점을 이동하여, H Sample Space에서 F 사건 발생을 표현함

- **만약 2개의 사건이 독립이라면 조건에 상관없이 각자의 상황은 발생해야함.**
  그러므로 F와 H가 독립이라면 서로에게 영향을 주지 않기에 서로에 대한 조건부 확률은 각 상황이 발생하는 확률과 같음 ($P(F|H) = P(F)$, $P(H|F) = P(H)$). 이를 통해서 F와 H가 독립일 때, $P(F \cap H) = P(F)P(H)$라 할 수 있음.

- **Rule of Total Probability ($P(E) = \sum P(B_i \cap E) = \sum P(B_i)P(E|B_i)$)**
  E라는 사건이 내부적으로 독립된 $B_1 \sim B_n$ 사건들로 구성 되어있을 시, E 사건이 발생할 수 있는 확률은 E사건과 B_i 사건이 같이 발생할 수 있는 모든 경우의 확률의 합과 같음. **조건부 확률로 바꿔서 표현하게 되면 B_i라는 각각의 사건에 대한 상태만 알아도 전체 상태인 E를 유추할 수 있음.**

- **Bayes' Rule**
  : **새로운 정보가 제공되면 불확실성에 대해 Update될 수 있음.**
  : $P(B_i|E)$에서 사건에 대한 새로운 정보는 E에 추가되어 확률을 Update할 수 있음.
  : Bayes' Rule은 확률의 Updating Rule을 알려줄 수 있음.

4) 확률 변수 (Random Variables) / 확률 분포 (Probability Distribution)

- **확률 변수 X**
  : Sample Space 내에 속한 각각의 사건을 기호가 아닌 실수 형태의 Label로 표현하는 방법
  : 각 사건의 발생에 대해 대응되는 실수를 배치하며, 사건의 불확실성을 실수 영역에서 다룰 수 있게됨.
  → **이산확률변수 (Discrete Random Variable)** : 사건에 대한 수학적 표현 • 실수 형태 Label이 Discrete한 경우
  → **연속확률변수 (Continuous Random Variable)** : 사건에 대한 수학적 표현 • 실수 형태 Label이 Continuous한 경우

- **확률 분포 P(X)**
  : Sample Space 내에 속한 각각의 사건에 대응되는 실수인 확률변수에 따라 매칭된 확률값의 집합
  → 확률 변수 X에 대한 확률 분포 P(X)를 통해 **"기존 사건 대 실수 확률값"**의 표현을 **"사건의 실수 표현 대 실수 확률값" 관계로 재정의**하면서 **사건의 불확실성을 숫자의 집합으로 만들어서 수학적 분포 • 수학적 수식으로 정의할 수 있게됨.**
  → **이산확률분포 (Discrete Probability Distribution) :** 이산확률변수에 따른 확률값의 Discrete한 집합
  → **연속확률분포 (Continuous Probability Distribution) :** 연속확률변수에 따른 확률값의 Continuous한 집합

5) Mean, Variance, Covariance, Correlation

- **Mean (기댓값, 평균)**은 확률 분포 내 사건 표현값과 확률값의 곱의 합이며, **확률 분포의 무게 중심을 보여주는 척도임.** 특정 사건이 발생할 때 해당 사건에 대해 예상되는 확률을 지칭함.

- **Variance (분산)**은 확률 분포 내 각 사건이 Mean으로부터 오차에 대한 평균을 구한 값이며, **확률 분포 내에서 Mean을 중심으로 전반적으로 확률이 얼마나 넓게 퍼져있는지를 보여주는 척도임.** Variance가 높을수록 사건이 Mean을 중심으로 넓게 펴져 다양한 사건이 발생할 수 있음. Variance가 낮을수록 사건이 Mean을 중심으로 모이게 되어 Mean에 유사한 사건들이 발생할 수 있음.

- **Covariance (공분산)은 2개 변수 X, Y 간의 선형성(Linearity)을 파악하기 위한 척도임. Covariance는 2개 변수 사이에 선형성 유무와 선형성 방향을 알려줄 수 있음. 그러나 Covariance는 단위를 가지는 척도이기에 선형성의 강도를 표현할 수 없음.**

- **Correlation (상관관계)은 Covariance와 같이 2개 변수 X, Y간의 선형성을 나타내기 위한 척도임. Covariance와 달리 단위가 없는 무단위 척도이기에 2개 변수 사이의 선형성 유무, 선형성 방향, 선형성 강도 모두 표현할 수 있음.**
  (※ Covariance, Correlation은 2개 변수 사이의 선형성을 알려줄 뿐 '인과 관계'를 알려주지 않음.)

6) Statistics

- **Random Sample**
  : Random Sample은 모집단에서 구성원이 Sampling에 의해 선택될 확률이 동등하다는 원칙하에 모집단으로부터 선택될 Sample (표본)을 의미함.
  : 모집단 내에서 선택될 확률만 같은 상황이기에 Random Sample은 아직 모집단에서 실제로 Sampling • 실험 행위가 수행되기 전의 표본임. 모집단에서 어떤 요소를 뽑은 것이 아니라 앞으로 뽑을 예정인 것임.
  : Random Sample은 실제 Sampling • 실험을 진행하기 전까지는 어떤 값이 나올지 모르기에 Sample 간에는 독립적이며 (Indepdent), 모집단 내에서 어떤 것이든 동등한 확률로 선택될 수 있기에 (Identically Distributed) 모집단과 동일한 확률변수 특성을 가지게 됨.

- **Statistic (통계량)**
  : Statistic은 Random Sample로 구성된 값을 의미함. Sample을 뽑아야 값을 알 수 있음.
  : Statistic은 모집단에서 앞으로 선정될 요소인 Random Sample로 구성된 값이기 때문에 Sampling • 실험을 진행하면서 Statistic (통계량)에 대한 값을 점차적으로 알아갈 수 있음.
  : 그러므로 Statistic은 Sampling • 실험하기 전까지는 불확실성을 지니기 때문에 '확률변수'라 정의할 수 있으며, Sampling • 실험을 수행에 따라 사건이 발생함에 따라 실수 확률 분포로 표현될 수 있음.

- **Estimator**
  : **Estimator는 Sample을 가지고 모집단의 특성을 추정하기 위해 사용하는 Statistic • 수학적 도구**
  : **Estimator는 Sample의 평균인 Sample Mean과 분산인 Sample Variance를 사용함.** 각각은 Sampling • 실험을 통해 **실제로 Sample을 모으기 전까지는 Sample Mean과 Sample Variance를 구할 수 없음.**
  : Sampling • 실험을 진행할수록 Sample Mean과 Sample Variance 값이 업데이트됨. **Sample Mean과 Sample Variance는 고정된 값이 아닌 Sampling • 실험 횟수에 따라 변동되기에 불확실성을 지니게되며, 확률로 취급되어 확률 분포로서 표현될 수 있음.**
  : 만약 모집단에서 **Sample을 계속 무한히 뽑을수록 Sample이 모집단과 유사해지면서 Sample Mean과 Sample Variance가 모집단의 확률 분포와 유사해짐. 그러므로 Sample Mean과 Variance를 통해 모집단의 Mean과 Variance를 추정할 수 있음.**

- **최적의 Estimator 조건**
  ① **Unbiasedness :** 모집단 분포의 중심 Mean이 Estimator (Sample Mean)와 최대한 일치해야 모집단 추정이 정확해짐
  ② **Efficiency :** 모집단을 추정 정확도의 분포가 짧아야함
  ③ **Consistency :** Sample을 점점 많이 뽑을수록 모집단에 유사해짐
  ④ **Sufficiency :** Estimator가 모집단에 대해서 정보를 충분히 제공해야함

- **Bias-Variance Decomposition을 통해 모집단 vs Sample간의 Error가 Bias와 Variance의 조합으로 구성되는 것**을 볼 수 있으며, **Bias와 Variance의 조합에 따라 Error가 모집단과 Sample 사이의 Error가 최소화**될 수 있음.
  → **Bias가 생겨서 모집단의 Mean과 Sample의 Mean이 서로 조금 어긋나더라도 Variance가 좁으면 모집단과 가까운 위치에 있기에 모집단 vs Sample간의 Error가 적을 수 있음.**

- **Machine Learning의 학습 ≒ 최적의 Estimator를 찾아내는 문제**
  : 모집단에서 Sample을 추출하면서 Estimator를 통해 모집단을 추정함
  : **Machine Learning에서 내가 원하는 Dataset에 대해 Prediction을 잘해주기 위해 Sample 데이터 (Train Dataset)을 제공해서 학습시켜서, 그에 따라 모집단에도 동일하게 정확하게 Prediction이 작동되게함.**
  : Machine Learning에서 학습을 통해 최적 성능의 Network를 설계하는 문제는 모집단에 대해 Unbiased하고 Efficient한 최상의 Estimator를 구하는 것과 같음.

7) Derivative • Differentiation / Gradient

- **Differentiation (미분)**
  : 특정 변수 X의 변화에 대한 결과값 Y의 변화율을 관찰함.
  : 그래프 상 특정 지점의 변화율인 기울기를 의미함.
- **Partial Differentiation (편미분)**
  : 다변수로 구성된 공식이나 관계 W = f(X, Y, Z)에서 특정 변수 X의 변화에 대해서만 결과값 W의 변화율을 관찰함.
  : 미분 대상인 X를 제외한 나머지 변수는 상수로 취급하여 X에 대해서만 W의 미분을 계산함.
- **Gradient**
  : W = f(X, Y, Z)와 같은 다변수 함수의 기울기를 정의하기 위해 각 독립변수 X, Y, Z에 대한 W의 편미분 모음

8) Vector / Dot Product / Norm

- **Vector** : 크기와 방향을 가지는 물리량
- **Dot Product** : Vector와 Vector가 곱해서 실수가 나오게 하는 곱셈 연산
- **Norm**
  ① **L^1 Norm ($||a||_1 = \sum |a_i^2|$) : Vector의 모든 요소의 절대값의 합**
  ② **L^2 Norm ($||a||_2 = \sqrt{\sum a_i^2}$) : Vector의 각 성분의 제곱을 더해서 루트를 적용한 값**

2. Machine Learning 접근 방법

1) Machine Learning 정의

- **Field of study that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959)**
  - 명시적인 프로그래밍 없이 데이터만을 제공하여 경험 누적을 통한 성능 개선
  - Data 를 제공하여 문제를 해결하거나 기능을 수행하는 모델을 만들어내는 기법
  - Input : Model Input & Output Dataset
  - Output : Model (Algorithm)

- **Classical/Conventional Approach (Non-ML / Non-NN)**
  - 목표하는 기능 및 문제에 대해 매번 모델/알고리즘을 재설계해야함.
  - 목표하는 기능 및 문제에 대해 모델/알고리즘의 Parameter 를 Fine-Tuning 해야함.

- **Machine Learning / Neural Network Approach**
  - 목표 기능 및 문제를 표현하는 Dataset 을 이용한 학습을 통해 모델/알고리즘을 도출함
  - Dataset 에 추가 데이터를 업데이트하고, 새로운 추가 데이터를 학습하여 모델/알고리즘의 성능을 개선함.
  - 모델/알고리즘의 Parameter 는 학습을 통해 Tuning 되기에 학습에 대한 Hyper Parameter (반복 학습 횟수 Epoch, 1 회 학습 데이터 크기 Batch Size, 학습 속도 Learning Rate 등)을 조정하여 모델/알고리즘의 학습 과정을 최적화함
  - 모델/알고리즘의 성능 개선을 위해서는 좋은 Feature 로 구성된 많은 데이터가 필요하며, 목표하는 기능 및 문제에 따라 매번 다른 학습이 요구됨.

2) Machine Learning 기법 분류

- **Supervised Learning**
  - Input Dataset 과 Output Label (Input 에 대한 정답)을 모두 제공하여 모델을 학습하는 방법
  - Input 데이터에 따라 Error 가 최소화될 수 있도록 (Loss Function 값이 최소화도록) 적합한 정답을 도출하도록 모델/알고리즘을 학습시킴
  - 정해진 Label 을 추정하도록 학습되기에 Classification 문제를 해결하는 데에 특화됨
  - Linear Regression, Logistic Regression, KNN (K-Nearest Neighbors), SVM (Support Vector Machine), Decision Tree, Random Forest, Neural Network
- **Unsupervised Learning**
  - Output Label 없이 Input Dataset 만 사용하여 모델을 학습하는 방법
  - Clustering, Dimensionality Reduction, Association Rule Learning
- **Semi-Supervised Learning**
  - Supervised & Unsupervised Learning 의 조합으로 이루어짐
- **Reinforcement Learning**
  - Agent 가 상태(State)의 보상 (Reward) 최대화하는 방향으로 Neural Network 의 Parameter 를 최적화함

3) Dataset 구조

	Column 1 (Feature 1)	Column 2 (Feature 2)	· · ·	Column M (Feature M)
Data 1				
Data 2				
⋮				
Data N				

- Dataset 은 NxM 형태의 Matrix 로 구성되어있음. 데이터셋은 N 의 데이터로 구성되어있으며, 각 데이터는 M 개의 Feature 로 구성되어 표현됨
- Dataset 의 Row Vector 는 데이터셋의 데이터 1 개를 지칭함 (데이터 N 개)
- Dataset 의 Column Vector 는 각 데이터를 표현하는 Feature 를 상징함 (Feature M 개)
- Dataset 의 데이터는 ML 및 Neural Network 모델에 1xM 형태의 벡터로 입력됨

4) Machine Learning Implementation 순서

(1) Dataset 준비

- 목표하는 기능과 문제와 관련된 데이터셋을 확보해야함
- 데이터셋은 ML/DL 라이브러리와 연동되기 위해 NxM Matrix 형태로 구성되어야함

(2) Dataset Preporcessing

- 데이터셋의 각 Feature 는 다른 단위의 값으로 구성됨
- 각 Feature 별로 다른 단위를 사용하게 되면서 데이터의 분포가 달라지고 학습에 대한 영향력이 달라지는 Scale 문제로 인해 학습 과정 중 Overfitting 과 같은 Fitting 문제가 발생함
- Standardization, Normalization 등 데이터셋의 단위를 통일시키는 Preprocessing 을 수행하여 모델/알고리즘의 Fitting 을 개선함
- 만약 현재 데이터셋에서 사용하는 Feature 가 너무 많아 Classification 을 위한 학습이 복잡할 경우 데이터셋을 더 적은 Feature 로 재구성하는 Dimensionality Reduction 이 요구될 수 있음. 이 때, PCA (Principle Component Analysis)를 통해 설명력이 높은 소수의 새로운 Feature 로 데이터셋을 재구성하여 Classification 학습에 좀 더 쉬운 방향으로 유도할 수 있음.

(3) Dataset Analysis 를 통한 ML 기법 선정

Machine Learning 과 Neural Network 는 데이터셋에 매우 의존적이기 때문에 데이터셋의 Feature 성격에 따라 적합한 모델을 선정해야함.

➢ **Multi-Colinearity 존재할 경우**

- 우선적으로 데이터셋의 Feature 간 선형성 (Colinearity)를 파악해야함.
- 다수의 Feature 로 구성된 데이터셋에서 Feature 간에 서로 독립적인 것이 이상적이나 Feature 간 서로 영향을 주는 Multi-Colinearity 가 존재할 수 있음.
- 이를 파악하기 위해 Feature M 개에 대한 MxM Correlation Matrix 를 구성해서 각 Feature 가 서로에게 Correlation (선형성 존재여부)을 가지는지 파악함.

Fig 1: Correlation Matrix (Breast Cancer Dataset)

- Multi-Colinearity 가 존재하는 Feature 사이에는 Linear Classifier 는 적용할 수 없음.
- Multi-Colinearity 가 존재하는 Feature 사이에 Classification 을 위해서는 보다 복잡한 구조의 모델/알고리즘이 요구됨.

  (ex : Neural Network, SVM, Random Forest, Decision Tree)

➢ **데이터셋의 Non-Linearity 존재할 경우**

- 데이터셋 Feature 의 물리적 특성이 달라 Non-Linearity 가 존재할 수 있음.

  (ex : 2D 이미지 vs 3D Translation / 2D 이미지 vs 3D Rotation)
- Non-Linear 데이터셋에 대해서는 Linear Classifier 를 적용할 수 없으며, 더 많은 Parameter 를 학습하여 Non-Linear 한 추정을 할 수 있는 복잡한 모델/알고리즘이 요구됨. (ex : Neural Network)

(4) Training/Validation/Test Dataset 분리

데이터셋은 학습 및 성능 평가를 위해 전체에서 일정 비율로 Training Set, Validation Set, Test Set 으로 각각 분리하며, 서로 중첩이 없어야하며 Random 하게 선정됨.

- **Training Set**
  - ML 모델 학습을 위해 사용하는 데이터셋
  - **<u>데이터셋에 Fitting 하기 위해 학습을 하는 것이 아니라 Prediction 을 하기 위해 모델을 학습하는 것임</u>**
- **Validation Set**
  - ML 모델 학습에 필요한 Hyper Parameter 를 찾기 위해 사용하는 데이터셋
  - 학습이 종료된 후 Test Set 이 주어지기 전까지 학습 과정 중 모의 평가를 하기 위해 사용
  - 모의 평가를 통해 Unseen Data 에 대한 정확도를 확인하여 Test Set 에 대한 정확도를 가늠할 수 있음.
  - 모의 평가를 통해 Training Set, Validation Set 2 개에 대해 모두 정확도가 높게 나오게 만듬으로서 Training Set 에만 최적화되는 Overfitting 을 방지함.
- **Test Set**
  - 학습된 모델을 평가하기 위한 데이터셋

(5) Training 수행 + Optimization

- **Cross Validation**
  - **현재 사용하는 데이터셋에 대해 ML 모델의 일반화 성능 및 가능성을 판단하기 위한 데이터셋 활용 방법**
  - **데이터셋을 단순히 Training / Test 2 가지로만 나누면 운에 따라서 Training 과 Test 의 난이도가 서로 달라지게됨. 이를 방지함으로서 모델의 일반화 성능 및 가능성을 판단할 수 있음.**
    - **K-Fold Cross Validation (5-Fold / 10-Fold Validation)**

    - 전체 데이터셋을 K 개 Fold 로 분리함
    - K 개의 Fold 중 1 개를 Test Set 으로 설정하고 나머지를 Training Set 으로 설정함. 이 과정을 총 K 번 반복하면서, 매번마다 다른 Fold 를 Test Set 으로 설정함. 각 Fold 는 한 번씩 Test Set 으로 설정됨.
    - 여러 경우의 Train / Test 시나리오를 제공함으로서 보다 더 Generalized 된 시나리오를 네트워크에 시도해볼 수 있음.
    - K 개의 모델에 대한 각각의 Accuracy 가 서로 너무 다르면 데이터셋에 대해서 Generalization 이 잘 안되고, 데이터셋에 매우 민감하다는 것을 알 수 있음. 이를 통해 일반화 성능 및 가능성을 판단함.

- K-Fold Cross Validation 은 K 개의 Model 중 최상의 Model 을 선정하는 것이 아니라 현재 사용하는 ML 모델이 현재 데이터셋에 대해 일반화가 가능한 적합한 모델인지 판단하는 기법임.
- K 개만큼 시나리오를 배로 늘리기 때문에 연산 소요시간이 K 배로 증가함.

- **Stratified K-Fold Cross Validation**

  - **Stratify**
    : 데이터셋을 나눌 때 각 Label 별 데이터가 같은 비율로 들어가게 분리함.
  - Stratified K-Fold 는 Fold 를 생성할 때 각 Label 별 데이터가 같은 비율로 들어갈 수 있도록 데이터셋을 분리하는 기법임.

- **Shuffling**
  - 데이터셋을 랜덤하게 섞어서 분리함 **(※ 데이터 Ordering 제거)**
  - Shuffling 시 Random State 를 부여해서 Split 의 분포가 일정하게 유지되게 하여 Validation Score 가 매번 발산하기 않게함.
    (※ Reproducibility 보장)
  - **Random State = 0**
    : Shuffle 과 Folding 을 섞으면 예상치 못한 영향을 데이터셋에 주게됨.

- **Leave-One-Out**
  - 데이터셋에서 매번 1 개만 Test 로 선정해서 학습시키는 전략
  - 데이터가 적을 경우 적용하는 학습 전략 (큰 데이터셋에서는 매우 느림)

- **Shuffle-Split Cross Validation**

  - Sub Sampling 을 통한 유연한 데이터셋 선정 전략
  - 데이터셋 분리를 위한 모든 요소를 제어함

■ **GroupKFold**

■ 종류에 따른 Accuracy 를 높이는 학습을 위해 종류별로 그룹으로 묶어서 Train / Test 를 분리함. **(※ Group / Label 을 혼용하지 않음)**

■ 데이터셋에서 전체 종류가 아닌 일부 종류만 Training Set 에 반영되게 하여 일부러 모든 종류의 데이터를 학습시키지 않음.

■ 새로운 종류의 데이터에 대해 정확도를 높이기 위한 학습 전략 **(※ 새로운 종류 인식이 목적)**

➢ **Grid Search**

- 주어진 모델에 대한 최상의 중요 Parameter 찾아서 최상의 모델 선정하기
- 다양한 Parameter 를 모두 시도해서 최상의 경우를 찾아감
- **Training / Validation / Test**

  : Training – Model Fitting

  : Validation – Parameter Selection (Grid Search)

  : Test – Evaluation

  : ML 모델을 평가하기 위해서는 모델 생성(Fitting & Parameter Selection)에 직간접적으로 관여하지 않은 새로운 데이터가 필요함

  : Parameter Selection 에 Test Set 을 사용하면 모델이 간접적으로 Test Set 을 알게됨

  : 데이터셋은 Training, Validation, Test 3 가지로 나누며, Test 는 Model Fitting 또는 Parameter Selection 에 일체 관여하지 않는 독립적인 데이터셋이어야함.
- Grid Search 를 위해 데이터셋을 Training, Validation, Test 로 나누더라도 각각 나누는 기준 / 방법에 따라 모델의 학습 결과와 일반화 특성이 달라짐. **일반화 특성을 개선하기 위해서는 Grid Search 에 대해 Cross Validation 을 적용함.**
- **Grid Search + Cross Validation 을 통해 평균 정확도가 제일 높은 모델을 선정함**
- 통상적으로 Grid Search + Cross Validation 을 Cross Validation 이라 지칭하기도함

- **Grid Search + Cross Validation**

  : 데이터셋에서 Train/Validation 을 여러 시나리오로 나누되 Test 는 1 가지만 사용

- **Grid Search + Nested Validation**

  : 데이터셋에서 (Train + Validation) / Teste 를 여러 시나리오로 나누고, Train + Validation 사이에서 여러 시나리오로 나눔 (다중 Test Set 사용)

Heat map of mean cross-validation score as a function of C and gamma

- Grid Search 에 사용하는 Parameter Setting 에 따른 성능을 평가하기 위해 Heat Map 을 사용함. Parameter 단위를 고정된 크기로 나누기 보다는 가변적인 크기 단위가 더 많은 정보를 볼 수 있는 거시적인 관점을 제시할 수 있음.
  (※ 거시적인 변화를 보기 위해 가변적인 단위 Axis 사용)
  (※ 고정적 크기 단위는 거시적인 변화를 보기가 힘듬)

➢ 학습 Metric 선정

- **Loss Function**
  - 목표하는 Output 을 최대한 원하는 정답에 가깝게 만들기 위해 Error 를 최소화 시키는 방향으로 학습해야함
  - 사용하는 모델의 Output 형태와 Label/Target Output 의 형태에 따라 Error 의 형태와 종류가 다름. 그러므로 최소화하려는 Error 의 형태에 따라 적합한 Loss Function 을 선정해서 학습을 수행함.
    (ex : MSE (Mean Squared Error), Cross Entropy Error)

- **Maximum Likelihood Function**
  - 목표하는 Output 의 발생 확률이 최대화되도록 모델을 설계함

- **Classification Evaluation Metric**
  - Accuracy 는 전체 Test 데이터 중 정확하게 Classification 한 비율을 지칭함. 그러나 Accuracy 는 오인식 비율을 반영하지 못하기 때문에 오인식 비율 vs 정확도 상황 / 오인식이 정확도보다 우선시 되는 상황에서 명확한 성능 지표가 될 수 없음. 특히 Imbalanced Dataset (Label 별 개수가 평등하지 않은 데이터셋) 경우 Accuracy 는 각 Label 별 분류 정확도를 반영해줄 수 없음.
  - Imbalanced Dataset 의 경우에는 각 Label 별 분류 정확도와 오인식률을 개별적으로 확인해야함.
  - **Binary Classification / Error & Correction Types**
    : Binary Classification (0 또는 1 분류) 상황에서는 오인식에 대한 정의가 필요함

: Type 1 Error (False Positive / FP) – Negative 인 것을 Positive 라 오인 분류함

: Type 2 Error (False Negative / FN) – Positive 인 것을 Negative 라 오인 분류함

(※ Type 1 Error 와 Type 2 Error 를 모두 관리해야 하지만 둘 사이에 Trade-off 관계가 존재함)

: True Positive (TP) – Positive 인 것을 제대로 Positive 라고 분류함

: True Negative (TN) – Negative 인 것을 제대로 Negative 라고 분류함

■ **Confusion Matrix**

Confusion matrix for binary classification

: Classification 모델에 대해서 오인식 비율과 정확인 인식 비율을 모두 표현한 행렬

: Binary Classification 의 경우 TN, TP, FN, FP 로 구성됨

■ Accuracy, Precision, Recall, Specificity, Sensitivity, f-2 Score

Accuracy (정확도)	$\frac{TP+TN}{TP+TN+FP+FN}$	전체 중에서 제대로 분류한 것만 고려함
Precision (정밀도)	$\frac{TP}{TP+FP}$	Positive 로 분류한 것 중 제대로 Positive 로 분류한 것만 고려함 (분류 성공률에 집중함)
Recall (재현율) (Sensitivity)	$\frac{TP}{TP+FN}$	실제 Positive 인 것 중에서 제대로 Positive 로 분류한 것을 고려함 (실제 Positive 를 제대로 찾는 것에 집중함)
Specificity	$\frac{TN}{TN+FP}$	실제 Negative 인 것 중에서 제대로 Negative 로 분류한 것을 고려함 (실제 Negative 를 제대로 찾는 것에 집중함)
f-2 Score	$2 * \frac{Precision*Recall}{Precision+Recall}$	Precision 과 Recall 을 모두 고려하기 때문에 Accuracy 보다 좋은 지표로서 사용됨 (오인식 비율을 반영하여 성능 평가함)

✓ 만약 데이터셋에 Negative Label 이 더 많은 경우에는 Classifier 가 자동으로 Negative 에 더 집중해서 분류하게 학습됨

✓ 이러한 Classifier 에 대해 Positive 를 좀 더 잘 분류하기 위해서 Recall 을 키우는 방향으로 Parameter 를 수정함. 이 때, Recall 을 높이면서 나머지 지표를 손해 보지만 목표를 달성할 수 있음.

✓ Precision, Recall, Specificity 중 하나의 지표를 높이기 위해 Parameter 를 바꾸면 다른 지표가 손해를 보기 때문에 손해를 최소화하면서 목표치에 도달해야함.

■ **Precision-Recall Curve**

Comparing precision recall curves of SVM and random forest

: Classification 을 결정하는 Threshold 기준에 따라 Positive/Negative 를 결정함

: 이 때, 현재 모델이 사용하는 Threshold 와 Parameter 를 변경해가면서 Precision – Recall 변화 추세를 확인함

: Precision 이 높아질수록 Recall 이 높아진다는 것은 각각 Positive / Negative 에 대한 정확한 분류 확률이 높아진 좋은 Classifier 라는 의미임.

■ **ROC (Receiver Operating Characteristic Curve / Sensitivity – Specificity Curve)**

Comparing ROC curves for SVM and random forest

: Classification 을 결정하는 Threshold 기준에 따라 Positive/Negative 를 결정함

: 이 때, 현재 모델이 사용하는 Threshold 와 Parameter 를 변경해가면서 Sensitivity – Specificity 변화 추세를 확인함

: Specificity (FPR / False Positive Rate)가 줄어들수록 Sensitivity (True Positive Rate)가 높아질수록 Positive 를 제대로 분류하면서 낮은 오인식률을 가진다는 의미임.

■ **AUC (Area Under Curve)**

: ROC 를 하나의 숫자로 나타내는 지표로서 ROC 의 면적인 AUC 를 사용함

: AUC 는 0 ~ 1 사이의 값으로 데이터셋이 아무리 불균형해도 랜덤한 예측에 대해 0.5 를 가짐.

: Imbalanced 데이터셋 상황에서 AUC 는 불균형한 Label 에 대해 각각의 정확도를 반영하여 연산되기 때문에 Accuracy 보다 좋은 지표임.

➢ **Optimizer 선정**

- ML/NN 모델 학습 방향과 방식에 따라 Optimizer 를 선정하여 사용함

➢ **Training 수행 및 수행 결과 분석**

- Training 수행 시 Training Set 과 Validation Set 에 대해 각각 학습을 수행하여 Overfitting 을 방지함.
- Training Set 에서의 Loss Function, Validation Set 에서의 Loss Function 을 각각 확인하여 Overfitting, Underfitting 을 판단함.

5) Machine Learning Dataset Preprocessing 및 Analysis 기법

● **Dataset Preprocessing**

➢ **Dataset Rescaling**

- **Rescaling 이 필요한 이유**
  - Feature 별로 사용하는 단위가 다른 경우 어떤 단위를 사용하는 가에 따라 데이터 분포가 극심하게 달라짐.
    (ex : cm 단위의 데이터는 m 단위로 표현할 경우 서로 차이 없이 분포됨)
  - Distance 기반 모델 (ex : SVM) 경우 사용하는 단위에 따라 성능이 달라지게됨. 그러므로 단위 Scale 에 의한 영향을 최소하기 위해 Feature 들의 사용 단위/Scale 을 통일시켜야함.

- **Standardization (Z-Score / 표준화)**
  - **Standardization 공식 및 효과**

$x' = \frac{x - \mu}{\sigma}$	X' : 표준화된 데이터 (Z-Score) X : 원본 데이터 μ : 특정 Feature 에 대한 X 의 평균 σ : 특정 Feature 에 대한 X 의 표준편차

◆ M 개의 Feature 로 구성된 데이터에 대해 각 Feature 의 평균과 표준편차를 사용하여 각 Feature 에 대해 Z-Score 로 데이터를 Rescaling 하여 표준화함.

◆ Standardization 을 통해 데이터셋의 각 데이터는 1 x M 으로 구성된 Z-Score 데이터로 재구성됨.

◆ Standardization 을 통해 데이터셋의 모든 Feature 는 동일한 평균과 표준편차를 가지게 되며, 동일한 범위의 분포 내에 재구성될 수 있음.

◆ Standardization 을 통해 각 Feature 별로 다른 Scale 을 가지는 것을 동일한 분포를 가지게 하여 Scale 을 통일시킬 수 있음.

- **Normalization**
  - **Min-Max Normalization 공식 및 효과**

$x' = \frac{x - min(x)}{max(x) - min(x)}$	X' : 정규화된 데이터 X : 원본 데이터 max(X) : 특정 Feature 에 대한 X 의 최대값 min(X) : 특정 Feature 에 대한 X 의 최소값

◆ M 개의 Feature 로 구성된 데이터에 대해 각 Feature 의 최대값과 최소값을 사용하여 각 Feature 에 대해 0 ~ 1 사이의 값으로 Rescaling 하여 정규화함

◆ Min-Max Normalization 은 데이터셋의 각 데이터를 0 ~ 1 사이 값으로 구성된 1 x M 데이터로 재구성함.

◆ Min-Max Normalization 을 통해 각 Feature 별로 다른 Scale 을 가지는 것을 0 ~ 1 사이 값으로 통일시킴으로서 Scale 을 통일시킬 수 있음.

➢ Dimensionality Reduction (차원 축소)

I. 차원 축소가 필요한 이유

- 데이터셋의 데이터가 과도하게 많은 Feature 로 구성되어있을 시 학습에 필요한 시간과 학습 속도가 매우 느림.
- Feature 가 너무 많을 경우 학습하기 쉬운 Feature 에만 특화되는 Overfitting 이 발생할 수 있음.

II. Simple Projection 의 문제점 – Swiss Roll 문제

- Feature 의 개수를 줄이기 위해 데이터를 특정 Feature (Dimension)로만 구성된 공간에 Projection 함. 그러나 Feature (Dimension)을 줄이는 행위는 정보 손실이 발생할 뿐만 아니라 오히려 Classification 하기 어려운 분포가 되는 경우가 있음.

- 3D Swiss Roll 예제가 대표적인 경우임. 3 개의 Feature 로 구성된 데이터셋은 3 차원 좌표계에 표현될 수 있음. 그러나 데이터셋을 1 개의 Feature (Dimension)을 제거하고 2D 평면에 바로 Projection 할 경우 각 Label 별 데이터가 오히려 섞이는 현상이 발생함. 그러므로 단순히 Projection 하는 것은 효과적인 데이터셋 전처리 방법이 아니라는 것을 알 수 있음.

III. Manifold Approach

- Manifold 는 다양체라고도 하며 국소적으로 Euclidean 공간과 닮은 위상 공간임

	전체 구조는 원이지만 노란색 구간에서만의 미시적인 관점에서 원 전체를 느낄 수 없으며, 미시적으로 직선이라 생각할 수 있음.

- Manifold 는 국소적인 구간과 미시적인 관점에서 전체 구조를 다른 구조와 위상적으로 동형이라는 것을 사용하여 데이터셋의 Feature 를 재정의하고, 데이터셋을 보다 이해하기 쉬운 구조로 재구성함.

- Manifold 를 응용하여 고차원/다차원에서 복잡한 데이터셋을 재구성하고, 이를 기반으로 모델 학습을 수행하는 것을 Manifold Learning 이라함.

- **Isomap (국소적인 분포에 집중하여 Clustering 하는 기법)**

: 3D Swiss Roll 데이터와 같은 경우 각 데이터간 관계를 Euclidean Distance 로 표현하면 서로 간의 거리감과 실제 관계가 괴리가 생김.
: 국소적인 Neighborhood 그래프를 생성하여 데이터셋 내부의 데이터간 관계도 Visualization (국소적인 Euclidean Distance) + 고차원적인 Euclidean Distance 정보 (Geodesic Info)가 어느 정도 보존되게하는 Manifold 기반 차원 축소 기법

- **LLE (Locally Linear Embedding)**

: Select neighbors → Reconstruct with linear weights
→ Map to embedded coordinates
: 각 데이터에 대해 Linear Weight 를 반영한 K Closest Neighbor 를 찾음. 그리고 각 데이터를 새로운 축/Dimension 에서 해석하더라도 Linear Weight 가 보존되는 축/Dimension 을 찾음
(※ Linear Weight 가 보존되게 차원 축소 수행)

- **t-SNE**

: 데이터 주변에 가까운 데이터일수록 Weight 를 크게 줘서 더 가깝게 / 멀리 있는 데이터일수록 Weight 를 작게 줘서 멀게 취급될 수 있도록 차원 축소를 수행하는 기법

: 주변에 가까운 데이티를 국소적으로 보존하고 해석하는 데에 집중함

: Local 한 데이터와의 Grouping 을 알 수 있으나 Global 데이터 분포는 알기 힘듬

① **Local Weight 정의 (Gaussian 분포와 같은 확률 분포 모델을 사용하여 데이터간 거리에 따른 Weight 를 부여함)**

② **Gaussian 분포 대신 t 분포를 적용하여 Local Weight 를 더 독보이게 만들어서 데이터와 이웃 관계를 더 강조시킴**

③ **저차원 / 고차원 각각에서 동일한 Local Weight 가 반영되는 새로운 축/Dimension 을 찾음**

IV. PCA (Principle Component Analysis)

- PCA 는 데이터셋의 전체를 최대한 많이 설명할 수 있는 새로운 Feature Axis (Dimension Axis)를 새로 산출하는 방법임.
- PCA 는 표준화된 데이터셋이 특정 Eigen Vector (고유 벡터)에 Projection 되었을 시에 최대한 많은 데이터를 수용되어야함. 최대한 많은 데이터셋을 수용/설명할 수 있는 특정 Eigen Vector 를 구하기 위해 Projection 된 결과의 Variance (Eigen Value)가 최대가 되는 경우를 찾음.

> **※ Eigen Vector 를 사용하는 이유 : 돌아가지 않는 고정된 Axis 필요**
> **새로운 Feature/Dimension Axis 로 사용될 것이기 때문에 Matrix 에 곱해져도 Translation 만 변하고, Phase 가 유지되는 Vector 가 필요함**

- PCA 는 다수의 Eigen Vector 인 PC (Principle Component)를 제공함. M 개의 Feature 로 구성된 데이터셋은 PCA 를 통해 최대 M 개의 PC 를 생성할 수 있음.

- 1st PC 는 PC 중 제일 설명력이 강한 새로운 Feature/Dimension Axis 임. PCA 를 통해 구해진 PC 를 새로운 Feature 로 사용하여 데이터셋을 재구성함으로서 설명력을 최대한 유지하면서 이전보다 더 적은 Feature 로 데이터셋을 구성할 수 있음.

- **Dataset Analysis**
  - **Linearity 판단 : Correlation Matrix**

    I. Correlation
    - 특정 데이터 X, Y 간의 선형성 여부를 알려주는 지표
    - 선형성 여부와 강도만 알려줄 뿐 선형성의 크기를 알려주지는 않음 (Correlation ≠ 기울기)

    II. Correlation Matrix 를 사용하는 이유
    - 데이터셋의 Feature 간 독립적인 것이 이상적이지만 항상 모든 데이터셋의 Feature 는 독립적이지 않음.
    - 특정 Feature 의 변화가 다른 Feature 에 영향을 주면서 Output 에도 영향을 주기 때문에 그에 따른 Feature 간 선형성 관계 여부와 강도를 알아야함.
    - Correlation Matrix 를 통해 다수 Feature 간 선형성 (Multi-Colinearity)가 발생할 경우 그에 따라 적합한 모델과 전처리 기법을 사용해야함.

6) Machine Learning 설계 주요 중점 : Model Fitting

Fig 2: Overfitting vs Underfitting

(https://meditationsonbianddatascience.com/2017/05/11/overfitting-underfitting-how-well-does-your-model-fit/)

- **Bias – Variance Decomposition**

$$MSE = E(Y_i - \hat{f}(x))^2 = (E[\hat{f}(x)] - f(x))^2 + E(\hat{f}(x) - E[\hat{f}(x)])^2 + E[\varepsilon^2]$$

Expected squared error between true and estimated model	$Bias^2$	$Variance$	$Irreducible\ error$
			• Irreducible error is the error that can't be reduced by creating good models. It is a measure of the amount of noise in our data.

  - **Bias-Variance Decomposition 을 통해 모집단 vs Sample 간의 Error 가 Bias 와 Variance 의 조합으로 구성되는 것을 볼 수 있으며, Bias 와 Variance 의 조합에 따라 Error 가 모집단과 Sample 사이의 Error 가 최소화될 수 있음.**
  - **Bias 가 생겨서 모집단의 Mean 과 Sample 의 Mean 이 서로 조금 어긋나더라도 Variance 가 좁으면 모집단과 가까운 위치에 있기에 모집단 vs Sample 간의 Error 가 적을 수 있음.**

- **Overfitting**
  - 학습 과정 중 Training Loss 와 Validation Loss 가 서로 유사하지 못하면서 Validation Loss 가 Training Loss 보다 높은 상황
  - **발생하는 이유**
    - 과도한 양의 Feature 로 구성된 매우 복잡한 모델을 학습하는 과정에서 학습하기 쉬운 Feature 에 대해서만 하면서 모델이 Training Set 에 특화되어 버리기 때문임.
    - Training Set 에 특화되어버리기 때문에 다른 데이터셋인 Validation Set 에 대해서는 낮은 성능을 보임.

- **해결방법**

  Overfitting 이 발생하는 이유는 데이터셋이 너무 많은 Feature/Dimension 으로 구성되어있는 고차원 모델이기 때문임. 많은 Feature/Dimension 으로 구성된 만큼 데이터를 더 사용하거나 Feature/Dimension 을 축소시켜서 모델을 간소화시켜야함

  - **데이터셋의 Feature/Dimension 개수를 줄이는 작업** (Dimensionality Reduction 또는 Feature 일부만 선정해서 학습)**을 통해 입력 학습 데이터셋을 간소화시킴**
  - **더 많은 학습 데이터를 모아서 제공함**
  - **학습 데이터의 노이즈를 줄임**

- **Underfitting**
  - 학습 과정 중 Training Loss 와 Validation Loss 가 낮게 유지되지 못하면서 Training Loss 가 Validation Loss 보다 높거나 같은 상황
  - **발생하는 이유**
    - 학습에 사용되는 Feature 가 너무 적어서 학습 데이터를 모두 표현할 수 있는 모델을 만들지 못하는 상황임.
    - Training Set 에서 제대로 학습을 하지 못했기에 Training Set 에서 낮은 성능을 보이며, Validation 에서도 높은 성능을 보이지 못함.
  - **해결방법**

    Underfitting 이 발생하는 이유는 데이터셋에 사용되는 Feature/Dimension 이 적어서 모델이 학습할 정보량이 부족하기 때문임. 그러므로 데이터셋에 사용되는 Feature 의 개수를 증가시켜야함.

    - 학습 데이터셋에 더 많은 Feature/Dimension 를 추가해서 낮은 차원의 데이터셋을 고차원으로 개선하여 모델이 각 데이터로부터 더 많은 정보를 학습할 수 있게 만듬
    - 기존에 사용하던 Feature 보다 더 좋은 Feature 를 선정해서 학습함.
    - 모델의 제약을 줄임.

3. Supervised Learning

1) Linear Regression

- **주요 특성 및 원리**
  - X, Y 사이에 서로 Linear 한 관계가 있다는 가정하에 X, Y 의 Linear 한 관계를 표현할 수 있는 수식 (모델 / Y = f(X))을 찾아내는 방법.
  - X 는 N x M 으로 구성된 데이터셋에서 1 x M Vector 형태로 M 개의 Feature 로 구성된 데이터임. Y 는 X 의 Feature 의 조합으로 구성된 모델의 Output 임.
    Linear Regression 모델 : $Y = \sum_{i=1}^{M} X_i = w_0 * X_0 + w_1 * X_1 + w_2 * X_2 \ldots + w_M * X_M$
  - 모든 ML 기법의 기본이 되는 모델 학습 방법이며 여러 종류의 데이터셋을 최대한 Linear Regression 의 형태에 유사하게 만들어서 학습을 수행하려함.
  - 모델의 Prediction Y'와 데이터셋의 정답 Y 간의 Error 를 최소화하는 방향으로 Gradient Descent 를 수행하여 Training Set 과 Validation Set 에서 최소의 Error 를 가지는 f(X) 모델을 구함

- **한계점**
  - Input X 와 Output Y 사이에 서로 Linear 한 관계가 있다는 가정하에 모델을 도출하는 학습을 하기 때문에 Non-Linear 한 데이터셋, Multi-Colinearity 데이터셋에 대해서는 학습할 수 없음.
  - Non-Linear 한 상황, Multi-Colinearity 상황에 대해 데이터셋의 Feature 를 재구성하여 데이터를 Linear 한 분포로 유도하는 기법이 요구됨

2) Ridge / Lasso / ElasticNet

- **주요 특성 및 원리**
  - **Regularization (Shrinkage)**
    : Multi-Colinearity 상황에서 Input X 의 Feature 사이에 Linearity 가 발생하여 모델 추정 시 각 Feature 에 대해 독립적인 Weight 를 보장하지 못하게됨.
    : 이에 대해서 Regularization 을 통해 Multi-Colinearity 가 발생하는 Feature 에 대해 Penalty 값을 부여하여 유의미한 Feature 만 모델 생성에 기여하도록함.
  - **Ridge**
    : L2 Penalty 를 사용하여 Multi-Colinearity 를 가진 Feature 의 Weight 를 줄임으로서 모델에 대한 기여도를 약화시킴.
    : L2 형태로 Penalty 가 들어가기 때문에 완전히 0 이 되지 못하고, 0 에 근사화됨.
  - **Lasso**
    : L1 Penalty 를 사용하여 Multi-Colinearity 를 가진 Feature 의 Weight 를 0 으로 수렴하게하여 유의미한 Feature 만 남게하는 Feature Selection 을 수행함
  - **ElasticNet**
    : Ridge 와 Lasso 를 같이 사용하여 Lasso 가 과도하게 Feature 를 없애는 것을 Ridge 를 통해 완화시킴.

- **한계점**
  - Ridge : Feature 를 0 으로 만들지 않기 때문에 불필요한 Feature 를 없애지 못함.
  - Lasso : Colinear 한 Feature 가 다수 존재하는 경우 1 개만 Penalty 를 가할 수 있음.

3) KNN (K-Nearest Neighbors)

Fig 3: KNN Classification Process

(https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn)

- **주요 특성 및 원리**
  - 전체 데이터셋 분포에서 입력 데이터와 거리가 가까운 데이터를 K 개 선정함. 선정된 K 개의 데이터의 Label 에 대해 Majority Voting 을 수행하여 입력 데이터에 대한 Label 을 정함.
  - 입력 데이터와 가깝게 배치된 데이터들이 서로 유사한 Label 을 가진다는 가정하에 설계된 Classification 방식.
  - 별도의 학습 과정이 필요 없으며, 데이터만 충분히 있다면 Classification 을 손쉽게 구현할 수 있음.

- **한계점**
  - 효과적인 Classification 을 위해서는 많은 데이터를 요구함
  - Label 별로 Linear 한게 분포된 데이터에 대해서는 효과적이나 Non-Linear 하게 분포된 경우 효과적인 Classification 을 수행할 수 없음
  - 효과적인 데이터셋의 분포 및 K 개 최단 거리 데이터 선정을 위해 상황에 따른 전처리 과정과 최단 거리 연산이 요구됨.

4) Logistic Regression

Fig 4: Logistics Regression

(https://nittaku.tistory.com/478)

- **주요 특성 및 원리**
  - Prediction 하고자하는 Output Y 가 범주형 (Categorical)이면서 0 또는 1 인 경우 사용하는 회귀 분석임.
  - 기존의 Linear Regression 은 Continuous Input X, Continuous Output Y 에 대해 Fitting 될 수 있는 모델을 찾는 것이 주요 목적이었음. 그러나 Discrete 한 Y 로 구성된 데이터셋에 대해서는 효과적으로 Fitting 할 수 없음.
  - Discrete Output 에 대한 Fitting 을 위해 Linear 한 직선이 아닌 Curve 를 사용하여 Fitting 함.
  - 이를 위해 Input X 에 대해 각각 Output Y 가 발생할 수 있는 확률을 통해 Odds 를 구하고, Logit 함수를 도출하여 Curve Fitting 함.

- **한계점**
  - Output Y 가 Discrete 한 상황(Yes or No / 0 or 1)에 대해서 적용할 수 있기 때문에 사용 가능한 상황은 매우 제한적임.

5) Naive-Bayes Classifier

- **주요 특성 및 원리**
  - Y 라는 Target Output 이 있을 때, Feature X 에 대한 확률 분포가 있다고 가정함.
  - Feature 가 주어졌을 때 확률이 제일 높은 Label 을 찾을 확률(P(Y|X))을 찾도록 모델을 학습시킴. Bayes Rule 을 이용하여 간접적으로 데이터로부터 답을 유추하도록 학습됨.

6) SVM (Support Vector Machine)

- **주요 특성 및 원리**
  - Linear Regression 은 Training Set 내에서 Prediction Y'와 Tareget Output Y 간의 Error 를 최소화 시키는 방향으로 모델/Decision Boundary 를 산출하도록 학습됨.
  - SVM 은 Linear Regression 과 다르게 전체 데이터셋에서 Label 간의 간격이 최대가 되도록 Decision Boundary 를 산출하도록 하는 기법임.
  - 전체 데이터셋에 대해서 직접 Decision Boundary 를 산출하기 때문에 별도의 학습이 요구되지 않음.
  - Non-Linear 한 데이터셋 분포에 대해서는 Kernel Trick 을 적용하여 Dimensionality Reduction 과 유사하게 데이터셋을 좀 더 Linear 한 분포로 재구성함.

- **한계점**
  - SVM 은 기본적으로 Label 간의 거리를 최대화하는 방향으로 Decision Boundary 를 만들기 때문에 거리를 산출하는 데에 영향을 주는 Feature 의 단위에 크게 영향을 받음. 데이터셋의 Feature 에 사용되는 단위/Scale 에 따라 데이터 분포가 바뀌고 그에 따라 Decision Boundary 의 성능이 크게 달라짐. 그러므로 데이터셋에 대한 Rescaling 전처리가 요구됨.
  - Non-Linear 한 데이터셋 분포에서 Kernel 의 종류에 따라 성능이 크게 달라짐.

- **SVM Kernel Trick**

  - 오리지날 Non-Linear 데이터 분포에 바로 SVM 을 적용하는 것은 효과적이지 않음. 왜냐하면 SVM 은 Distance 를 기반으로 만들어진 Classification 기법이기 때문임.
  - 오리지날 Non-Linear 데이터 분포를 Linear 하게 변환시킬 수 있는 Kernel 을 사용하게되면 변형된 데이터 분포는 Feature Space (Higher Dimension)에서 Linear 한 분포를 가질 수 있게됨. 오리지날 Input Space 보다 Feature Space 에서의 데이터 분포가 SVM 에 더 적합함.
  - Non-Linear 데이터셋에 Kernel 을 적용하여 데이터셋 분포를 Linear 하게 변형하여 Linear Model 이 적용되게 할 수 있음.

7) Decision Tree

- **주요 특성 및 원리**

Information	$I(E)=\log_2 \frac{1}{p(x)}=-\log_2 p(x)$	**불확실성의 양 특정 상황이 발생 확률이 낮으면 그에 따라 불확실성이 높아 Info 증가**
Entropy	$H(X)=E(I(X))=\sum_i p(x_i)I(x_i)=-\sum_i p(x_i)\log_2 p(x_i)$	**Impurity / Uncertainty Information 에 대한 기댓값**
Information Gain	IG(X,Y)=H(X)-H(X|Y) (information before split) – (information after split) =(uncertainty before split) – (uncertainty after split)	**Decision Tree 에서 특정 Feature 에 대해 선택을 했을 경우 감소되는 불확실성의 양**

- ➢ 스무고개 하듯이 데이터셋의 Feature 를 하나씩 확인하면서 각 Feature 별로 조건 만족 여부에 따라 Tree 를 탐색하면서 Classification 을 수행함.
- ➢ Decision Tree 에서 매 선택을 해야하는 순간에 Uncertainty 가 존재함.
- ➢ 매번 선택을 할 때마다 Uncertainty 가 최대로 감소하는 방향으로 Tree 를 탐색함.

(※ 선택을 할 수록 더 확고한 결론에 도달하는 것을 목표함)

- ➢ Decision Tree 에서는 Information 을 Uncertainty 의 양이라 정의하며, Information 에 대한 기대값을 Entropy 라 지칭함. Information Gain 은 선택하면서 Uncertainty 가 얼마나 감소하는지 보여주는 지표임.
- ➢ **Decision Tree 는 Information Gain 이 큰 방향으로 Tree 를 탐색함**으로서 선택할 때마다 **더 확고한 Classification 결론에 도달하게 만드는 Greedy Algorithm 임.**

- **주요 문제점**
  - ➢ Tree 를 구성하는 Feature 를 어떤 것을 사용할 것인가?
  - ➢ Tree 의 가지치기 (Split)을 어느 지점까지 할 것인가?
  - ➢ 새로운 Feature 데이터가 추가될 때마다 새로운 가지치기 (Split)이 요구됨.
  - ➢ Decision Tree 1 개가 만들어지면 수정하는 것이 매우 힘듬.

8) Random Forest

Fig 5: Random Forest Structure / Majority Voting

- **주요 특성 및 원리**
  - Decision Tree 를 1 개 만들면 Classification 방식이 고정되는 단점이 있음.
  - 이를 보완하기 위해 Random Forest 는 데이터셋의 모든 Feature 를 사용하는 Decision Tree 와 달리 데이터셋의 Feature 의 일부를 다양하게 조합하여 여러가지 Decision Tree 를 만듬.

    (※ Bagging : 동일한 데이터셋에 대해 일부 Feature 만 사용해서 여러 버전을 준비함 / 중복 추출)
  - 다양한 Feature 의 조합으로 만들어진 다수의 Decision Tree 는 Input X 에 대해 각각 Classification 결과를 도출함. 도출된 결과에 대해 Majority Voting 을 수행하여 최종 Classification 결과를 결정함.

- **한계점**
  - Decision Tree 를 여러 개로 구성되기 때문에 대규모의 Tree 를 저장할 수 있는 메모리 공간이 요구됨.
  - Decision Tree 여러 개를 모두 탐색해야하기 때문에 Time-Critical 한 시스템에 적용시키기에는 부적합함.

9) ANN (Artificial Neural Network)

- **Artificial Neuron (Perceptron) Modeling**

Fig 6: Artificial Neuron (Perceptron) Mathematical Modeling

- 인간의 뉴런은 외부 전기적 자극을 받아 종합되어 외부로 전기적 자극을 방출함.
- Artificial Neuron (Perceptron)은 인간의 뉴런의 작동 과정의 수학적 모델임.
  - **외부 전기적 자극을 받아 종합**

    = 입력 데이터에 대해 Weighted Sum 을 계산 (= Linear Regression)
  - **외부로 전기적 자극을 방출**

    = Weighted Sum 을 Activation Function 에 대입하여 다음 단계에 연결된 뉴런에 입력으로 사용
- Artificial Neuron 은 ReLU, Leaky ReLU, Sigmoid 등 Non-Linear 함수를 Activation 함수로 사용함. 왜냐하면 Non-Linear 한 Input/Output 데이터셋에 대해 Non-Linear 한 Prediction 을 수행하기 위해 Non-Linear Activation Function 을 사용함.

10) Perceptron

- **주요 특성 및 원리**
  - Perceptron 은 Prediction Output Y'와 Target Y 의 Error 가 최소화 되도록 Gradient Descent 를 사용하여 Weight 를 최적화하는 학습을 수행함.
  - 단일 Perceptron 은 Input 에 대한 Weighted Sum 에서 Gradient Descent 를 통해 Weight 를 최적화하기 때문에 Linear Regression 과 유사함.

- **한계점**
  - 단일 Perceptron 으로는 Non-Linear 데이터셋에 대해 효과적인 Prediction 을 할 수 없음. 이를 극복하기 위해 다수의 Perceptron 을 여러 Layer 로 구성한 Multi-Layer Perceptron (MLP) / Neural Network 를 사용함.

11) Neural Network / MLP : Multi-Layer Perceptron

Fig 7: Neural Network / MLP Structure

- **주요 특성 및 원리**
  - Non-Linear 데이터셋에 대한 단일 Perceptron 의 한계를 극복하기 위해 다수의 Perceptron 으로 다중 Layer 네트워크를 구성하여 Linear/Non-Linear 한 상황 모두에서 Decision Boundary 를 생성할 수 있음.
  - 다수의 Perceptron 이 연결되어있는 구조이기 때문에 Linear Regression 과 유사한 단일 Perceptron 과 달리 Error Optimization 을 위한 Gradient Descent 가 매우 복잡함. 이를 극복하기 위해 Chain-Rule 을 기반으로 Backpropagation 을 사용함.

Fig 8: MLP Decision Boundary

  - 더 많은 Perceptron 과 Layer 로 Neural Network 를 구성할수록 Non-Linear 한 데이터셋에 대해 더 효과적인 Non-Linear Decision Boundary 를 만들 수 있음.

- **Backpropagation**

Fig 9: Backpropagation Calculus & Chain Rule

(https://www.youtube.com/watch?v=tIeHLnjs5U8)

- Prediction Output Y'과 Target Output Y 간의 Error 값은 Neural Network 최종 Output Layer 에서 도출할 수 있음.
- 그러나 역방향으로 Prediction Y'의 변화에 따른 Error 변화에 비례해서 이전 Layer 의 모든 Weight 값을 업데이트하기 위해 이전 Layer 를 매번 방문하는 것은 매우 비효율적임. 특히 Layer 가 늘어날 수록 Weight 업데이트에 소요되는 시간이 기하급수적으로 증가함.
- 이를 해결하기 위해 Prediction Output 대비 Error (Loss)의 변화량(편미분)을 Chain Rule 를 통해 각 Layer 별 Input 대비 Output 변화량 (편미분)의 조합으로 재정리하여 보다 쉽게 Gradient Descent 에 필요한 편미분을 모두 산출할 수 있음.
- Prediction Output 대비 Error (Loss)를 편미분하면서 이전 Layer 의 Activated 된 Weighted Sum 의 편미분이 밖으로 연속적으로 꺼내지게됨. 각 Layer 별 Weighted Sum 의 편미분이 Backpropagation 을 통해 모두 구해지기 때문에 모든 Layer 의 Weight 업데이트에 필요한 업데이트량을 알 수 있음.
- Backpropagation 을 통해 어떤 Weight 가 다음 Layer 의 Output (최종적으로 Error/Loss Fucntion)에 연속적으로 얼마나 영향을 주는지 알 수 있음.

● Backpropagation Node 단위 해석

- Neural Network의 학습 대상은 Neuron Node간 연결되는 Weight값이며, 이 Weight값은 Prediction Error L에 비례해서 변경되어야함
- 그러므로 Prediction Error L에 대한 Weight의 변화량을 각 Weight에 대한 L의 편미분으로 표현함
- 각 Node의 Weight는 자신이 Error에 얼마나 기여하는 지만 집중함
- Prediction Error L에 대한 Weight의 변화량은 바로 최종 Layer의 Weight를 제외하고 바로 구할 수 없음. 왜냐하면 각 Node별 연산을 거쳐가기 때문임.
- 각 Node가 알 수 있는 것은 다음 Weight – 현재 Weight의 편미분만 구할 수 있음.
- 그러므로 각 Weight가 Error에 대해 얼마나 기여하는지를 다음 Layer의 Weight의 편미분으로 Chain Rule을 통해 연속적으로 표현함.
- 각 Weight별 Error에 대한 기여도를 알 수 있고, 이를 기반으로 Weight를 Learning Rate에 비례해서 구할 수 있게됨.

Prediction Error L에 대한 현재 Weight의 변화량을 Learning Rate 만큼 반영하여 현재 Weight를 업데이트 시킴

$$t = x + y, \qquad z = t^2$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x}$$

$$\frac{\partial z}{\partial t} = 2t, \frac{\partial t}{\partial x} = 1$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 2t \cdot 1 = 2(x+y)$$

➢ 덧셈 연산 Node 에 대한 Backpropagation 결과

$$z = x + y, \qquad \frac{\partial z}{\partial x} = 1, \frac{\partial z}{\partial y} = 1$$

더하기 연산은 Backpropagation 시 이전 Node 로 Output 대비 Error 미분이 바로 곱셈으로 반영됨

➢ **곱셈 연산 Node 에 대한 Backpropagation 결과**

$$z = xy, \qquad \frac{\partial z}{\partial x} = y, \frac{\partial z}{\partial y} = x$$

곱셈 연산은 Backpropagation 시 이전 Node 로 Output 대비 Error 미분에 현재 Input Weight 을 제외한 나머지 Input Weight 를 바로 곱해서 전달됨.

(Input 이 Swap 되어 Backpropagate 됨)

(해당 Node 에서의 Input 만 사용해서 Error 변화량을 구함 / Node 별 Error 산출 가능)

➢ **ReLU 연산 Node 에 대한 Backpropagation 결과**

$$y = \begin{cases} x & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

$$\frac{\partial y}{\partial x} = \begin{cases} 1 & (x > 0) \\ 0 & (x \leq 0) \end{cases}$$

ReLU 를 사용할 경우 Input 이 양수인 경우만 Backward 에 반영되어 이전 Node 로 편미분이 전달됨.

(ReLU 는 양수일 때만 미분이 가능한 것을 응용)

➢ **Sigmoid 연산 Node 에 대한 Backpropagation 결과**

$$\begin{aligned}\frac{\partial L}{\partial y} y^2 exp(-x) &= \frac{\partial L}{\partial y} \frac{1}{(1+exp(-x))^2} \exp(-x) \\ &= \frac{\partial L}{\partial y} \frac{1}{1+exp(-x)} \frac{exp(-x)}{1+exp(-x)} \\ &= \frac{\partial L}{\partial y} y(1-y)\end{aligned}$$

Sigmoid 함수는 실질적으로 여러 단계의 연산으로 구성된 것과 같음

각각 단계별 연산의 Backpropagation 결과가 합쳐진 것이 Sigmoid 의 미분과 동일하게 반영됨

- **주요 문제점**

  - **Vanishing Gradient**

    - Deep Learning 과 같이 매우 많은 Layer 로 구성된 Neural Network 의 경우 Output Layer 에서 Input Layer 로 갈 수록 미분하는 횟수가 증가함.
    - 미분 횟수가 너무 증가하게 되면 결국 Input Layer 에 가까운 Weighted Sum 의 Error/Loss 에 대한 편미분은 거의 0 에 수렴하게 되면서 Weight 업데이트량이 0 에 수렴하게됨. 이로 인해 Input Layer 쪽에 가까운 Weight 가 업데이트 되지 않는 Vanishing Gradient 현상이 발생함.
    - 이는 각 Layer 의 Weighted Sum 이 거치는 Activation Function 이 미분을 여러번 거치면서 점점 0 에 수렴하기에 발생하는 현상임.
    - 이를 해결하기 위해 미분을 해도 0 에 수렴되지 않는 Non-Linear Function 인 ReLU, Leaky ReLU 등을 Activation 함수로 사용함.

  - **매우 많은 데이터량 및 Computing 연산 능력 요구**

    - Neural Network 의 경우 Layer 가 깊어지고 Perceptron 을 많이 사용할수록 그에 따라 사용되는 Weight 의 개수, Backpropagation 을 위해 저장하는 데이터의 개수가 기하급수적으로 증가함.
    - 또한 CPU 의 경우 병렬적인 연산보다는 단일 고속 연산에 특화되어있기 때문에 CPU 를 이용한 Backpropagation 및 Neural Network 학습은 매우 느림.
    - 이를 해결하기 위해서 CPU 대신 병렬 연산에 특화된 GPU 를 사용해야함. 그리고 Neural Network 를 작동시키는 프로그램이 데이터를 시스템에 등재시키기 위해 대량의 RAM 와 VRAM 을 사용해야함.

4. Unsupervised Learning

1) Clustering

- **Hard Clustering (K-Means, Hierarchical Clustering)**
  - 1 개의 데이터를 1 개의 Cluster 에만 소속되게 하는 방식
- **Soft Clustering (Fuzzy C-Means, Gaussian Mixture Model)**
  - 1 개의 데이터가 복수의 Cluster 에 여러 비율로 소속되게 하는 방식
  - 여러 특성을 가지거나 소속되는 것을 허용함으로서 좀 더 유연한 판단을 허용함
  - 부분적인 소속감
- **좋은 Clustering 의 기준**
  - **High Intra-Class Similarity : Cluster 내에서는 높은 유사도를 가져야함**
  - **Low Inter-Class Similarity : Cluster 간 서로 낮은 유사도 / 높은 구별성을 가져야함**
  - **Hidden Pattern (ex : 타원형 구조)에 대해서도 효과적으로 Clustering 할 수 있어야함**

2) K-Means

- **주요 원리**
  - 최초 K 개 Cluster 의 Centroid (Codebook)은 랜덤하게 배치됨.
  - 각 데이터는 자신에게서 제일 가까운 Centroid 기준으로 Cluster Label 이 배정됨. 각 데이터가 어느 Cluster 에 속하게 되는 지에 대한 정보는 Partition Matrix 에 0 또는 1 로 업데이트됨. **(Partition Matrix Update)**
  - 각 Centroid 는 자신의 Cluster 에 속한 데이터의 Mean (평균) 위치로 업데이트 되어 이동함. **(Centroid / Codebook 좌표 Update)**
  - 이후 각 데이터는 다시 자신에게 제일 가까운 Centroid 기준으로 Cluster Label 이 배정되고, 해당 Cluster 의 Centroid 가 자신의 Cluster 에 속한 데이터의 Mean 의 위치로 이동함. 이러한 과정이 Recursive 하게 발생하면서 Cluster 의 Centroid 가 데이터 클러스터의 중심 (Mean) 위치로 이동하게됨.

$$E(\Gamma, V) = \sum_{i=1}^{k} \sum_{j=1}^{n} \gamma_{ij} \| x_j - \bar{v}_i \|^2$$

Data set: $X = \{x_1, x_2, \cdots, x_n\}$
Clusters: $C_1, C_2, \cdots C_k$
Codebook : $V = \{\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_k\}$
Partition matrix: $\Gamma = \{\gamma_{ij}\}$

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \bar{x}_j \in C_i \\ 0 & \text{otherwise} \end{cases}$$

  - 이와 같은 Cluster Centroid 의 Recursive 한 이동은 각 클러스터의 소속 데이터와의 거리값 총합이 최소화되는 순간 (각 데이터가 자신에게 제일 가까운 Centroid 에 속하게 되는 순간 / Distance Error E 최소화 순간)에 멈추게됨

- **장점**
  - 데이터 클러스터 중심 (Mean)으로 Cluster 의 Centroid 를 이동시키기 때문에 Cluster 의 구성 형태는 거의 항상 Circular (구형)을 띄게 됨.
  - 쉽게 구현할 수 있음.
  - 대규모 데이터셋에 대해 쉽게 적용할 수 있음.

- **한계점**
  - 최초 Centroid (Codebook)의 초기화 지점에 따라 Clustering 결과가 크게 달라짐.
  - 데이터 클러스터 집단에 멀리 떨어진 Outlier 마저 포함시키려는 시도함. Outlier 에 의해 Cluster 가 끌어당겨져서 Clustering 결과가 달라지며, Outlier 에 취약함.
  - 데이터 클러스터 중심으로 Cluster Centroid 를 이동시키려는 특징 때문에 원형 구조로 배치되지 않은 데이터 클러스터를 제대로 분리 못하는 단점이 있음.
  - 최초 Centroid (Codebook)의 개수 K 가 몇 개 인가에 따라 Clustering 의 효과가 크게 달라짐.
  - High Dimension (많은 Feature 로 구성된 데이터셋 상황) 에서 Euclidean Distance 가 쉽게 발산하기에 High Dimension 상황에 적용하는 것은 부적합하며, 연산 속도가 상대적으로 매우 느려짐.

    (※ Euclidean Distance 는 High Dimension 일수록 쉽게 발산함)

- **적절한 Cluster Centroid 개수 K 를 어떻게 선정하는가? – "Elbow Finding" / "Knee Finding"**

  - K-Means 에서 최소화하고자 하는 것은 각 데이터와 Cluster Centroid 간 거리 총합 (Distance Error E)가 최적화 대상 / Objective Function 임.
  - K 값을 변경해가면서 Objective Function 이 급격하게 감소하는 지점이 최초의 최적 K 값이라 할 수 있음. 해당 K 값을 기준으로 값을 변경하면서 Clustering 결과를 확인함으로서 최적의 K 를 찾아갈 수 있음.

3) Fuzzy C-Means

- **주요 특성 및 원리**
  - K-Means 의 연장선으로 Fuzzy 에서 사용하는 모호한 경계의 개념을 K-Means 에 적용하여 각 데이터에 부분적 소속을 허용하는 Soft Clustering 기법
  - 1 개의 데이터가 1 개의 Cluster 에만 속하는 것이 아닌 여러 Cluster 에 거쳐서 여러 비율로 나눠서 속하게 허용함.
  - Cluster 사이의 경계에 존재하는 데이터에 대해 양쪽에 부분적 소속을 허용함으로서 각 데이터의 Clustering 에 대해 보다 유연성을 허용함.

- Minimize functional

$$E(U,V) = \sum_{i=1}^{k}\sum_{j=1}^{n}(u_{ij})^m \|x_j - v_i\|^2 \qquad \text{subject to} \quad \sum_{i=1}^{k} u_{ij} = 1 \qquad \forall j = 1,\ldots,n$$

- fuzzy partition matrix

$$U = \left[u_{ij}\right]_{k\times n}$$

Data set: $X = \{x_1, x_2, \cdots, x_n\}$
Clusters: $C_1, C_2, \cdots C_k$
Codebook : $V = \{\bar{v}_1, \bar{v}_2, \cdots, \bar{v}_k\}$
Partition matrix: $\Gamma = \{u_{ij}\}$ $u_{ij} \in [0,1]$

  - K-Means 와 유사하게 각 데이터와 Cluster Centroid 사이의 거리 총합이라는 Objective Function 을 사용하지만 Soft Clustering 이기 때문에 Partition Matrix 가 0 또는 1 이 아닌 0 ~ 1 사이의 실수값 (Fuzziness)로 구성됨. 이를 통해 여러 Cluster 에 걸쳐서 부분적으로 소속되는 것을 구현함.

- **한계점**
  - 최초 Centroid (Codebook)의 초기화 지점에 따라 Clustering 결과가 크게 달라짐.
  - 최초 Centroid (Codebook)의 개수 K 가 몇 개 인가에 따라 Clustering 의 효과가 크게 달라짐.
  - Partition Matrix 의 내부 값을 0 ~ 1 사이의 실수값을 사용하여 부분적 소속을 허용하지만 최종적으로 각 데이터가 어느 Cluster 에 속하는지 판단을 해야함. 이에 대한 판단은 Membership Cut Off Value 소속 기준값 및 소속 기준 조건에 의해서 결정됨. Membership Cut Off Value 에 따라 Clustering 결과가 크게 달라짐.

4) DBSCAN (Density-Based Spatial Clustering of Application with Noise)

- **주요 원리**
  - K-Means 와 같이 Global 한 Euclidean Distance 기반 판독은 Outlier 까지 모두 포함하면서 Clustering 이 Skew / 치우쳐지는 문제와 비원형 구조에 대해 효과적으로 Clustering 하지 못하는 문제점을 지니고 있음.
  - 이와 같은 문제는 Global Euclidean Distance 기반 판독에서 비롯된 것이며 이를 극복하기 위해 국소적인 Euclidean Distance 기반 Clustering 판독이 요구됨. 이는 국소적인 특징을 중심으로 보는 Manifold Learning 과 유사함.
  - DBSCAN 은 국소적으로 1 개의 소규모 클러스터에 대한 기준(Minimum Points & Epsilon)이 만족되는 국소적으로 여러 개의 클러스터를 생성함. 그리고 각 클러스터가 특정 거리 (Epsilon) 이내에 있다면 하나의 클러스터로 병합시킴.
  - 국소적인 클러스터 생성 후 병합을 연속적으로 함으로서 비원형 분포의 데이터에 대해서 효과적으로 Clustering 할 수 있음.

- **Local Cluster 병합을 통한 Cluster 증식**
  - **Density 판단 기준**
    - Epsilon (eps)
      : 1 개 Cluster 의 최대 지름 / Cluster 간 병합을 위한 거리 기준
    - Minimum Points (minPts)
      : 1 개의 클러스터로 취급되기 위해 Epsilon 이내에 존재해야하는 최소 데이터의 개수

  - **Density 기준으로 데이터 분류**
    - Core Point
      : 자기를 중심으로 eps 내에서 minPts 이상으로 데이터를 보유하고 있는 Data Point
    - Border Point
      : 자기를 중심으로 eps 내에서 minPts 를 만족하지 못하지만 다른 Cluster 에 속하는 Data Point
    - Noise Point (Outlier)
      : 자기를 중심으로 eps 내에서 minPts 를 만족하지 못하고 다른 Cluster 에 속하지 못하는 Data Point
    - Directly Density-Reachable : Core Point 중심의 Cluster 내에 속하는 상황
    - Density-Reachable : Border Point 를 찾아서 Cluster 를 통해 서로 연결되는 상황
    - Density-Connected : Density-Reachable 한 요소끼리 서로 근접하면서 동일한 Cluster 에 속하도록 Cluster 를 병합시킴

  - **Density 기준으로 Cluster 병합**
    - 최초에 각 데이터별로 eps 와 minPts 를 만족하는 경우 국소적인 Cluster 를 생성함. 그리고 주변에서 Density-Reachable 인 모든 Point 를 찾음. 만약 Density-Reachable 한 Point 가 Core Point 인 경우 해당 Point 의 클러스터와 병합함. 만약 Core Point 가 아닌 경우에는 Border Point 이기 때문에 제일

근접한 Cluster 에 배정시킴. 이와 같은 병합 과정을 새로운 데이터를 더 찾을 수 없을 때까지 지속함.

- 전체 데이터 분포에서 Core Point 와 Border Point 를 찾아가면서 병합하고, Noise Point 를 제외함으로서 Clustering 을 수행함.

- **장점**
  - 1 개의 국소적인 Local Cluster 가 구성되기 위해 최소 특정 개수의 데이터가 밀집되어야하기 때문에 Outlier 를 효과적으로 배제할 수 있음. 그리고 Noise 가 있는 상황에서도 효과적으로 Outlier Rejection 과 Clustering 을 수행할 수 있음.
  - 비원형 구조의 복잡한 데이터 클러스터에 대해서도 효과적으로 Clustering 을 수행할 수 있음.

- **한계점**
  - 데이터의 분포가 너무 Sparse 한 경우 대부분의 데이터를 Outlier 로 취급할 수도 있음.
  - 데이터의 밀집이 일정하지 못한 경우 동일한 기준으로 국소적인 Local Cluster 를 못 만듬.
  - 1 개의 Cluster 를 구성하는 주요 기준인 Epsilon 과 Minimum Points 에 따라 Clustering 결과가 크게 달라짐.

- **적절한 국소적인 Cluster 생성 기준 Epsilon 과 minPts 를 어떻게 설정하는가?**
  - 만약 국소적인 Local Cluster 가 Dense 할수록 각 데이터는 자신의 이웃들과의 거리 합이 매우 낮을 것임. 만약 Epsilon 크기가 너무 커서 Noise Point 를 포함하기 시작한다면 자신의 이웃들과의 거리 합이 갑자기 증가할 것임.
  - 특정 Epsilon 과 minPts 로 DBSCAN 을 수행한 후 KNN 을 사용하여 각 Cluster 의 중심에 K 개의 Neighbor 를 선정함. 그리고 Cluster 중심 – Neighbor 데이터에 대한 거리 합을 구함.

  - Neighbor 개수에 따라 Distance 가 갑자기 커지는 순간의 Neighbor 개수 K 와 Distance 값을 최초 eps 와 minPts 로 설정할 수 있음.

5) GMM (Gaussian Mixture Model)

- **주요 원리**
  - **Distance 만으로 판단하기 보다는 Distribution Model 을 사용하여 등고선을 그려서 Soft Clustering 을 수행함**
  - **Multi-Variate Gaussian Model 의 주요 Parameter**
    - **Mean**

      : 데이터 클러스터의 평균

      : 만약 데이터가 클러스터를 이룬다면 평균 중심으로 제일 많이 모일 것이라 생각함

    - **Variance**

      : Mean 을 중심으로 데이터 클러스터의 퍼짐 정도를 표현함

    - **Covariance**

      : 데이터 클러스터내 데이터의 Feature 간의 선형적 연관성 / 방향성 표현

      : Covariance 를 반영하면서 원형 또는 타원형 구조의 분포를 가지게됨

      : 모든 Feature 사이의 Covariance 는 Covariance Matrix 로 표현함

  - **Multi-Variate Normal 의 경우...**
    - Feature X, Y 가 독립이라면 Cov(X, Y) = 0
    - Cov(X, Y) = 0 이라면 X, Y 는 독립 **(※ Multi-Variate 경우에만 성립함)**

  - **Maximum Likelihood Estimation (MLE)를 통해 Gaussian Model 의 평균점에 최대한 많은 데이터가 배치되게 하여 평균점에 데이터 클러스터 요소가 많이 나타나도록 만듬. (Maximize Log Likelihood Function)**

- Probability of observing all points (assume independence)

$$P(X) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- We want to find the parameters $\theta = (\mu, \sigma)$ that maximize the probability $P(X|\theta)$

➢ 만약 데이터 클러스터가 Gaussian Model 을 가진다면 각 데이터가 Gaussian Model 에서 평균점을 중심으로 얼마나 분산되어 퍼져서 배치 확률을 가지는지 구하고, 각 데이터의 확률들을 모두 곱하여 전체 확률로 표현함.
(※ 각 데이터는 독립이라 가정하면 모든 데이터의 발생 확률은 각 데이터의 확률의 곱임)

➢ GMM 의 문제는 **'Gaussian Model Parameter θ (Mean, Variance)가 주어졌을 때 모든 데이터의 발생 확률 P(X)를 최대화 시키는 것'**로 표현될 수 있으며, 이는 **P(X|θ)**라는 조건부 확률로 표현할 수 있음.

- The probability $P(X|\theta)$ as a function of θ is called the Likelihood function

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- It is usually easier to work with the Log-Likelihood function

$$LL(\theta) = -\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} - \frac{1}{2} n \log 2\pi - n \log \sigma$$

➢ P(X|θ)는 **'데이터셋 X 가 주어졌을 때 Model Parameter (θ (Mean, Variance))에 대한 가능도' L(θ|X)**로 표현될 수 있음.

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)}$$

➢ **이 때, P(X|θ)와 L(θ) (= P(θ|X))는 서로 전제 순서가 다르지만 각자 최대화하는 방향은 같음. 왜냐하면 두 확률식은 Bayesian Theorem 으로 Linear 한 관계로 동일하게 취급될 수 있기 때문임. 데이터셋 X 가 주어진 상황에서 Model Parameter θ 에 대한 가능도 P(θ|X)를 최대화하는 것은 Model Parameter θ 를 사용했을 때 모든 데이터 X 의 발생확률 P(X|θ)을 구하는 것과 동일한 방향임.**

$$p(\mathbf{x}) = \sum_{k=1}^{M} \pi_k \mathrm{Normal}(\mathbf{x} \mid \mu_k, \Sigma_k)$$

➢ **Real World 데이터셋은 Unimodal 하지 않기 때문에 단순 Gaussian Model 1 개로 표현할 수 없음. 여러 개의 Gaussian Model 에 가중치를 곱해서 더하여 다중 Gaussian Model 을 하나로 묶어서 여러 개의 Gaussian Model 중심점이 반영되게함.**

- **주요 특성**
  - 다중 Gaussian Model 의 Mean 값과 Variance 로 구성된 Gaussian Mixture 를 통해 데이터가 1 개의 Cluster 에만 소속되는 것이 아니라 여러 Gaussian Model / Cluster 에 부분적으로 소속될 수 있기에 GMM 은 Soft Clustering 기법임.
  - 기본적으로 원형 또는 타원형 구조의 Gaussian Model 을 사용하기 때문에 원형 또는 타원형 구조의 데이터 클러스터를 만들려고함.

- **EM 알고리즘 (Expectation & Maximization)**
  - GMM 에 속하는 여러 Gaussian Model 은 최초에 개별적인 랜덤한 위치 (평균점, Mean)에서 랜덤한 Variance 를 가지고 시작함. 그러나 각 데이터 클러스터의 중심점으로 Gaussian Model 의 평균점을 일치시키고, 데이터 클러스터 범위까지 Variance 를 맞추기 위해서는 탐색 방식이 필요함.
  - EM 알고리즘은 K-Means 와 같이 모델의 중심점 (Centroid / Mean / 평균점)을 소속 데이터의 Mean 으로 이동시키고, Variance 를 조정하면서 Log Likelihood 함수가 최대가 되도록 조정함.
    - **K 의 Gaussian Model 을 랜덤한 위치 (평균점, Mean)와 랜덤한 Variance 로 초기화함**
    - **E-Step**
      - **데이터셋의 모든 데이터는 개별 Gaussian Model 의 중심점에 대한 거리에 비례해서 소속 비율로 Labeling 함**
        (※ 현재 평균과 분산에 따라 가능도 평가)
    - **M-Step**
      - **각 Gaussian Model 은 모든 데이터에 대해 자신의 소속 비율인 Responsibility 가 최대가 되도록 Mean, Variance, Mixing Coefficient 를 업데이트함 (※ 최대 가능도 방향을 향해서 업데이트)**
    - **전체 데이터에 대한 Log Likelihood 가 최대화 되어 수렴될 때까지 E-Step 과 M-Step 을 반복함.**

- **한계점**
  - GMM 은 기본적으로 원형 또는 타원형 구조를 가진 Gaussian Model 을 기반으로 하기 때문에 비원형 데이터 분포에 대해서는 효과적으로 Clustering 을 수행할 수 없음.
  - EM 알고리즘에서 초기 Gaussian Model 의 초기화 위치와 분포에 따라서 Clustering 결과가 달라짐. 초기 위치에 따라 서로 떨어진 원형 데이터 클러스터를 하나의 클러스터로 묶으려고 할 수도 있음.
  - EM 알고리즘에서 사용하는 Gaussian Model 의 개수에 따라 Clustering 결과가 달라질 수 있음. 데이터 클러스터 개수와 Gaussian Model 의 개수가 일치하지 않으면 1:1 로 Clustering 되지 않을 수 있음.

6) **Hierarchical Clustering (거리 기반 Clustering)**

- **Hierarchical Clustering 의 장점**
  - Cluster 의 개수를 사전에 미리 정할 필요없이 자동으로 Cluster 가 병합 / 증식함
  - 구조적 의미 / 분류적 의미 (Taxonomy)를 도출할 수 있음
  - Dendrogram 을 통해 Cluster 병합 / 분리를 표현할 수 있음

- **Hierarchical Clustering 한계점**
  - 연산 소요량이 높음
  - Cluster 사이 병합이 이루어지면 다시 풀 수 없음 (Feedback 불가)
  - Objective Function 을 최적화 하는 구조가 아님
  - Noise / Outlier 에 대해 민감하며, 다양한 크기 / 모양의 클러스터를 처리하기 힘듬

- **Agglomerative Clustering**
  - **주요 원리**
    - 최초에 각 데이터의 한 점을 개별 Cluster 가 취급하여 시작하며, 주변 데이터와의 근접함으로 Proximity Matrix 로 계산하여 표현함.
    - 각 데이터는 제일 가까운 데이터들과 병합하면서 Pair 를 이루고, 각 Pair 는 주변에 가까운 데이터 또는 Pair 를 병합하면서 증식함. 병합할수록 Proximity Matrix 를 업데이트함
    - Cluster 의 병합과 증식은 데이터가 1 개만 남거나 목표 Cluster 개수에 도달했을 때 정지함.
  - **주요 특성**
    - 데이터 간의 근접도 Proximity 를 계산하는 방법에 따라 결과가 달라짐

- **Divisive Clustering**
  - **주요 원리**
    - **최초에 큰 데이터 Cluster 덩어리로 시작해서 데이터 1 개만 남거나 목표 Cluster 개수에 도달할 때까지 큰 데이터 Cluster 덩어리를 연속적으로 점점 쪼개나감**

- **Inter-Cluster Similarity (Hierarchical Clustering 에서 Cluster 간 근접함 / 병합 기준)**
  - **Min : 거리가 짧은 순서대로 Pair 또는 Cluster 를 이루어나감**
    : 비원형 구조에 대해 Clustering 할 수 있는 장점이 있음
    : Noise 와 Outlier 에 취약함
  - **Max**
    : Noise 와 Outlier 에 비교적 덜 취약함
    : 큰 데이터 클러스터를 하나로 이루지 못하고 분리되는 경향이 있음
    : 더 큰 Cluster 에 치우쳐져 구형성 모양으로 Biased 됨
  - **Group Average**
  - **Distance between Centroids**
  - **Objective Function-based Approach (Ward's Method – Squared Error)**

5. Feature Engineering : Feature 를 어떤 구조로 표현할 것인가?

1) Types of Features

- 데이터셋 Feature 의 종류에 따라 적합한 데이터 분석 방식과 모델을 도입해야함
  - Binary 가 아닌 Categorical 데이터에 대해서는 Logistics Regression 을 적용할 수 없음
  - String 형태로 표현되는 데이터는 직접적으로 Machine Learning 기법을 사용할 수 없음
- 데이터셋 Feature 의 종류에 따라 적합한 표현방식을 사용해야함
- 데이터셋 Feature 의 종류에 따른 단위를 반영해서 Scaling 을 해줘야함
- **Continuous Features** : **연속적인 값으로 구성된 Feature**
- **Categorical Features (Discrete Features)**
  - **순서형 : Label 별 상하관계가 존재함 (중간값이 존재함)**
  - **명목형 : Label 별 상하관계가 존재하지 않음 (중간값이 없음)**

2) Feature Encoding for Categorical Features : 숫자로 어떻게 Feature 를 표현할 것인가?

- **One-Hot Encoding (Dummy Variables)**

workclass	Government Employee	Private Employee	Self Employed	Self Employed Incorporated
Government Employee	1	0	0	0
Private Employee	0	1	0	0
Self Employed	0	0	1	0
Self Employed Incorporated	0	0	0	1

  - Label 의 모든 종류를 모아서 1 x M 구조로 재구성함.
  - 1 x M Feature Vector 에서 각 데이터는 자신이 해당하는 Label 의 Index 에 대해서만 1 을 배정하고 나머지는 0 을 배정한 형태로 Label 을 다시 부여받음.
  - 변형된 데이터를 Label 로 별도로 만들어서 원래 데이터 대신 ML 모델 학습에 들어감.

- **Numeric Labels for Categories**
  - 문자열 Label 에 대해서 개별적으로 특수한 숫자를 Mapping 하여 Category Label 을 만들 수 있음 (숫자는 항상 연속적인 표현이 아닐 수 있음)
  - 여러 Char 데이터로 구성된 문자열보다 숫자 몇 개로 구성된 구조가 데이터 소모가 적음
  - 어떻게 데이터를 재표현하는가에 따라 통계적 표현이 가능하게 할 수 있음

- **Binning (Discretization)**

Comparing linear regression and decision tree regression

Comparing linear regression and decision tree regression on binned features

- As features are constant within each bin, any model must predict the same value for all points within a bin.

➢ 1 개의 Feature 를 여러 구간으로 나눠서 여러 Feature 가 나오게함. 이를 통해 데이터셋에 Non-Linear 한 특성을 부여할 수 있음.

➢ 전체 입력 Feature 에 대해서 Linear Model 과 Decision Tree 는 서로 완전 다른 방식으로 Decision Boundary / Regression Line 을 그리면서 다른 방식으로 작동함.

➢ 입력 Feature 를 구간별로 나눠서 개별적으로 Linear Model 과 Decision Tree 를 적용하면 서로 유사하게 작동함. 이를 통해 Linear Model 이 보다 Non-Linear 하게 작동할 수 있도록 만들 수 있음.

➢ Feature 의 표현 종류에 따라 기존 모델의 특성이 개선될 수 있음.

➢ **Interactions and Polynomials**

- **Binning 을 적용한 후 추가적으로 각 Bin 별로 기울기와 Offset 의 새로운 Feature 로서 제공함으로서 Feature 의 종류를 증가시키고 모델 학습을 개선시킬 수 있음.**

Linear regression with a separate slope per bin

- **Linear Model 에 Polynomial Feature 또는 Interaction Feature 를 추가해서 Non-Linearity 특성을 더 부여할 수 있음.**

Linear regression with tenth-degree polynomial features

Comparison of different gamma parameters for an SVM with RBF kernel

- **Linear Model 에 Polynomial Feature 를 부여할 경우 Kernel SVM 과 유사하게 작동함.**
- **항상 복잡한 모델을 사용하는 것 보다 간단한 모델을 개선해서 복잡한 기능을 유사하게 작동시킬 수도 있음.**

- **Univariate Non-Linear Transformation**
  - 대부분의 ML 모델은 데이터의 분포가 Gaussian 이라 생각 또는 가정함
  - 그러나 데이터셋의 표현에 따라 Gaussian 으로 나타나지 않을 수 있음. **(예 : 빈도 수 데이터)**
  - Gaussian 분포가 아닌 데이터에 대해 ML 모델이 제대로 작동하지 않을 수 있기 때문에 데이터셋을 Gaussian 분포로 바꾸는 작업이 필요함.

Histogram of feature values
Histogram of feature values after logarithmic transformation

  - 특수한 Non-Linear Transformation 연산을 통해 데이터셋을 Gaussian 으로 변형하여 적용하지 못하던 모델을 사용할 수 있게됨.
  - 대표적으로 Tree 기반 모델은 데이터셋의 Gaussian 여부를 따지지 않지만 Linear Model 의 경우 데이터셋 분포가 Gaussian 이어야 제대로 작동함. 특수한 Non-Linear Transformation 을 적용하여 데이터셋 분포를 Gaussian 으로 변형하면 Linear Model 을 적용할 수 있게됨. 이를 통해 성능 개선으로 이어질 수 있음.

3) Feature Selection

- **Automatic Feature Selection : 어떻게 하면 핵심 Feature 만 뽑아낼 수 있는가?**
  - **Univariate Statistics**
    - Analysis of Variance (ANOVA)를 통해 3 개 이상의 평균이 같은지 확인하여 평균의 차가 존재하는지 판단함. 그리고 P-Value 가 높은 Feature 를 모두 제거함.
      (※ P-Value 가 높을수록 ML 을 통해 얻으려는 Target 과 관련되어있음)
      (※ P-Value 가 희박 할수록 귀무가설을 기각할 증거가 됨)
    - Noise Feature 를 제거함으로서 성능 개선을 도모할 수 있음.
    - Feature 가 너무 많은 경우 / Feature 중 무의미한 경우가 많은 경우 ANOVA 와 같은 Univariate Statistics 를 적용함

  - **Model-based Selection**
    - 별도의 Supervised Machine Learning 모델을 사용하여 핵심 Feature 를 선정함.
    - Feature Selection 에 사용하는 모델은 학습시키려는 모델과 같은 필요는 없음.
    - Feature Selection Model 은 각 Feature 에 대한 중요도를 수치로 제공함.
    - Decision Tree / Tree 기반 모델은 Feature Importance Attribute 라는 중요도 수치를 제공함. Linear Model 은 각 Feature 에 대한 Coefficient 가 중요도 수치로 사용할 수 있음.

- **Iterative Selection**
  - 여러 개의 Feature 를 모두 사용하지 않고 일부만 사용하여 여러 버전의 ML 모델을 만듬. 목표하는 성능에 도달할 때까지 Feature 를 추가하면서 여러 버전의 모델을 테스트함.
  - 연속적인 여러 버전의 모델 설계와 테스트를 통해 최적의 모델을 찾아냄.
  - 여러 버전의 모델을 계속 만들고 테스트 해야함으로 연산 소요량이 매우 높음.

- **Utilizing Expert Knowledge**
  - 처리하려는 데이터셋의 Feature 에 대해 사전 전문 지식을 가지고 있다면 보다 손쉽게 핵심 Feature 를 뽑아낼 수도 있음.