@ Data Mining Methodology

# Statistics vs Data-Mining

* Statistic : 현실 재현에 부적합함(가정을 전제로 함), 알고리즘이 선형적인 구조
* DM : 현실의 noise를 반영함. 알고리즘이 비선형적이고 Robust(low 가정의 제약)
* DM의 기법들의 구분 기준 : Data의 ‘size, pattern-type, noise’, 해당 기법의 기본 가정을 data가 만족하는지, ..

# Supervised Learning

* X(feature)->Y(label)의 training data가 주어졌을 때, test-data(미래)를 예측함
* Regression : data type = Int (ex. 24.5, 31.2)
* Classification : data type = Class (ex. A,B,C)

# Unsupervised Learning

* Y(label)의 Training data 없이 미래를 예측함
* Dimension Reduction, Clustering, Recommendation 등의 기법이 있음

@ Data Mining Process

# Process : Biz 목적 설정(사용자,일회성) > data 수집 > data 탐색(변수 설정) -> DM 문제 설정(회귀,분류,군집..) > Data division(학습,검증,평가 용도로 구분) > Select DM method(Decision Tree, Logistic Regression, ..) > Data Analysis(Base model, Model tuning)

> 결과 해석 > 모델 적용

# Variable’s type : Numeric(수치형,int), Categorical(범주형,class)

* Categorical : Nominal(no-rank / ex. M/F), Ordinal(rank / ex.A,B,C
* One-hot encoding : Class는 통계적 활용이 어려워서, R에서는 ‘Dummy Variable’을 생성해 Numeric으로 바꾸는 과정

# R studio tips

* Index를 찾을 때, c(n,m,l)을 통해서 column의 Index 구간을 설정할 수 있음
* DF.TOTAL.VALUE[] : 해당 구간의 value 모두 보여줌
* dim()을 통해 구한 값은 length(DF%TOTAL.VALUE) + length(DF) 과 같다.
* summary(DF) : 각 row의 간단한 통계 지표 확인
* $ : DF에서 Var/Index에 접근W할 때 사용됨. Vector에서는 Indexing을 이용함.
* DF Index에서 -c(1:4) : 1~4 Column을 제외한 Index를 조회함
* na.omit() : omit=생략하다 > Remove NA Value
* setdiff(A, B) : A,B의 차집합

# Outlier(이상치) 처리 : Novelty detection

* 극단적인 Value. 데이터 전처리에서 가장 중요한 작업은 이상치를 제거하는 것

# NA(결측치) 처리

* Removal : 적은 수의 record가 NA인 경우, Column을 제거함
* Imputation(대체) : Entity의 많은 곳에 NA가 있는 경우, 적절한 값(mean,midian)으로
* R) sample(rownames(DF), N) : row에서 N개를 Sample로 추출

# Normalizing(Standardizing) and Rescaling Data

* Var의 단위가 차이가 너무 큰 경우, 통계량 측정이 어려우므로, 모든 Var을 같은 척도(Scale)에 넣는 것. (ex. 10^20 vs 10)
* Standard scaling(z-score) : (x-mean)/SV
* Min-Max scaling : (x-min)/(max-min). 0~1 사이 Value로 변환

# Overfitting & Predictive Power (과대적합 문제)

* 개발된 Model이 old-data는 ‘매우 잘’ 설명하지만, new-data는 설명하지 못하는 문제
* predictor var이 너무 많아서 발생함. Overfitting 할수록 분류 선이 Straight->Curved로 변한다. ( <> Under-fitting)
* Sol) Data partitioning > Validation data을 통해서 여러 모델의 성능을 비교 > Hyperparameter 조정(ex. Decision Tree, kNN) > 최종 평가

# Cross validation (교차 검증)

* data 양이 적을 경우, 전체를 k(직접 설정)개로 분할한 뒤, k-1(자신 제외) 개의 Train-data와 1개의 Valid-data(자신)을 활용한 평가치의 평균을 성능 평가에 활용

# 인식률 개선

* Classification(분류)를 위해서는 Feature을 수치화하고, 이를 비교한다.
* 소수의 Feature로는 accuracy가 떨어지는 경우가 많아서, 여러 개의 Feature이 결합
* 이 결합된 Feature은 Vector 형태로 Array로 저장된다.

# Accuracy vs Cost

* 만약 분류 작업을 하는 과정에서 잃는 Cost가 크다면, 때로는 하지 않는 것이 더 효율적이다. (lower accuracy)
* Overfitting으로 과한 curved line이 형성되면, 미래에는 Accuracy가 떨어질 수도 있다.

@ Similarity Measure

# Similarity Measure for Numerical data

* groups of observation의 Similarity를 측정함. 평가 지표는 ‘거리’
* 거리 측정 방법 : Euclidean(피타고라스), Manhattan(격자로 위치 구분)
* Outlier(이상치)가 많다면 Manhattan을 이용한다.
* Ex. k-Nearest Neighbors, Cluster Analysis

# Data Scaling : Standardizing / Normalizing

* Data Scale의 차이가 심한 경우, 거리 측정이 어려우므로, 표준화 해준다.
* Standardize, Min-max normalization이 있음. 이전 회차 강의 참고

# Matching coefficient : Similarity measure for categorical data

* N(Var with matching outcomes) / N(total Var)
* 값은 0~1 사이의 척도. 1이 완벽한 match라고 할 수 있음
* Problem) Negative Value가 같은 경우에 Misleading outcome 발생함. 또한, 두 data set의 크기가 다른 경우에도 발생할 수 있음.

# Jaccard’s coefficient : Similarity measure for categorical data

* N(Var(positive outcome)) / N(total Var) - N(Var(negative outcome))

@ Dimension reduction

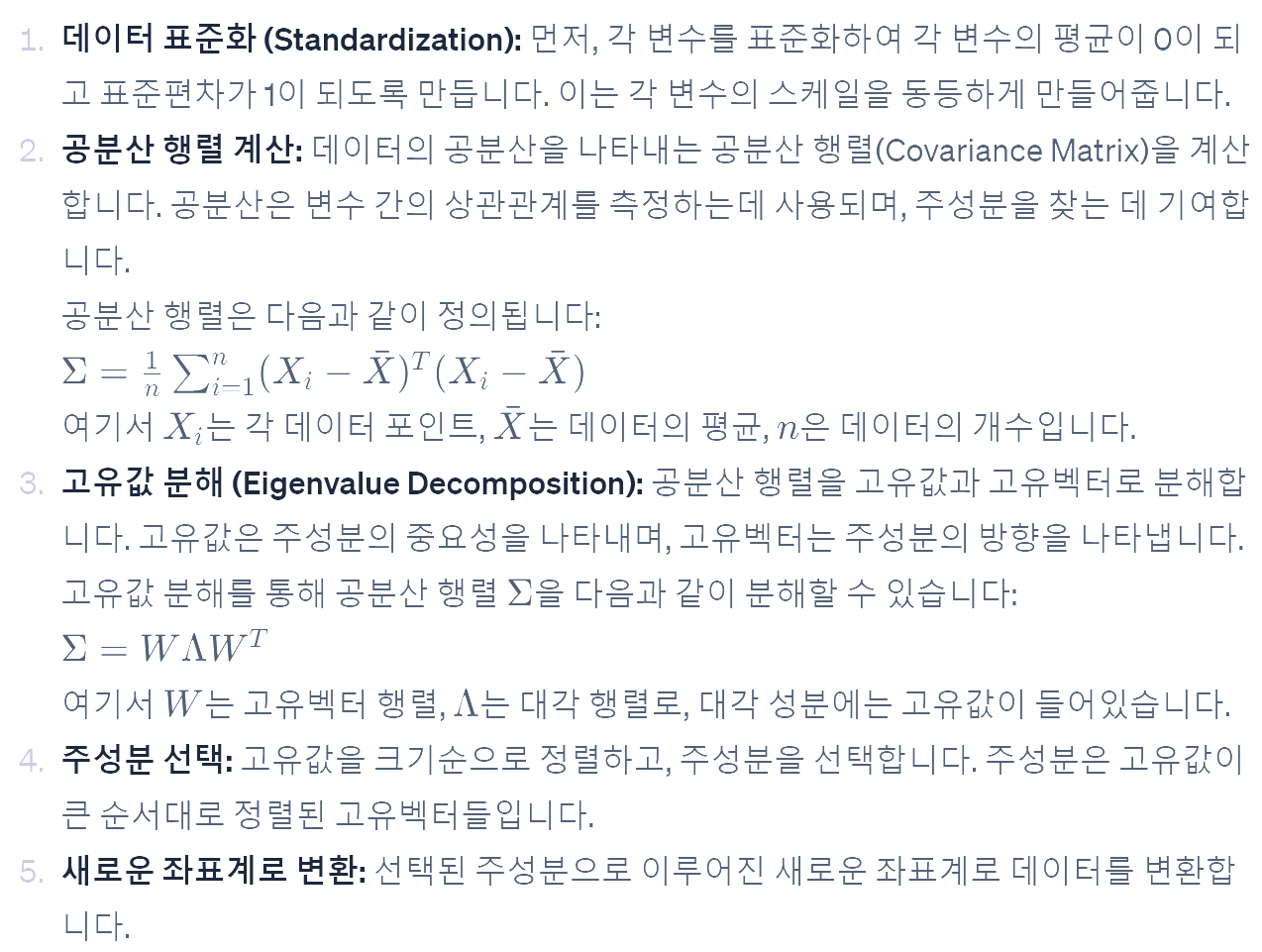
# Dimension Reduction

* N(Var=X)이 큰 Table은 모든 Var을 반영하는 것이 오히려 정확도와 신뢰도를 약화시키는 Overfitting 현상이 발생함
* 불필요한 Var(feature)을 제거함에 따른 Accuracy의 희생을 최소화하면서, 비용을 감소시키는 방법을 찾는 과정

# Curse of high dimensionality

* 차원이 하나씩 늘어날 때마다, 공간의 크기가 기하급수적으로 증가함 > Data becomes sparse(영공간이 증가해서, 데이터를 찾기 어려움) > Distance 증가 > Pattern 찾기 어려움 (lower Model’s accuracy)
* Problem of feature Selection : Selected Var의 해석 쉽지만, un-selected Var과의 Correlation 찾기 어려움 (sol. Manifold Learning)

# PCA (Principal Component Analysis, 주성분 분석)

* ‘Projection’을 통해서 원본 data의 Covariance를 최대한 잘 설명하는 low dimensional space로 차원 축소를 수행함.
* Eigenvalue Decomposition을 통해서 Eigenvalue을 구하는 것이 핵심
* 이 Eigenvalue로 만들어진 새로운 축들은 원본의 정보를 담고 있으므로, 축소된 차원 내에서 원본의 데이터 손실이 적음.
* Process) Standardization > Covariance Matrix > Eigenvalue(Vector) > Sort Eigenvalue > **N**개의 EigenValue 선택 > 새로운 축소된 차원의 Matrix 생성 > 원본 dataset을 새로운 Matrix에 Projection
* 선형 data의 차원 축소에 효과적. 비선형적이면 Manifold Learning을 이용함

@ Performance Evaluation

# Regression Model’s Valuation

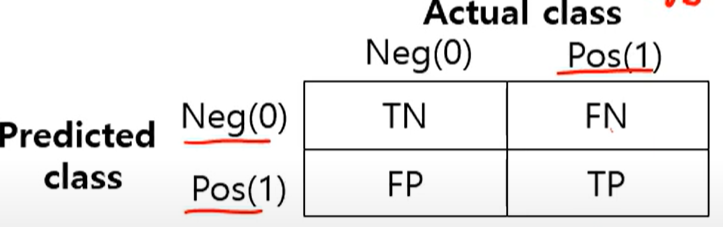
* Supervised Learning은 Y(출력변수) 예측을 목표로 함을 상기하자.
* Naive(순진한) BenchMark : 모델을 사용하지 않고, 단순히 Y의 Average를 통해 예측
* **RSS**(잔차 제곱 합) : 모든 데이터 포인트에서 예측값과 실제 값 사이의 차이를 제곱하여 합한 것

# Classification Model’s Valuation

* Binary Classification : 0(Negative/False) - 1(Pos/True)
* Measure : Accuracy(얼마나 Model이 분류를 잘 했는가)
* Actual+Predicted class : data를 옳게 분류했는지 정리한 표

Ex. FP(False+Positive) : False(0)으로 분류해야하는 것을 Positive(1)로 분류했다. 따라서 잘못 분류한 것

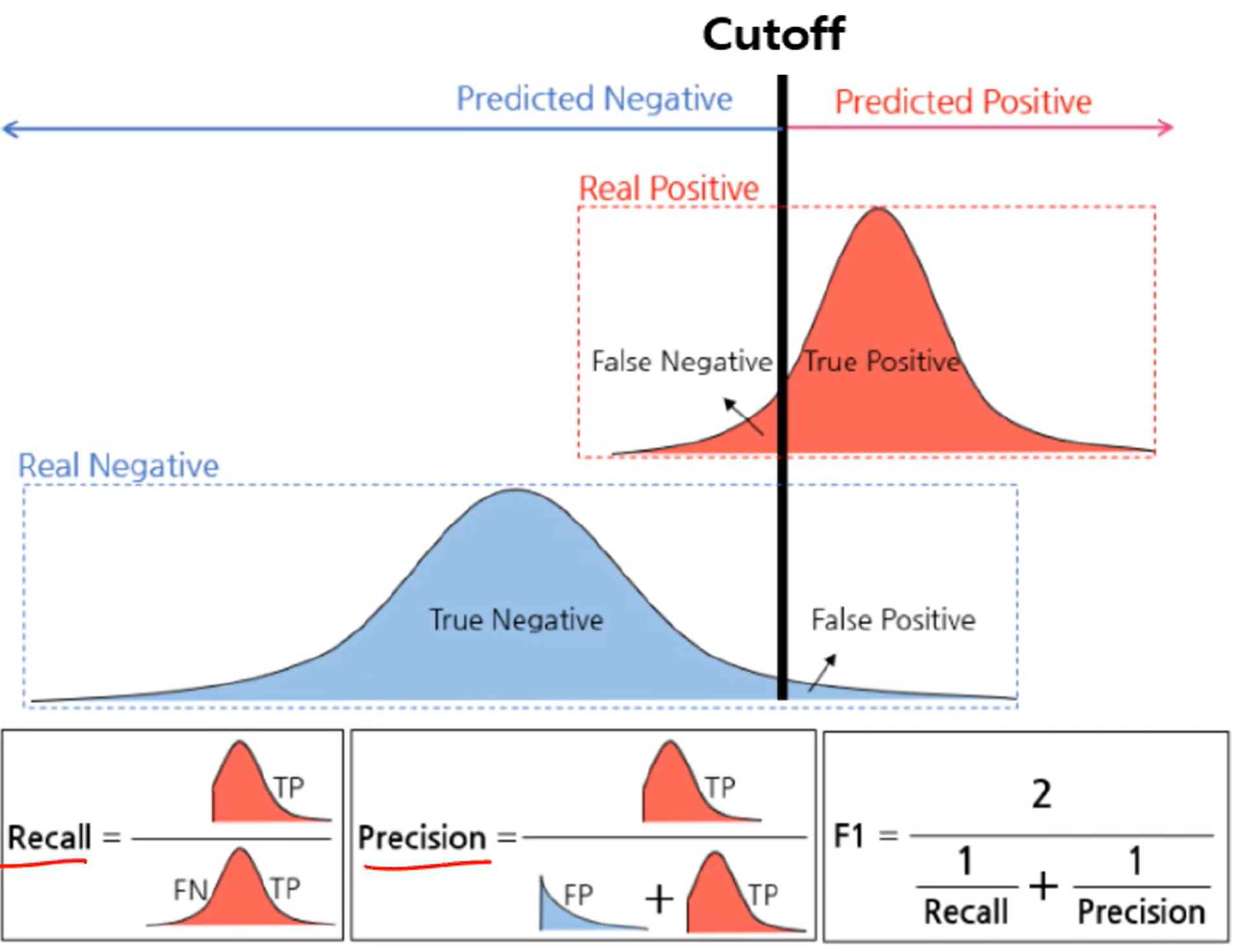
* Accuracy = (TN+TP) / (TN+TF+FP+FN)



# Propensity & Cutoff

* Propensity(경향) : 해당 관측치가 특정 class에 속할 확률
* 특정 Class에 대한 Propensity가 Cutoff(기준선)보다 크면, 해당 class로 분류한다.
* Confusion-Matrix : Class가 잘못 분류되는, 오분류율을 정리한 Matrix
* Class의 중요성이 균등한 경우에만, 오분류율이 낮을수록 좋은 Model

# Class의 중요성이 불균등한 경우의 성능 평가

* Precision(TPR) = TP / (FP+TP) : Neg인 것을 Pos로 잘못 판단하면 더 큰 문제.

Ex. 정상-mail을 Spam-mail로 잘못 판단하는 것이 역보다 더 큰 문제.

* Recall(FPR) = TP / (FN+TP) : Pos를 Neg로 잘못 판단하는 것이 더 큰 문제 발생.

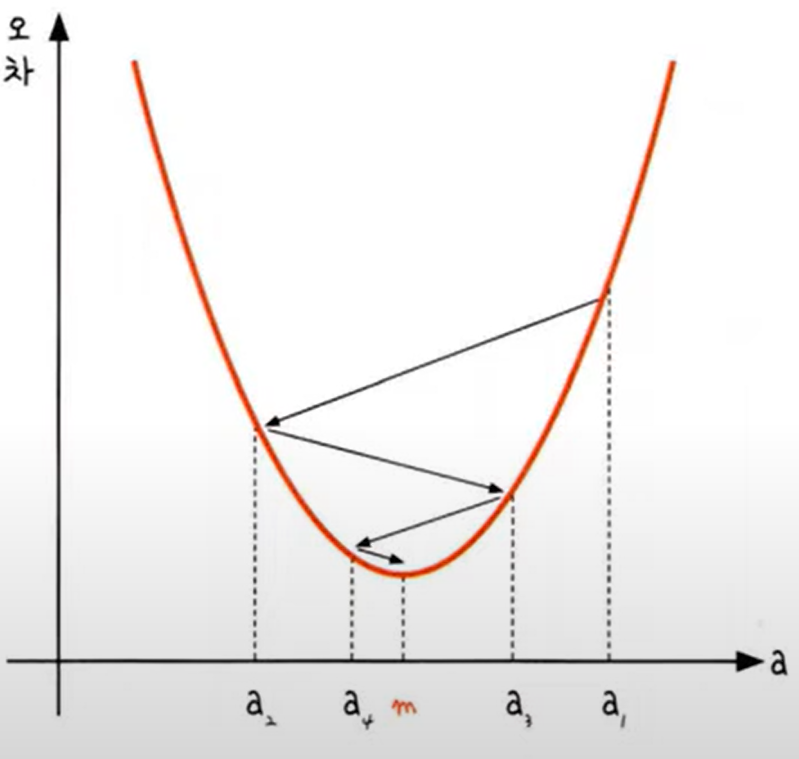
Ex. 암 진단 : T를 F로 판단하는 것이 역보다 더 큰 문제.

* F1 : Precision, Recall을 균등하게 반영함 (+ROC Curve)

@ Multiple Linear Regression

# Linear Regression

* 목표 : 새로운 관측치에 대한 정확한 prediction
* Find Linear relationship between variables (y=ax+b)
* MLP는 Variable 갯수만 증가한 것일 뿐, 관계성을 찾는 것이 목표임은 같다.
* Process : 일단 아무 값이나 넣는다 > ‘오차’를 구한다 > 값을 수정한다

# Gradient Descent (경사 하강법)

* 목표 지점(m)에 대한 방향으로 기울기를 하강시키면서 m에 접근함
* gradient가 감소할수록, 하강하는 거리도 줄어든다.
* Learning Rate(학습률, DL) : 어느 정도의 weight로 접근하는지. 점의 이동 거리가 클수록 학습률도 크다.

# Variable Selection in Linear Regression

* 자료 수집 비용 때문에, MLP에서는 모든 Variable을 분석하기 어려움
* Feedforward method(전진 선택법) : 첫 변수부터 하나씩 추가하며 성능을 비교. 무식하지만 확실함
* Backward Elimination(후진 소거법) : 전체 변수에서 하나씩 변수를 제거해가며 성능 지표를 비교함
* Stepwise(단계적, Selection/Elimination) :전체 변수 집합에서 시작해, 각 단계에서 변수를 추가/제거 함. 변수들이 추가/제거될 때마다 기존 변수도 추가/제거될 수 있음

@ KNN (K-Nearest Neighbors)

* New data가 입력되었을 때, K개의 가장 가까운 기존 data들의 Class를 통해서, class의 경계를 정함
* Instance-based Learning : 각각의 Instance(관측치)를 이용하여 New data에 대한 Prediction 진행
* Memory-based Learning : 모든 학습된 data를 memory에 저장하고, 이를 바탕으로 Prediction
* Lazy Learning : test data가 입력되어야만 작동하기 시작하는 게으른 알고리즘

# Hyper-parameter for KNN

* K : 몇 개의 인접한 data까지 탐색할 것인가
* Distance Measures : data 사이에 거리를 어떻게 측정할 것인가. Normalization을 필요로 하며, 단위의 차이로 생기는 왜곡 때문에 Scaling도 필요함.
* Overfitting : K가 매우 작을 때, 관계성을 파악하기 어려워서 과한 곡선이 형성
* Underfitting : K가 매우 클 때, 너무 완만한 곡선이 형성됨

@ Naive Bayes

# 통계학 용어 복습

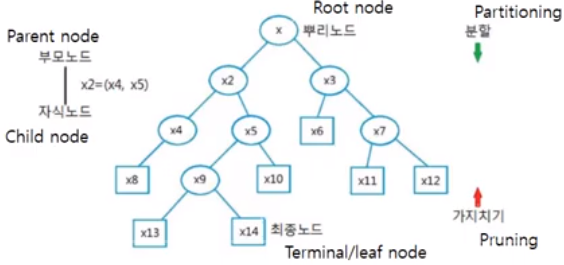
* trial(독립 시행)
* Intersection of events(곱사건, 교집합)
* Difference of events(차사건,차집합) : A - (A and B)
* Mutually exclusive events(배반) : 공통 원소가 없음
* P(Classical, 고전적) : P(A) = N(A) / N(s(all))
* P(Axiomatic, 공리론적) : 확률이 존재하면 0~1 범위임을 전제로 함. 차집합, 합집합 연산 대부분이 성립함
* P(Conditional, 조건부)

# Bayes Theorem

* Subset(A1, A2, A3)이 표본공간 S를 분할(S=A1+A2+A3)한다고 하자. 임의의 사건 B에 대하여, P(B)=P(A1andB)+P(A2andB)+P(A3andB) 라고 할 수 있다.
* Goal) 이때, B가 발생한다는 조건 아래에서, 조건부 확률 P(Ai|B)를 구하는 것. Ai가 B의 원인을 제공하면 사전확률, B 발생 이후의 확률이면 사후확률 이라고 한다.
* 즉, 과거의 불확실성을 바탕으로 미래를 예측하는 것을 목표로 한다.
* 다양한 조건들(A1,A2,..)를 통해서 B의 조건부 확률을 추정함
* 장점 : model이 단순하고, 계산이 효율적이며 분류 성과가 좋다.
* 단점 : 좋은 성과를 얻기 위해서는 많은 수의 관측치가 요구된다. 만약 원하는 값이 관측치에 없다면 성과를 얻을 수 없는데, 이를 Laplace smoothing로 해결한다.

@ Decision Tree

# Intro to Decision Tree

* 의사결정 규칙을 나무 구조로 만들어서 소집단으로 분류/회귀를 수행하는 분석 기법
* 스무고개와 구조가 비슷하며, 구분하기 위한 최적의 질문(기준)을 찾는 것이 중요함
* Classification Tree : Purity(순도, 동질성)가 높을 수록, group에서 이질적인 value가 적다. 즉, 비슷한 Class의 data-set이 형성되므로, 결정 경계가 뚜렷해지므로, 모델의 정확성이 향상된다. (Impurity(Im)을 측정해서 성능 평가)
* Regression Tree : RSS를 최소화(실제 값과 가장 가깝게 예측) 하는 것이 목표.

# Pros and cons of Decision Tree

* 모든 방법을 시도해보기 때문에, Greedy Algorithm의 특징을 닮았다.
* +) Feature type에 대한 제약이 적다. 따라서 계량, 비계량, 혼합 특징을 모두 가능함.
* +) Feature Normalization 과정이 불필요한, nonparametric(비모수적)인 방법이다.
* +) 분류 결과가 ‘해석 가능’하며, 인식 작업이 매우 빠르다.
* P) 2개의 가지는 모든 feature을 반영하지 못해 Instability함.
* S) Random forest Algorithm: Bagging, Random Subspace를 통해서 Ensemble함.
* 추후에 AdaBoost, GBM, XGBoost, LightGBM, Stacking 등 다양한 기법들을 배움