www.hira.or.kr

파이썬을 활용한 데이터·AI 분석 사례

건강보험심사평가원
HEALTH INSURANCE REVIEW & ASSESSMENT SERVICE

CONTENTS

데이터 분석과 통계기초

1. 빅데이터란?(What is Big Data?)
2. 데이터 분석 필수지식(Data Analysis Knowledge)
3. 데이터 분석 모델평가(Data Analysis Model Assessment)
4. 데이터 분석 도구 소개 : python

I. 데이터 분석과 통계기초

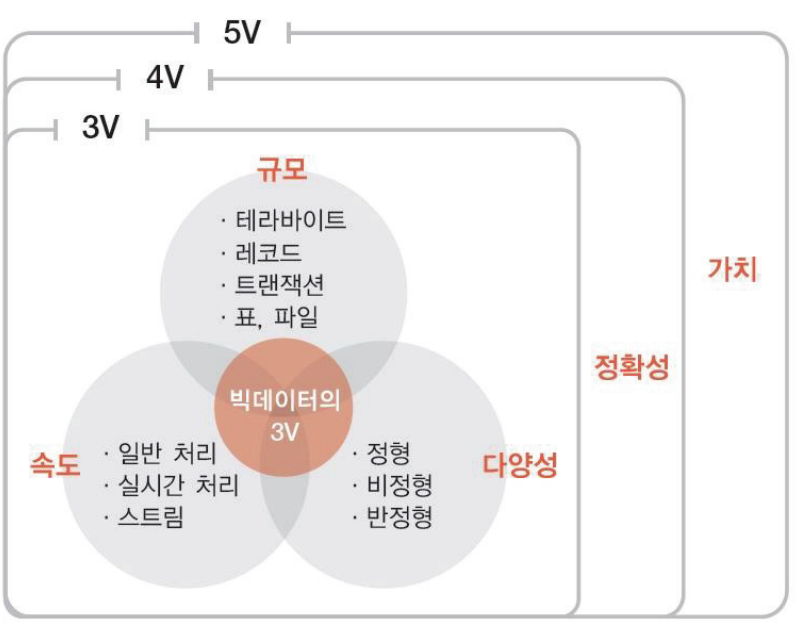
1 빅데이터란?(What is Big Data?)

빅데이터의 정의

- 기존 데이터베이스 관리도구의 데이터 수집, 저장, 관리, 분석 역량을 넘어서는 대량의 정형 또는 비정형 데이터 세트 및 이러한 데이터로부터 가치를 추출하고 결과를 분석하는 기술(위키피디아)
- 대용량 데이터를 활용, 분석하여 가치 있는 정보를 추출하고 생성된 지식을 바탕으로 능동적으로 대응하거나 변화를 예측하기 위한 정보화 기술(국가전략위원회)
- 기존의 관리 및 분석 체계로는 감당할 수 없을 정도의 거대한 데이터 집합으로 이러한 대규모 데이터와 관계된 기술 및 도구(수집, 저장, 검색, 공유, 분석, 시각화 등)를 모두 포함하는 개념(삼성경제연구소)

가트너에서 정의한 빅데이터 5V

기본 3V(Volume, Velocity, Variety) + Veracity(정확성), Value(가치)



빅데이터 범주의 변화

기존 방식으로는 얻을 수 없는 통찰 및 가치 창출과 사업 방식, 시장, 사회, 정부 등에서 변화와 혁신 주도

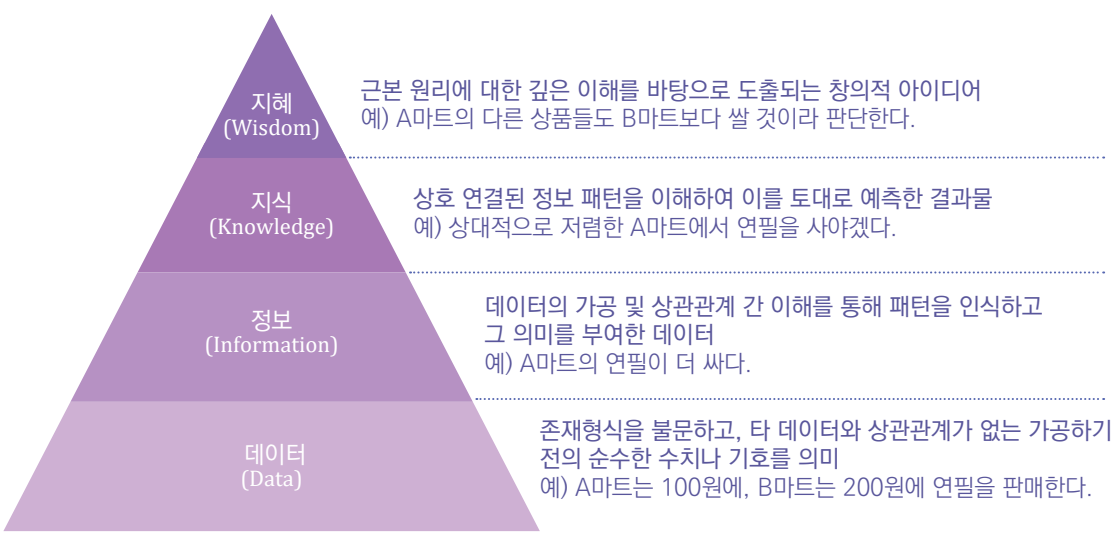
데이터 변화	기술 변화	인재, 조직 변화
• 규모(Volume) • 형태(Variety) • 속도(Velocity)	• 데이터 처리, 저장, 분석 기술 및 아키텍처 • 클라우드 컴퓨팅 활용	• Data Scientist 같은 새로운 인재 필요 • 데이터 중심 조직

데이터 vs 정보 (Data vs Information)

데이터는 가공되지 않은 일련의 사실들을 의미한다.
정보는 의미 있고 유용한 형태로 가공된 데이터를 의미한다.
데이터를 가치 있는 정보로 가공하기 위해 데이터마이닝이 핵심적인 역할을 수행할 수 있다.

DIKW 피라미드

데이터 ▶ 정보 ▶ 지식을 통해 최종적으로 지혜를 얻어내는 과정을 계층구조로 설명한 것이다.

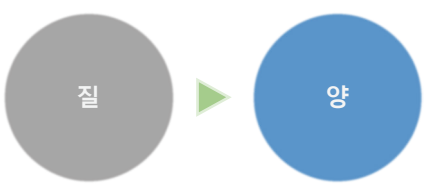


빅데이터가 만들어 내는 본질적인 변화

과거에서 현재로의 변화

필요한 정보만 수집하고 필요하지 않은 정보를 버리는 시스템에서 가능한 많은 데이터를 모으고 그 데이터를 다양한 방식으로 조합해 숨은 정보를 찾아낸다.

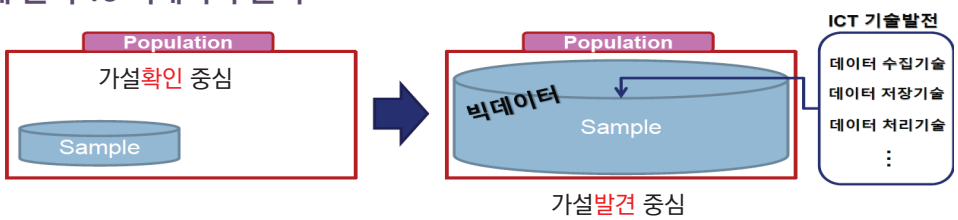
데이터 수집 비용의 감소와 클라우드 컴퓨팅 기술의 발전으로 데이터 처리비용이 감소하게 되었다. 이로 인해 표본을 조사하는 기존의 지식 발견 방식에서 전수조사를 통해 샘플링이 주지 못하는 패턴이나 정보를 발견하는 방식으로 데이터 활용 방법이 변화되었다.



데이터가 지속적으로 추가될 경우 양질의 정보가 오류 정보보다 많아 전체적으로 좋은 결과 산출에 긍정적인 영향을 미친다는 추론에 바탕을 둔 변화가 나타나고 있다.

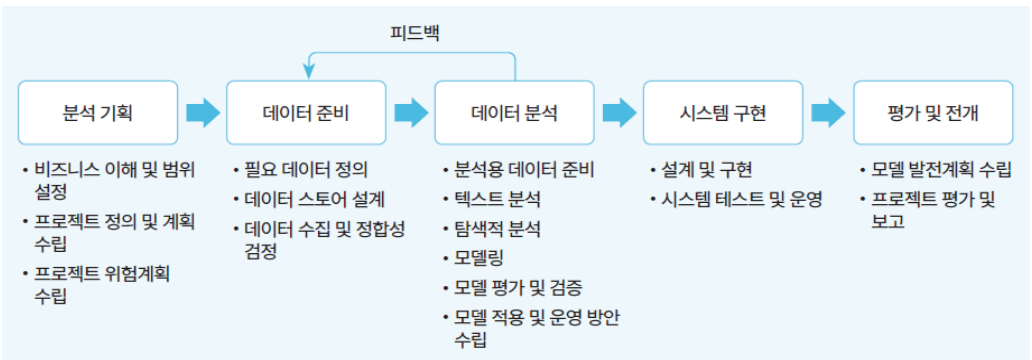
상관관계를 통해 특정 현상의 발생 가능성이 포착되고, 그에 상응하는 행동을 하도록 추천되는 일이 점점 늘어나고 있다. 이처럼 데이터 기반의 상관관계 분석이 주는 인사이트가 인과관계에 의한 미래 예측을 점점 더 압도해 가는 시대가 도래하게 될 것으로 전망된다.

통계 분석 vs 빅데이터 분석

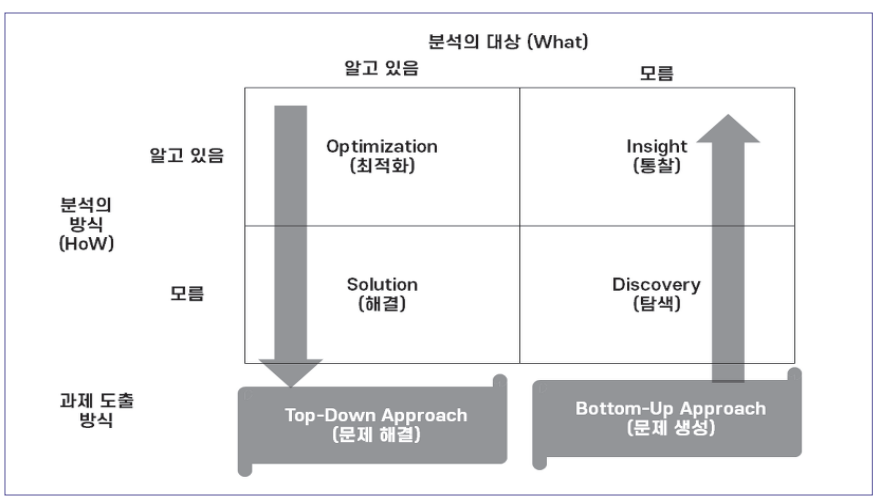


가설발견 중심

빅데이터 분석 5단계

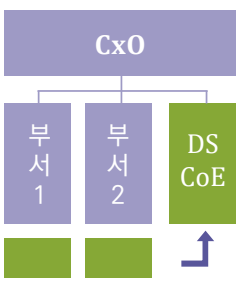
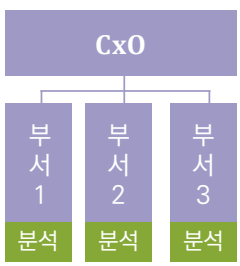
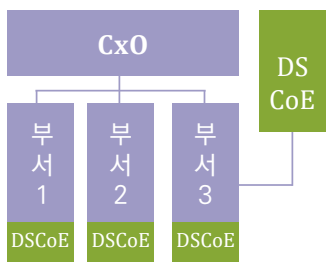


분석주제의 4가지 유형



분석은 분석의 대상(What)과 분석의 방법(How)에 따라서 4가지로 나누어진다.
분석의 대상을 모르고 분석 방식을 알고 있는 경우 Insight 유형이다.
Top-Down 접근법(하향식) : 분석 과제가 주어지고 이에 대한 해법을 찾기 위하여 각 과정이 체계적으로 단계화되어 수행하는 방식이다. 현황분석을 통해 기회나 문제를 탐색하고, 문제 정의, 해결방안 탐색, 타당성 검토를 거쳐 분석 과제를 도출한다.
Bottom-Up 접근법(상향식) : 문제의 정의 자체가 어려운 경우 데이터를 기반으로 문제를 지속적으로 개선하는 방식이다. 비지도 학습 방법으로 데이터 분석의 목적이 명확히 정의된 형태의 특정 필드의 값을 구하는 것이 아니라 데이터 자체의 결합, 연관성, 유사성 등을 중심으로 데이터의 상태를 표현하는 것이다.

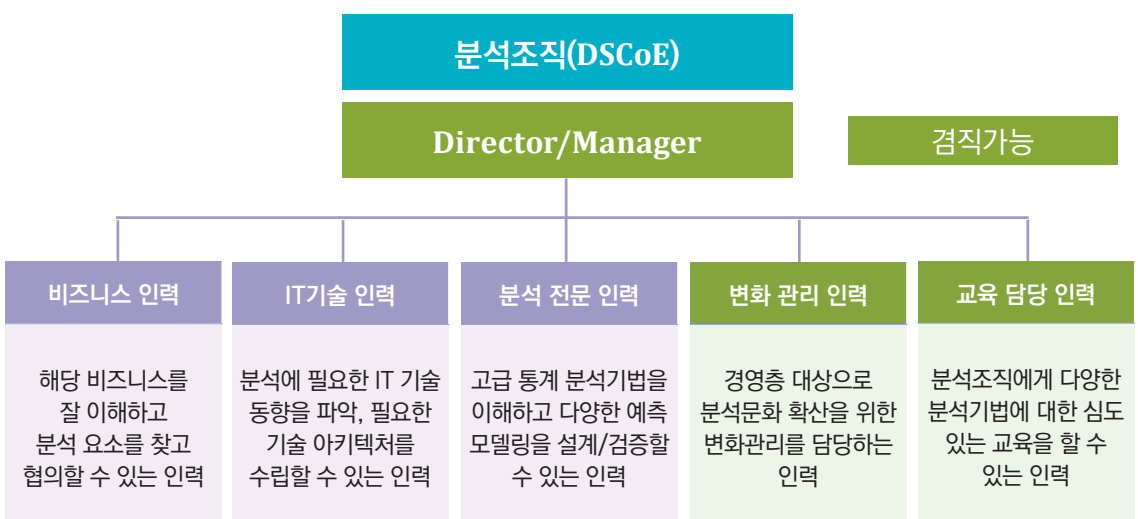
분석 조직 구조

집중구조	기능구조	분산구조
-전사 분석업무를 별도의 분석전담 조직에서 담당 -전략적 중요도에 따라 분석조직이 우선순위를 정해서 진행 가능 -현업 업무부서의 분석업무와 이중화/이원화 가능성 높음	-일반적인 분석 수행 구조 -별도 분석조직이 없고 해당 업무 부서에서 분석 수행 -전사적 핵심분석이 어려우며, 부서 현황 및 실적 통계 등 과거 실적에 국한된 분석 수행 가능성 높음	-분석 조직 인력들을 현업부서로 직접 배치하여 분석 업무 수행 -전사차원의 우선순위 수행 -분석결과에 따른 신속한 Action 가능 -베스트 프랙티스 공유 가능 -부서 분석업무와 역할 분담 명확해야 함(업무과다 이원화 가능성)
		

*DSCoE : Data Science Center of Excellence

분석 조직의 인력 구성

전문역량을 갖춘 각 분야의 인재들을 모아 조직을 구성하여 분석 조직의 경쟁력을 극대화할 수 있다.



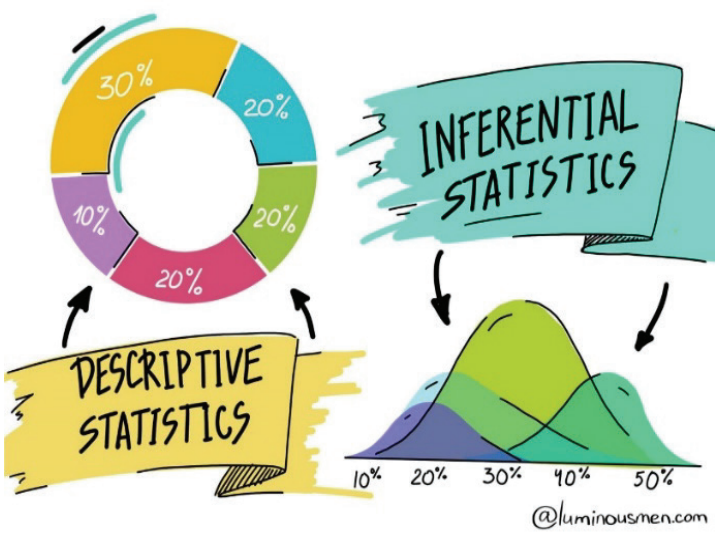
❷ 데이터 분석 필수지식(Data Analysis Knowledge)

기술통계(Descriptive Statistics)

기술통계는 직역하면 묘사적 통계이다. 데이터를 계량화한 수치값으로 표현한다. 평균, 최빈값, 중앙값을 구하고 분산을 구하는 것이 바로 기술통계이며 야구에서 투수의 방어율, 타자의 타율을 구하는 것이 기술통계에 속한다.

추론통계(Inferential Statistics)

추론통계는 추리, 추정하는 통계를 말한다. 어떤 데이터를 바탕으로 가까운 미래를 예측하는 것이 목표이다. 선거에서 일부 유권자들을 대상으로 출구조사를 진행하여 최종 투표결과를 한 발 더 빠르게 예측하는 것이 대표적인 추론통계에 해당한다. 이처럼 데이터 분석의 밑바탕에는 일부를 통해 전체를 바라보는 '통계적 추론'이 자리하고 있다.



〈기술통계와 추론통계〉
기술통계는 데이터의 계량화이고 추론통계는 데이터 바탕의 예측이 된다.
그림출처 : luminousmen.com

통계량

추출된 표본 데이터를 계량한 여러 종류의 수치(값)들을 말한다. 우리가 자주 사용하는 산술평균도 통계량 가운데 하나이다. 가령 어떤 표본의 산출평균값, 즉 표본평균이란 통계량이 있을 때, 이를 활용하여 모집단의 평균(모평균)을 예측하는 것이 가능해진다. 물리 분야의 물리량이 물리 법칙을 수학으로써 계량화한 것처럼 통계량 역시 같은 맥락의 개념이다.

주요 통계량	설명	수식
표본평균	· 데이터들의 중심 경향치로 산술, 조화, 기하평균 등 여러 종류가 있음	$E[\bar{X}] = \sum_{i=1}^{n} \bar{X}_i P(\bar{X}_i)$
표본분산	· 데이터 간의 퍼짐 정도를 표현하는 값 (= 데이터의 변동성을 대표하는 값)	$Var[\bar{X}] = \sum_{i=1}^{n} (\bar{X}_i - \mu_{\bar{X}})^2 P(\bar{X}_i)$
표준편차	· 평균과 데이터 간 표준 차이	$Std[\bar{X}] = E[(\bar{X} - \mu_{\bar{X}})]$
표준오차	· 표본평균들의 표준편차 (=표준분포의 표준오차)	$\sigma[\bar{X}] = \frac{\sigma}{\sqrt{n}}$
표본비율	· 어떤 성질을 갖는 요소가 표본에서 차지하는 비율	

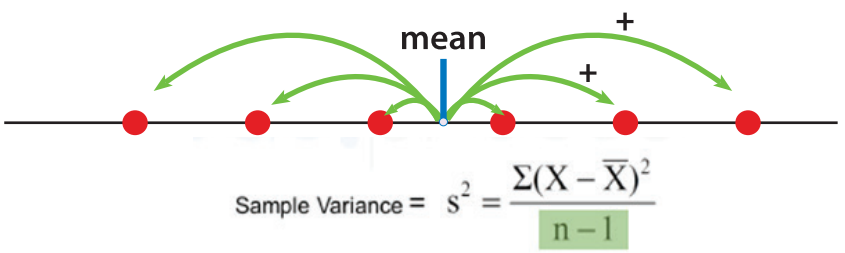
〈통계량의 종류〉

통계량은 표본을 수치 계량한 내용으로 표본의 특징을 설명해주는 기능을 한다.

분산(Variance)

데이터가 얼마나 '퍼져있는지'를 표현하는 통계량이다. 분산이 클수록 데이터 서로 간의 퍼짐이 크고, 작을수록 데이터 퍼짐 정도가 작다. 평균에서 데이터 각각의 값을 빼고(편차를 구하고) 제곱하여 그 값을 모두 더한 뒤, 전체 데이터 개수에서 하나의 데이터를 뺀 값(N-1)으로 나누어 분산을 구한다. 데이터 간 거리(편차)를 구한 값이 음수인 경우 양수로 바꾸어 데이터 간 거리를 설명한다.

데이터의 변동성(variability), 흩어짐 정도를 표현 : 편차제곱합/자료의 개수

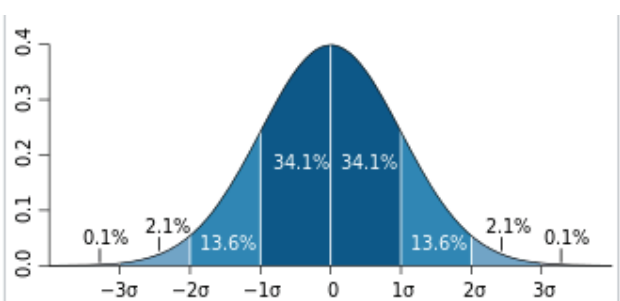


〈분산의 이해〉

분산은 데이터의 움직임(variability)을 설명할 수 있는 통계량이다.
분산은 데이터 간 거리를 표현한 것이고 이때 데이터 간 거리는 '변동성'을 설명한다.
분산을 다른 말로 '변량'으로 부르는 이유다.
그림출처 : 멀티캠퍼스

표준편차(Standard Deviation)

분산에 양의 제곱근하여 구한다. 평균과 개별 데이터 간 차이(=편차)의 표준값이라 생각하면 쉽다. 표본의 표준편차는 보통 S(또는 시그마 σ)로 칭한다. 각 개별 데이터와 평균값 간 차이들의 표준값이다. 따라서 표준편차는 데이터 분포의 간격을 설명할 수 있는 대푯값의 기능을 갖는다.



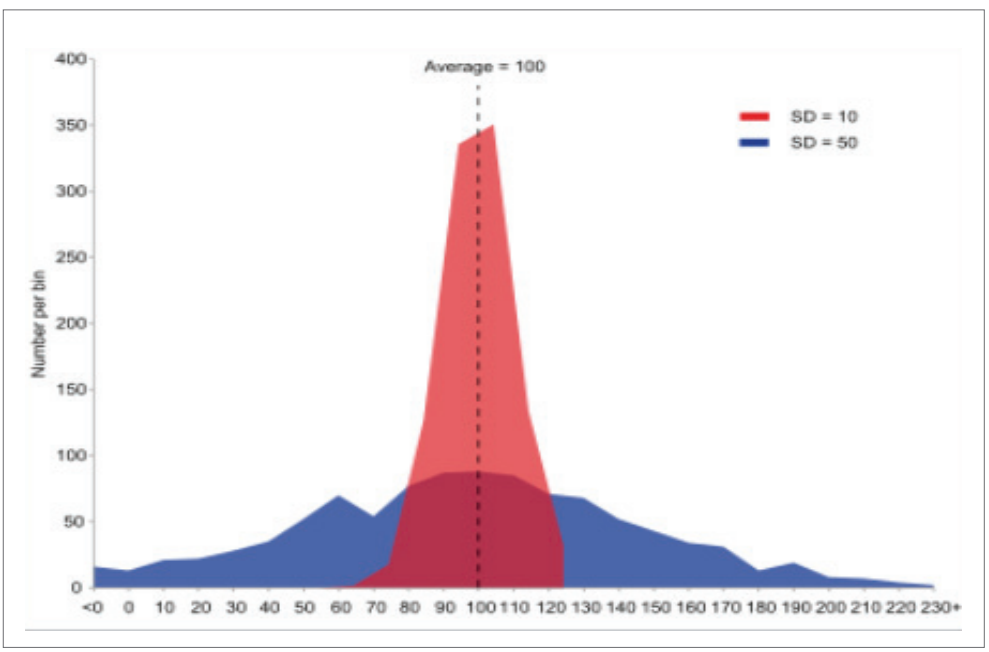
〈표준편차의 이해〉

표준편차는 데이터의 간격을 설명할 수 있는 값이다. σ의 차이(=간격)가 모두 1인 경우를 표현하였다.
그림출처 : Wiki

표준편차와 분산의 특징

분산은 평균의 한계를 보완할 수 있는 통계량이다. 아래 그림처럼 평균은 데이터를 설명하는데 뚜렷한 한계를 갖는다. 붉은색 영역이나 파란색 영역 모두 평균값은 동일하게 나오기 때문이다. 데이터 간 분포의 차이가 뚜렷함에도 불구하고 평균만으로는 그 차이를 설명할 수가 없다.

붉은색 영역에 비하여 파란색 부분의 데이터 분포는 표준편차가 상대적으로 크고 분산값도 훨씬 크다. 붉은색 영역의 표준편차는 10인데, 파란색 영역의 표준편차는 50으로 분산으로 보면 전자는 100, 후자는 2500이 된다. 표준편차와 분산은 그 값이 클수록 데이터의 변화 정도가 크다고 할 수 있다.



〈데이터의 성격을 말해주는 분산〉

두 자료의 평균은 같지만 데이터 간 표준 거리가 다르다. 파란색 영역의 편차들이 상대적으로 매우 큰 것을 알 수 있는데, 우리는 이때 붉은 부분에 비하여 파란 부분의 변동성(=분산)이 크다고 말할 수 있다.

자료출처 : Wiki

사분위수

데이터를 4등분하는 위치의 수를 사분위수라 칭한다. 전체 데이터를 순위별로 4등분 하는데, 이때 등분 지점이 되는 위치의 수를 Q1, Q2, Q3라 한다.

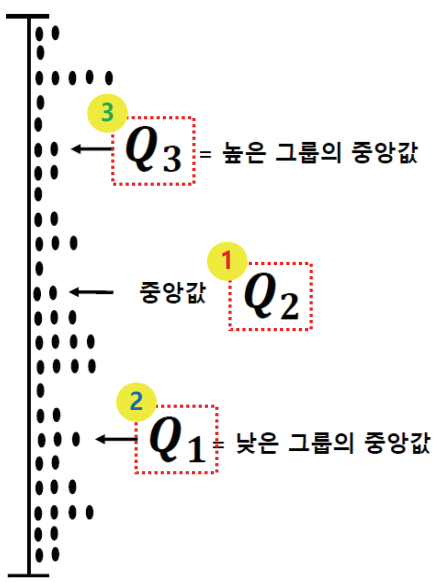
사분위수	설명	비고
제1사분위수(Q1)	· 누적 백분율이 25%에 해당하는 값	· 25번째 백분위수
제2사분위수(Q2)	· 누적 백분율이 50%에 해당하는 값	· 50번째 백분위수, 중앙값
제3사분위수(Q3)	· 누적 백분율이 75%에 해당하는 값	· 75번째 백분위수

〈사분위수의 구성〉
데이터를 백분율로 누적했을 때 25% 단위로 4등분 하는 3개의 지점이다.

사분위 범위(Inter-Quartile Range) 확인

사분위의 범위를 찾는 방법으로 먼저 할 일은 ①데이터를 숫자 순으로 정리하는 것이다. 다음으로 ②데이터를 숫자가 낮은 그룹과 숫자가 높은 그룹 2개로 나누고, 마지막으로 ③숫자가 낮은 그룹과 높은 그룹 각각의 중앙값을 순서대로 찾으면 된다.

이렇게 하면 전체 데이터의 중앙값을 하나 찾게 되고 숫자가 낮은 그룹의 중앙값 하나, 높은 그룹의 중앙값 하나, 총 3개의 중앙값(경계값)을 구할 수 있게 된다. 이와 같은 방법으로 얻어낸 **IQR**을 통해서 우리는 데이터가 얼마나 흩어져 있는지에 대한 정도를 알 수 있다.

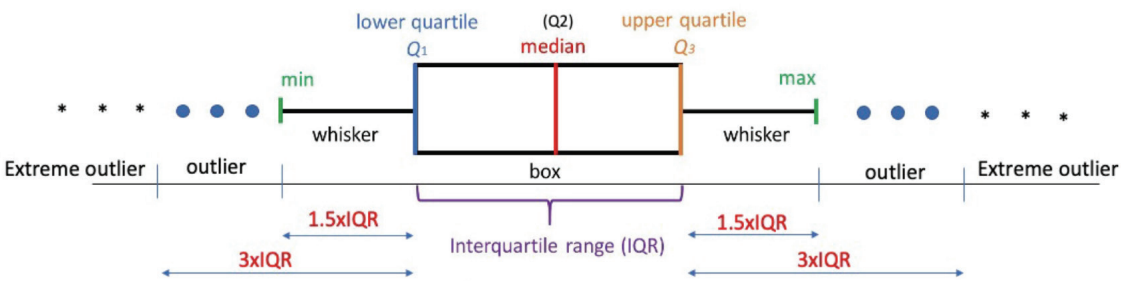


〈사분위 범위 찾기〉
$IQR = Q_3 - Q_1$
그림출처 : Wiki

IQR과 박스플롯(Box plot)

자료를 구성하는 데이터(산포)의 범위가 어느 정도인지를 IQR을 통하여 계산된 내용을 박스플롯 그래프로 시각화하여 직관적으로 파악해 볼 수 있다. 박스플롯은 Q1에서 Q3까지를 박스로 표현해주고 해당 박스에 전체 중앙값(Q2)을 실선으로 표기해 준다. 그뿐만 아니라 극단값(outlier)까지 확인할 수 있어 데이터 분석 전에 데이터의 구성과 특징을 한눈에 이해하는 데 자주 사용되는 시각화 도구 중 하나이다.

상자그림(box plot)



〈박스플롯의 구성〉

데이터 산포 구성을 IQR을 기준으로 박스 처리하여 표현하며 최솟값, 최댓값, 중앙값
그리고 이상치(outlier) 등 분석에 필요한 수치값들을 일목요연하게 시각화하는 기능을 갖는다.
그림출처 : 멀티캠퍼스

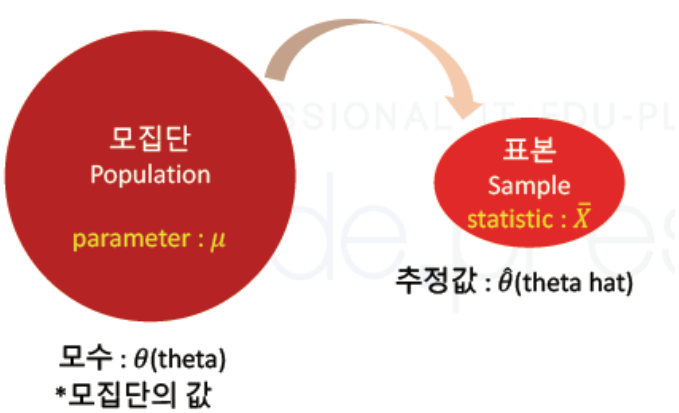
데이터전처리

분석 단계 전에 데이터를 정돈하는 개념이다. 엑셀 작업을 예로 들면 쉽다. 한 엑셀파일에 있는 데이터를 가지고 v-lookup 함수를 실행하고 싶을 때, 우리는 해당 함수를 편하게 쓸 수 있도록 필요 없는 컬럼을 삭제하고 컬럼 이름을 바꿀 수 있다. 그리고 성별이라는 항목이 있으면 남자를 '0'으로 표기할 수도 있고 여자를 '1'로 바꿔 표기(데이터 변환)해 놓을 수도 있을 것이다.

그리고 '이 빠진' 셀에 값을 나름의 논리로 채워 넣을 수도 있을 것(데이터 정제)이다. 또한 데이터를 선별해 별도의 도표를 만들어 데이터를 한 시트에 재정리(Sampling, 표본추출)하여 그 부분에만 함수를 적용할 수도 있다. 우리는 이 모든 행위를 '데이터전처리'라고 부른다. 다시 말해 데이터전처리란 분석의 편의와 정확성을 높이기 위해서 사전에 데이터를 정비하는 작업 일체를 말한다.

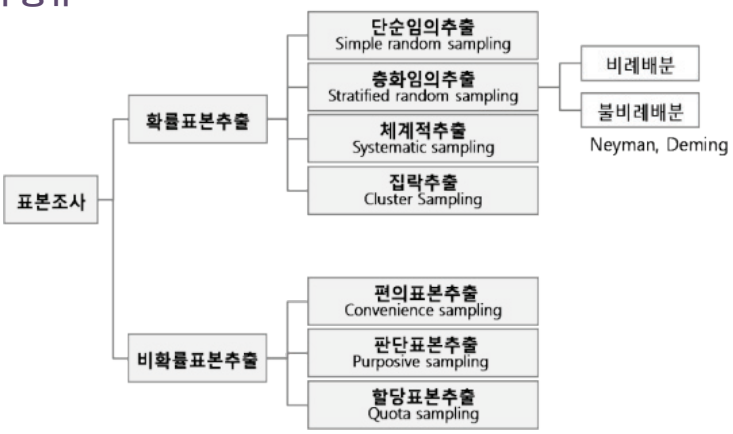
표본추출

모집단을 모두 분석하는 것이 불가능할 경우 수리통계적으로 타당하게 '샘플링'하여 분석할 수 있다. 100건의 데이터에서 10%를 추출하는 식으로 비중 중심의 단순한 데이터 준비가 아니라, 분석할 자료의 성질을 훼손하지 않으면서 합리적이고 타당하게 데이터를 추출하는 방법을 통계학에서 표본추출이라고 한다.



〈모집단과 표본〉
모집단을 대표하는 표본을 제대로 마련하는 것이 올바른 데이터 분석의 시작이다.
그림출처 : 멀티캠퍼스

표본추출의 종류



〈표본추출의 종류〉

크게 확률적/비확률적인 추출 방식으로 두 갈래가 있다.
본 책에서는 확률표본추출의 4가지 방식을 알아본다.
자료출처 : statictics.com

단순임의추출

무작위로 추출하지만 대신 추출 시 복원 또는 비복원할 수 있다. 복원은 자료를 추출한 뒤에도 그 자료를 다시 추출할 수 있는 개념이다. 비복원 랜덤 추출의 대표적인 예시로 로또 복권 추첨방식이 있다. 한 게임에서 한 번 뽑은 숫자는 복원될 수 없어 중복으로 나올 수 없다.

층화임의추출

계층(Class)별 랜덤 샘플링이다. 자료에서 클래스를 나누어 두고 모집단 클래스마다 샘플을 각각 추출하는데 모집단의 자료 구성비를 그대로 따르지는 않는다. 클래스 A에서 랜덤으로 추출한 샘플이 서른 개인데, 클래스 B의 자료 수 자체가 스무 개라면, 클래스 B에서는 표본을 스무 개만 추출해도 무방하다.

체계적추출

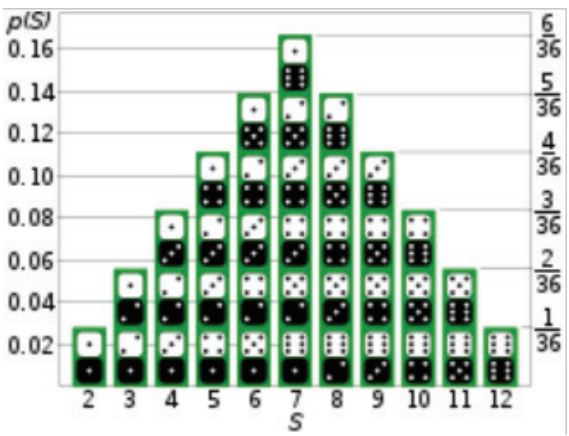
첫 번째 자료는 임의로 추출하고 다음 자료는 동일한 간격으로 뽑는다. 일종의 등차수열처럼 동일한 간격으로 자료를 추출하는 방법이다.

집락추출(=군집추출)

모집단 내에 나누어 놓은 클래스 중 하나를 선택하는 방식이다. 인구조사 중에 20대 남성 자료에서 강원도 원주시 거주하는 20대 남성만 뽑으면 집락추출이다.

확률변수와 확률분포

확률변수는 어떤 사건이 일어날 결과를 수로 표현한 것이다. 확률은 상황과 시간에 따라 변화함으로 그것을 기록해 놓을 필요가 있을 것이다. 이때 변화하는 확률변수의 값들을 정리한 것이 확률분포(표)이다. 당연히 확률변수(x)에 따라서 분포의 모양이 결정된다. 뒤에서 다루겠지만 확률변수(x)가 '연속형'일 때 우리가 잘 알고 있는 '정규분포'가 만들어진다.



〈확률값을 정리한 분포〉

주사위 2개를 던졌을 때 두 눈의 합에 대한 (이산)확률분포이다.
가령 두 눈의 합이 4가 나올 확률은 3/36이다.
그림출처 : statictics.com

자료의 종류

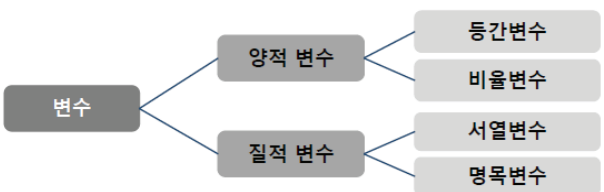
'이산형'의 경우에 별도로 이산형 확률분포를 참조하면 된다. 그러므로 우선적으로 분석할 자료의 종류를 살펴보고 관련된 분포(표)를 확인해야 한다. 자료의 종류에 따라서 설정되는 변수의 종류도 달라진다.

〈자료의 종류〉

크게 사칙연산이 가능한 양적자료와 그렇지 못한 질적자료로 나눠진다.
그림출처 : 교육자료

변수의 종류

연속형 자료일 경우에 변수는 등간변수가 활용된다. 여기서 등간은 간격이 일정한 독립변수(x)값을 의미한다. 일정한 시간 간격(Term)이 대표적이다. 이산형 자료에서 '이산'은 서로 이어질 수 없는 것을 뜻한다. 0을 포함한 자연수로 자료가 이뤄져 있다 하더라도 값들이 서로 이어질 수 없다면 해당 자료는 이산형 자료이다. 가령, 추신수 선수가 타석에서 안타를 치는 횟수를 10번의 타석마다 집계했을 때 해당 자료는 이산형이다. 만일 매 경기 순으로 안타 여부와 횟수를 파악했다면 연속형 자료가 될 것이다.

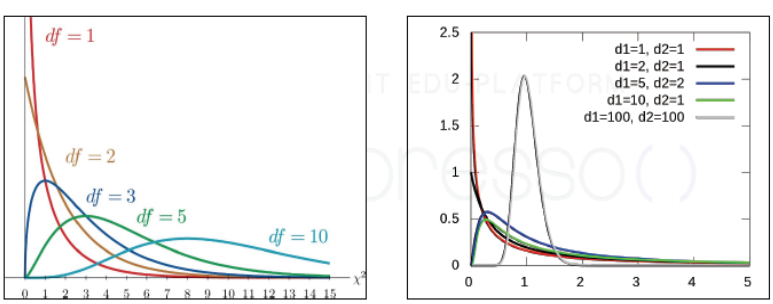


〈자료에 따른 변수의 종류〉

질적 변수는 범주 또는 순위에 부여된 기호로서의 의미를 가지며, 양적 변수는 유리수로 측정되는 절대적이거나 상대적인 양을 표현한다.
그림출처 : 교육자료

정규분포 그래프의 종류

중심극한정리에 의해 확률분포 함수를 정리한 것을 말한다. 정규분포를 좀 더 쉽게 정의하면 일종의 통계량에 대한 '답안지'라고 할 수 있다. 수학적인 방법을 통해 마련된 데이터 분포의 표준(안)으로써 우리가 어떤 값이 크고 작은지를 판단하는 기준으로 기능한다. 표준정규분포(z-분포), t-분포, 카이제곱분포, F-분포 등 수학적으로 증명된 분포(표)가 이미 존재하고 있으며 우리가 얻어낸 통계량을 적절한 분포(표)에 대조하여 관측치의 크고 작음을 판별할 수 있다.



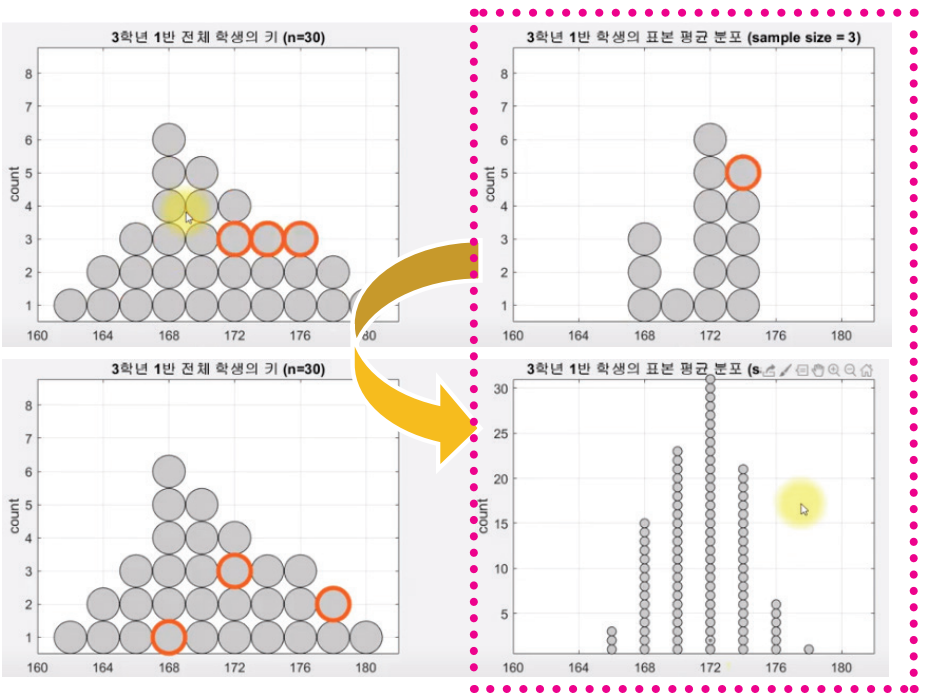
카이제곱분포 F-분포

〈분포 그래프의 예시〉

자료의 종류와 검정할 통계량에 따라서 답안지로 사용할 분포(표)가 다르다.
그림출처 : 교육자료

중심극한정리

표본의 수가 일정 수준을 충족하면 평균 데이터가 '정규분포' 형태에 근사하게 된다는 수학 이론이다. 다시 말해, 중심극한정리란 모집단의 크기와 상관없이 표본의 크기가 커질수록 표본평균값의 분포가 정규분포에 가까워지게 되는 원리이다.



〈중심극한정리 Matlab 예시〉

학급 학생의 키 데이터에서 표본 데이터를 3건 씩 추출하고 그 평균을 기록하는 것을 100회 반복한 프로그램 결과로 정규분포 모습처럼 Bell-curve를 나타내는 것을 확인할 수 있다.
그림출처 : statistical-engineering.com

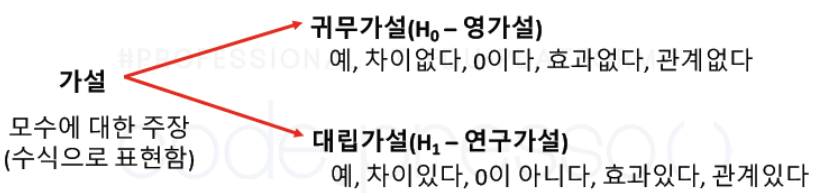
왜도 & 첨도

분포 그래프의 외형을 설명할 때 쓰인다. 분산과 표준편차로 자료의 분포도를 그려볼 수 있으나 특성을 바로 이해하기 어려울 수 있다. 이때 왜도는 그래프의 치우침을, 첨도는 그래프의 뾰족함을 표현해 주어 그 데이터의 모습을 직관적으로 이해할 수 있게 돕는다. 오른쪽으로 꼬리가 긴 분포는 평균 〉 중앙값 〉 최빈값의 특징을 가지며, 왼쪽으로 꼬리가 긴 경우에는 정반대이다.

귀무가설 & 대립가설

대립가설부터 이해하는 것이 편한데, 대립가설은 연구자 즉 분석가가 세운 가설이다. 분석을 통해 밝히고자 하는 내용이 바로 대립가설(=연구가설)인 것이다. 이때 귀무가설(H_0)은 대립가설(H_1)이 깨야 하는 가설이다.

이 두 가설은 논리적인 관계다. 가령 '남자의 평균 수명은 여자의 평균 수명보다 짧다.'는 대립가설을 세웠고 관련 데이터를 분석한 결과에서 어떤 관계성(유의성)을 찾지 못했다면, '여자의 평균 수명이 남자보다 더 길다.'가 귀무가설이 되는 것이 아니라, '성별은 평균 수명과 관계(유의성)가 없다.'로 귀무가설을 채택하여 결론 내야 올바른 해석이다. 남자의 반대는 '여자'가 아니라, '남자가 아니다.'란 것을 유념한다면 혼란스러운 일은 없을 것이다.



〈귀무가설과 대립가설〉

두 가설을 통해 가설에 대한 검정(Test)을 실시한다. 연구가설이 증명되지 못하면 귀무가설이 '채택'되고, 증명되면 귀무가설이 '기각'된다.
그림출처 : 교육자료

1종 오류

귀무가설을 잘못 기각했을 시 발생하는 오류다. 귀무가설이 실제로 참이라 채택해야 하나 반대로 기각해버린 경우다. 거짓 양성 또는 α 오류라고도 불린다.

1종 오류를 범할 확률은 α로, 가설 검정에 대해 설정한 유의 수준이다. α가 0.05면 귀무가설이 5% 확률로 잘못 기각된다는 의미이다. 이 위험을 낮추기 위해서는 더 낮은 α값을 사용해야 하지만, 더 낮은 α값을 사용하면 실제로 존재하는 실제 차이를 탐지할 가능성이 더 적다. 위의 귀무가설 예시에서 '성별은 평균 수명과 관계가 없는데 관계가 있다'고 판별한다면 1종 오류라고 할 수 있다.

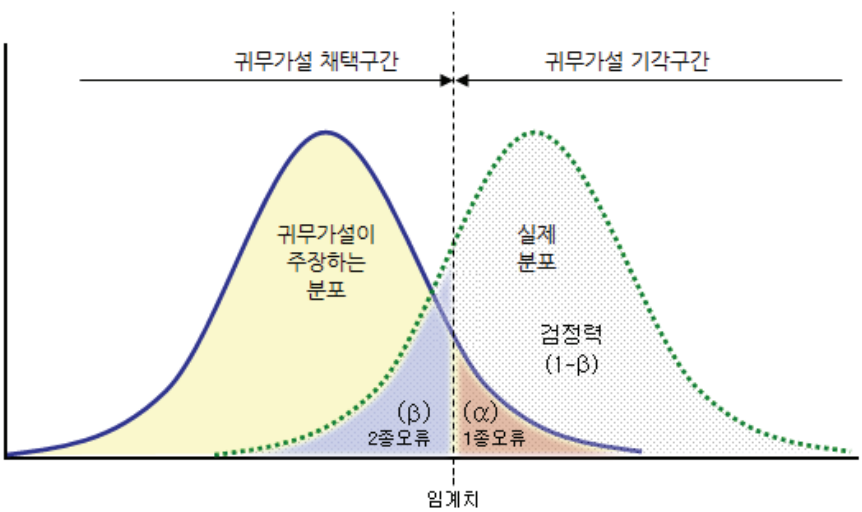
2종 오류

귀무가설을 잘못 채택했을 시 발생하는 오류다. 귀무가설이 실제로 거짓이라 기각해야 하나 반대로 채택해 버린 경우다. 거짓 음성 또는 β 오류라고도 불린다.

2종 오류를 범할 확률은 β로, 검정력을 충분하게 설정함으로써 2종 오류를 범할 위험을 줄일 수 있다. 실제 존재하는 차이를 탐지할 수 있을 정도로 표본 크기를 크게 만들면 된다. 거짓인 귀무가설을 기각할 확률은 $1-\beta$인데 이 값이 검정력이다. 위의 귀무가설 예시에서 '성별은 평균 수명과 관계가 있는데 관계가 없다'고 판별한다면 2종 오류라고 할 수 있다.

검정결과 \ 실제상황		모집단에 대한 사실	
		H_0가 참 (불이 나지 않았음)	H_0가 거짓 (화재 발생)
표본 기반 결정	H_0를 채택 (화재 경보 울리지 않음)	옳은 결정 $(1-\alpha)$	2종 오류 (β) (불이 났는데도 화재 경보 울리지 않음)
	H_0를 기각 (화재 경보 울림)	1종 오류 (α) (불이 나지 않았는데도 화재 경보 울림)	옳은 결정$(1-\beta)$ (=검정력)

〈1, 2종 오류에 대한 요약〉 H_0: 귀무가설



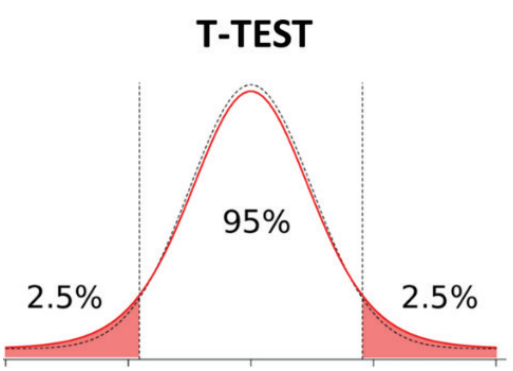
〈1, 2종 오류에 대한 확률 그래프〉

α와 β는 상반 관계이므로 α와 β의 크기가 동시에 감소하는 것은 불가능하다.
일반적으로 α에 기준을 두고 가설을 채택하거나 기각한다.

가설검정과 t-검정

먼저 가설검정이란 대립가설을 수립 후에 통계적 검정 방법에 따라 귀무가설을 기각할지 채택할지를 테스트하는 방법이다. 다음으로 t-검정은 몇 가지 통계적인 검정 방법 가운데 하나로 t-분포(표)를 활용한 두 자료 간에 통계적인 '비교'로 이해하면 접근이 쉽겠다. 비교를 위해 평균과 표준편차 통계량을 사용한다. 평균값의 차이가 얼마인지 표준편차는 얼마나 다른지 확인해 분석 대상인 두 자료가 같을 가능성이 우연의 범위 5%(좌우 2.5%씩)에 들어가는지 아닌지 판별하는 것이 핵심이다.

앞서 분산의 개념을 이해했다면 평균값은 데이터의 분포를 설명하는 데 한계가 있다는 사실을 알 것이다. 평균이 같더라도 데이터의 분포, 즉 분산(s^2)과 표준편차(s)가 다를 수 있기 때문이다. 따라서 데이터 분포의 모양까지 비교할 수 있는 통계량인 표준편차를 비교에 함께 활용하게 된다. 여기서 비교는 평균처럼 값에 대한 절대적인 비교는 아니고 비례적인 비교로 A집단의 표준편차와 B집단의 표준편차의 비율을 비교한다. 정리하면, t-test는 두 집단의 평균값과 표준편차 비율에 대한 대조 검정법인 것이다.

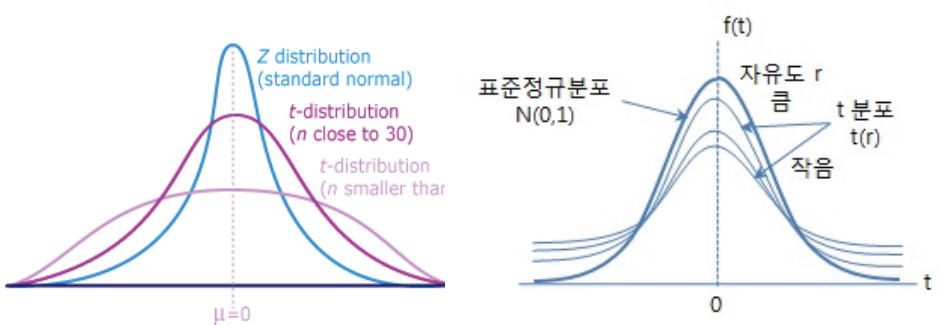


〈t-test〉
두 집단의 비교를 위한 평균과 표준편차의 비율 차이를 구하고
그 차이가 우연이냐 아니냐를 판단하는 검정이다.
그림출처:statictics.com

t-분포

서로 다른 두 집단의 평균에 대한 통계 검정에 주로 사용된다. 주로 모집단이 정규분포라는 정도만 알고 모표준편차는 모를 때, 모평균 추정을 위해 표준정규분포를 대신해 소표본(n < 30)의 분포인 t-분포를 사용한다. 정규분포로부터 표본을 구할 때 표본의 크기가 크지 않고, 표준편차(σ)를 모른다면 $t = \frac{X-\mu}{s/\sqrt{n}}$ 는 자유도가 n-1인 t-분포를 따른다.

여기서 t-분포의 특징은 모양이 정규분포보다 퍼져있으며 자유도가 커질수록 정규분포에 근접한다는 것이다. 표본의 크기가 작아 표본의 표준편차가 모집단의 표준편차보다 불확실성이 크기 때문인데, 표본의 크기 n이 커질수록 표본의 표준편차가 모집단의 표준편차에 접근한다. 그래서 t-분포의 모양은 자유도(≒ 좌푯값의 수)에 따라 달라진다.



〈t-분포의 예시〉

표본의 수(=자유도)가 작을수록 분포의 모양이 완만하고 자유도가 클수록 정규분포의 모양을 띄게 된다. 즉, 자유도가 커질수록 좌측의 그림처럼 표준정규분포에 근사하게 된다.
그림출처 : ktword.co.kr

자유도(Degree of freedom)

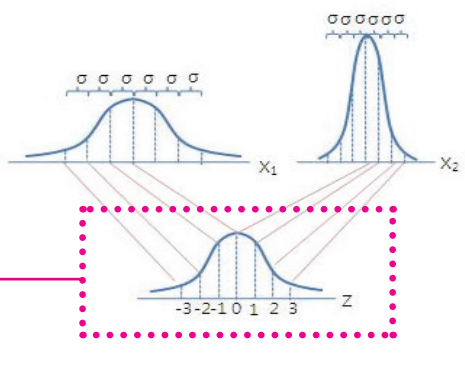
역학에서 어떤 객체(Object)의 움직임을 나타낼 수 있는 최소한의 독립변수(x)의 수를 의미한다. 통계 분야에서의 자유도(df)는 결국 좌푯값의 개수인 것이다. 따라서 자유도가 커질수록 표준정규분포에 근접하게 된다.

표준정규분포와 z-score

t-검정에 대한 개념과 함께 표준정규분포와 z-test를 이해하면 보다 쉽다. z-검정 역시 통계적인 검정 방법 중 하나인데 여기서 중요한 개념이 z-score이다. 앞서 다뤘던 정규분포의 개념을 명확하게 이해하고 있다면 우리는 z-score를 보다 쉽게 배울 수 있다.

정규분포는 평균값을 기준으로 좌우 대칭이 이뤄지는 분포 그래프였다. 이때 대칭인 종 모양(bell-curve) 그래프는 무수히 많은 수로 존재할 것이다. 그러므로 어떤 데이터의 정규분포마다 그 영역의 면적을 계산(적분)하긴 비효율적이다. 우리는 이럴 때 '표준'정규분포를 활용한다. 대상이 되는 정규분포를 바로 표준정규분포(모양)로 변환하는 것이다. 더불어 변화에 사용되는 공식이 바로 z-score이다.

$$z\text{-score} = \frac{\text{관측값}(X) - \text{평균}(\mu)}{\text{표준편차}(\sigma)}$$



z	+ 0.00	+ 0.01	+ 0.02	+ 0.03	+ 0.04	+ 0.05	+ 0.06	+ 0.07	+ 0.08	+ 0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56360	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524

〈여러 정규분포의 표준화〉

이미 마련된 표준정규분포표를 활용해 z-검정을 실시한다. t를 z로!

표 출처 : Wiki

z-분포

표준정규분포(Standard normal distribution)라고도 한다. 대표적인 연속형 확률분포인 정규분포에는 몇 가지 특징이 있는데, 우선 정규분포의 모양과 위치는 분포의 평균(μ)과 표준편차(σ)로 결정된다. 정규 분포의 확률 밀도 함수는 평균을 중심으로 대칭인 종 모양을 가지며, 정규분포를 나타내는 정규 곡선은 X축에 맞닿지 않으므로 확률변수 X가 취할 수 있는 값의 범위는 -∞ 〈 X 〈 +∞이다.(관측값의 99.7%가 ±3 σ 안에 속해 있다.) 정규분포의 평균과 표준편차가 어떤 값을 갖더라도 정규 곡선과 X축 사이의 전체 면적은 1이다. 정규분포의 확률 밀도 함수는 다음과 같이 나타낼 수 있다.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

이때 평균은 μ이고, 표준편차가 σ인 정규분포를 따르는 확률변수 X를 다음과 같이 표현하기도 한다.

$$X \sim N(\mu, \sigma^2)$$

이러한 정규분포를 평균 μ=0, 표준편차 σ=1이 되도록 표준화한 것을 표준정규분포라고 부르는데, 어떤 관측값의 확률분포 X를 원래 크기가 아닌 평균을 중심으로 값의 크기를 조정하고, 그 값이 평균으로부터 표준편차의 몇 배 정도나 떨어져 있는지를 나타낸 확률분포이다. 이때 표준정규분포는 확률변수 Z로 나타내며, N(0, 1)으로 표시한다.

z	+ 0.00	+ 0.01	+ 0.02	+ 0.03	+ 0.04	+ 0.05	+ 0.06	+ 0.07	+ 0.08	+ 0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56360	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524

〈표준정규분포표〉

z값 0부터 0.75사이의 영역값은 0.77337−0.5=0.27337임을 알 수 있다.

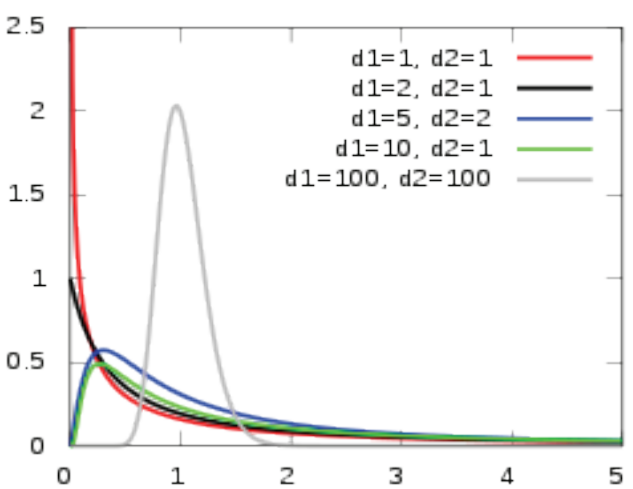
F-분포

F분포는 서로 다른 두 개 이상의 모집단의 분산이 서로 같은지를 확인할 때 사용된다. 즉, 분산분석과 회귀분석의 결과를 해석할 때 주로 사용된다.

모집단이 정규분포를 이루며 각각 σ_1^2, σ_2^2 이라는 분산을 갖는 두 개의 모집단에서 각 크기가 n_1, n_2인 두 표본을 추출하여 표본 분산을 계산한다고 하자. 두 표본 분산이 S_1^2, S_2^2이라고 할 때 표본 분산과 모분산의 비율로 이루어진 두 개의 χ^2의 비율은 F분포를 이루며, F분포는 두 개의 자유도를 갖는다.

$$\chi 1^2 = \frac{(n1-1)S12}{\sigma 12} \sim \chi 1^2(\mathbf{n1-1}),\ \chi 2^2 = \frac{(n2-1)S22}{\sigma 22} \sim \chi 2^2(\mathbf{n2-1})$$

$$\frac{\chi 1^2/(\mathbf{n1-1})}{\chi 2^2/(\mathbf{n2-1})} = \frac{S1^2/\sigma 12}{S2^2/\sigma 22} \sim \mathbf{F(n1-1, n2-1)}$$



〈F분포의 확률 밀도 함수 예시〉

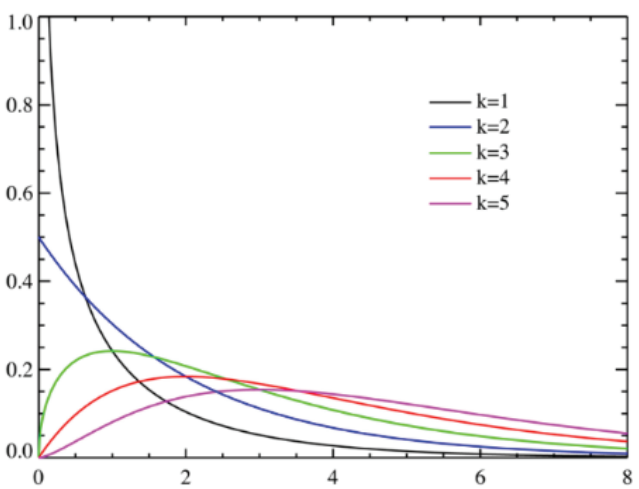
다른 분포와 마찬가지로 자유도에 따라서 그래프의 모양이 결정되며 값이 커질수록 좌우 대칭의 그래프가 나타난다.
그림출처 : Wiki

카이제곱분포(χ^2)

χ^2분포는 두 개 이상의 서로 다른 범주에 대해 가설 검정, 모분산의 추정, 분포 간의 차이 등에 많이 사용된다. 단일 모집단에서 서로 독립적인 확률변수를 제곱한 후 더 하는 분포는 χ^2 분포를 이용하여 나타낼 수 있다. 특히, 정규 모집단 $N(\mu, \sigma^2)$으로부터의 확률 표본 X1, X2, … , Xn에 대하여

$$X^2 = \frac{\sum(xi-\bar{x})^2}{\sigma 2} = \frac{(n-1)s2}{\sigma 2}$$

수식인 분포를 자유도 n−1인 χ^2 분포라 한다. x^2확률변수는 연속확률변수인 표준정규변수의 함수이므로 똑같이 연속확률분포를 따른다. 또한, 제곱의 합으로 정의되기 때문에 항상 0 이상의 값만을 갖는다. 더불어 E(U) = v, Var(U) = 2v가 성립하여 곧, 카이제곱분포는 평균이 자유도와 같고, 분산은 자유도의 두 배 이다.



〈χ^2 분포 그래프 예시〉

자유도 k에 따른 χ^2분포의 확률밀도 함수 그래프. 기본적으로 좌측에 치우친 분포인데, 자유도가 커질수록 대칭의 분포로 접근한다.
그림출처 : Wiki

3 데이터 분석 모델평가(Data Analysis Model Assessment)

분류성능평가

정답과 오답을 오차행렬(2×2)의 형태로 세분하여 분류성능평가지표를 만들고 이를 분석 모델의 정확도 평가를 적용하는 테스트 방법이다. 오차행렬은 정답과 오답을 True(Real)과 False(Fake)로 쪼개어 작성하는데, 이를 순서대로 표현하면 진양성(True Positive) / 위양성(False Positive) / 위음성(False Negative) / 진음성(True Negative)이라 하고, 이를 쉽게 설명하면, **맞는 정답 / 틀린 정답 / 틀린 오답 / 맞는 오답**으로 정리된다. 표로 정리하면 오차행렬로 확인할 수 있다.

		실제결과	
		양성	음성
모델예측	1 양성(P)	A 진양성(TP)	위양성(FP)
	2 음성(N)	위음성(FN)	B 진음성(TN)

〈오차행렬 구성〉
성능평가를 위해 분석 결과를 분류한 오차행렬을 작성하면, 모델이 예측한 정보에 대하여 실제 결과의 정/오를 가려 세분화할 수 있다.

분류성능평가지표

오차행렬의 결과를 활용하여 5가지의 성능 테스트에 관한 지표를 생성해 활용할 수 있다. 먼저 '**정밀도**'는 모델(분석모형)의 예측 내용 가운데 실제 결과에서 정답이 얼마나 있는지 확인하는 지표이다. 예측한 **1 전체양성(TP+FP)**값들로 **A 진양성(TP)**을 나누어 값을 구한다. 그러므로 정밀도의 숫자가 크다면, 모델이 정답 내용의 양성 정보를 보고 정답을 제대로 찾은 비율이 높다고 평가할 수 있게 된다.

두 번째 '특이도'는 모델 예측의 **2 전체음성(TN+FN)** 가운데 **B 진음성(FN)**을 제대로 맞춘 비중이다. 그래서 해당 지표 역시 값이 클수록 모델이 음성을 음성이라고 제대로 맞추었다고 판단하는 기준이 된다.

세 번째로 '**민감도**'란 지표가 있다. 실제 정/오 결과(TP+FN)가 일치하는 정도를 판단한 비율이다. 실제로 정답을 얼마나 맞추었는지에 초점을 맞춘 값이다. 정밀도가 모델의 관점에서 정답을 정답이라고 했다면, 민감도는 실제 정답(data)의 관점에서 정답을 정답이라고 맞춘 경우이다. 정밀도와 민감도는 상호 보완적으로 사용할 수 있으며, 두 지표가 높을수록 좋은 모델이다.

실제 맞고 틀림의 결과에서 정답 전체의 수를 전체 데이터 수로 나누면 보통 우리가 생각하는 '**정확도**'란 네 번째 지표가 나온다. 전체 데이터 가운데 정답을 맞춘 비율을 체크한 것이다. 정확도는 가장 직관적으로 모델의 성능을 나타낼 수 있는 평가지표이다.

분류성능평가지표	설명	수식
정밀도(Precision)	· 양성 가운데 맞춘 양성의 수	$\frac{TP}{TP+FP}$
특이도(Specificity)	· 음성 가운데 맞춘 음성의 수	$\frac{TN}{TN+FN}$
민감도(Sensitivity) (=재현율 Recall)	· 실제 양성을 양성으로 음성을 음성으로 판단한 비율	$\frac{TP}{TP+FN}$
정확도(Accuracy) (=정분류율)	· 전체 데이터에서 모델이 옳게 판단한 비율	$\frac{TP+TN}{TP+FN+FP+TN}$
성능점수(F1 Score)	· 정밀도와 민감도의 조화평균	$2\times\frac{\text{정밀도}\times\text{민감도}}{\text{정밀도}+\text{민감도}}$

정밀도와 정확도의 차이

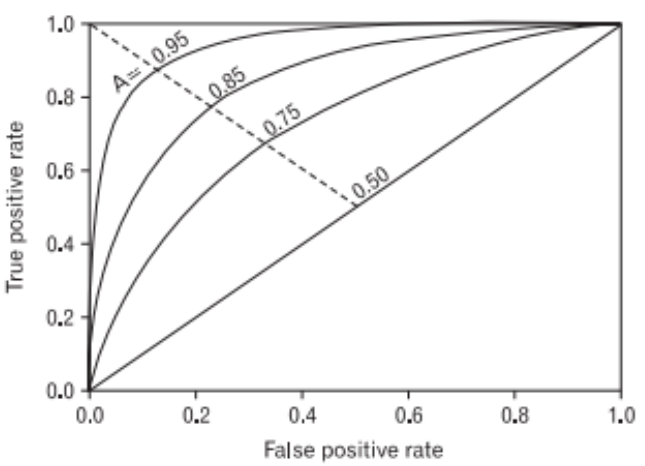
스포츠 사격에서의 개념을 빌려 설명하면, 두 개념의 차이를 이해하기 쉽다. 정밀도가 의미하는 바는 바로 '영점'이다. 과녁 중심에 사격점들이 모이지 않더라도 특정 부분에 사격점들이 모여 있다면 그때 정밀도가 높다고 표현한다.

한편 정확도가 의미하는 바는 과녁의 중심에 얼마나 사격점이 잘 분포하고 있느냐 이다. 그러므로 정밀도는 정확도를 포함하지 못하지만 정확도는 정밀도를 포함한다. 과녁의 중심에 사격점들이 모여 있다면 정확도가 높으면서 동시에 정밀도도 높은 것이고, 과녁의 중심과는 거리가 있으나 그래도 어떤 한 부분에 사격점들이 잘 모였다면 정확도는 낮되, 정밀도는 높다고 말할 수 있다.

마지막으로 '**성능점수(F1-score)**'는 정밀도와 민감도의 조화평균 계산을 한 결과인데 결괏값이 작아지는 특징이 있다. 성능을 하나의 숫자로 표현할 수 있으며, 1에 가까울수록 최적의 정밀도와 민감도이고 0은 최악이라는 뜻이다. 데이터 라벨이 불균형 구조일 때, 두 지표를 모두 균형있게 반영하여 모델의 성능을 정확하게 평가할 수 있다.

ROC곡선(Receiver Operating characteristic Curve)

기준선(Threshold)이 달라짐에 따라 분류 AI모델의 성능이 어떤지를 한눈에 볼 수 있다. 기준선에 따라 성능평가의 지표가 달라진다. ROC는 위양성률(1-특이도)을 x축으로, 그에 대한 실제 양성률(민감도)을 y축으로 놓고 그 좌푯값들을 이어 그래프로 표현한 것이다. 일반적으로 0.7~0.8 수준이 보통의 성능을 의미한다. 0.8~0.9는 좋음, 0.9~1.0은 매우 좋은 성능을 보이는 모델이라 평가할 수 있다.



〈ROC곡선〉

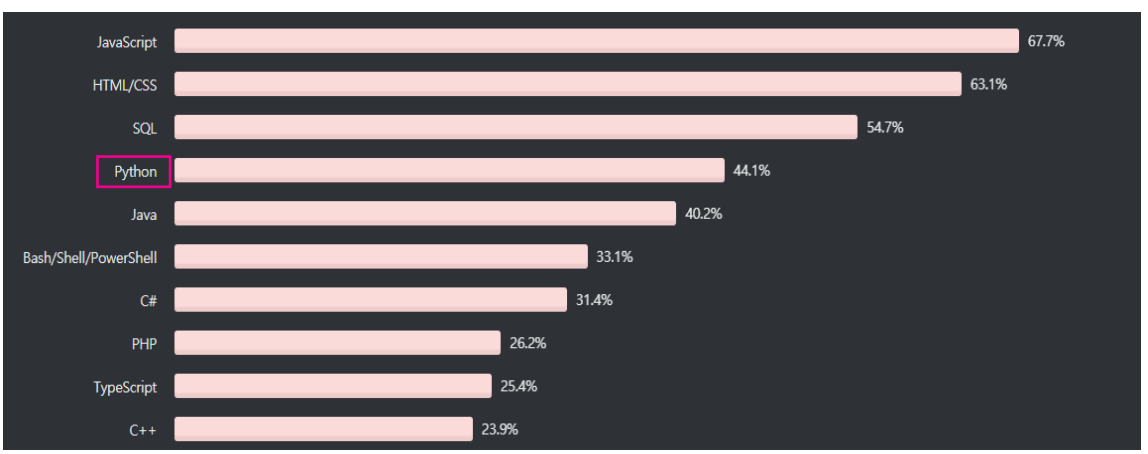
모델이 양성이라고 예측한 결과에서 맞춘 정답과 틀린 정답의 비를 나타낸 그래프이다.
당연히 맞춘 정답(=진양성)의 비가 높을수록 좋은 성능일 것이다.
그림출처 : Wiki

4 데이터 분석 도구 소개 python

파이썬과 통계

현대에 들어와 통계학이 발전한 것처럼 컴퓨터(공)학도 눈부신 발전을 거듭했다. 컴퓨터 분야의 하드웨어와 소프트웨어가 전자계산을 사람 대신 수행하게 되면서 여러 가지 계산을 하여 값을 계량하고 검정할 때 필요한 수학과 통계 분야의 지식들이 컴퓨터 기술에 녹아들기 시작했다. 게다가 다루어야 할 데이터의 값의 크기나 양도 많아지면서 반복적이고 거의 무한에 가까운 연산과 수치 계량을 컴퓨터가 대신하게 된 것이다.

데이터 분석에 주로 사용되는 프로그래밍 언어들은 이러한 배경에서 탄생했다. 그런 언어 중에 무료로 사용 가능한 통계용 언어가 R이고 파이썬(Pandas, Numpy 등)인 것이다. 따라서 여기서는 파이썬을 데이터 분석용 프로그래밍 도구라 이해하면 좋겠다. 여러 통계적 분석 기법들이 이미 파이썬으로 구현돼 있다. 우린 이것을 가져다가 사용하고 그 결과를 해석하면 되는 것이다. 우리는 앞으로 바로 이 내용을 확인해 갈 것이다.



〈프로그래밍 언어 사용 순위 상위 10건〉
현직 약 5만 8천 명을 대상으로 한 응답 결과다. 같은 조사에서 R은 18위를 기록했다. 파이썬은 분석 용도 외에도 서버 개발 등 범용성이 높아 R에 비해 높은 사용률을 보인다.
출처 : StackOverflow 2020 Survey

파이썬 라이브러리(Library)

'라이브러리'란 표현은 전산 개발 업무에서 많이 쓰이는 용어이다. 가령 우리가 프로그램을 하나 작성할 때 모든 로직(=프로그램 논리)을 일일이 다 작성하기 어렵다. 게다가 그러한 개발 방식은 굉장히 비효율적이다. 그래서 도서관에 모아 놓은 책처럼, 주제/목적별로 프로그램들을 따로 모아 놓고 필요한 책 제목만 코딩 시에 선언하면 그 프로그램을 사용할 수 있게 구성해 놓은 것이 바로 라이브러리이다. 파이썬이 참조하는 대표적인 라이브러리를 몇 가지 소개하면 다음과 같다.

라이브러리	설명
pandas	· 고유하게 정의한 Series 및 DataFrame 등의 자료구조를 활용하도록 지원
matplotlib	· 각종 그래프나 차트 등 데이터 시각화 기능 제공
seaborn	· 고급 통계 차트 기능을 제공하며 matplotlib과 상호의존(서로 참조)적
statsmodels	· 추정 및 검정, 회귀분석, 시계열분석 등의 기능을 제공
scikit-learn	· 파이썬의 대표적인 머신러닝용 패키지
tensorflow	· 구글에서 만든 라이브러리로 딥러닝 프로그래밍을 지원하는 패키지

〈파이썬 라이브러리 일부〉

파이썬이 '영어'라면, pandas는 '영문수학책'이고 tensorflow는 '영문AI책'과 같다.
내용출처 : Python.org

파이썬 도큐먼테이션(Documentation)

오픈소스인 파이썬은 공식 문서(매뉴얼) 역시 무료로 제공한다. Python.org 웹 사이트에 접속하면 정식 매뉴얼을 다운받아 열람할 수 있는데 내용이 매우 방대하나 가장 정확한 예제나 언어 사용 문법을 확인할 수 있으니 적극적으로 활용하면 개인적으로도 분석 코딩 공부가 가능하다.

II. 통계분석기법

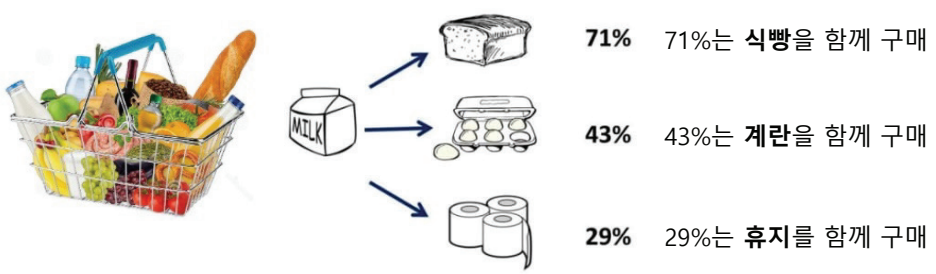
1. 연관규칙분석(Association Rule Analysis)
2. 교차분석(Cross-tabulation Analysis)
3. 분산분석(Analysis of Variance, ANOVA)
4. 상관분석(Correlation Analysis)
5. 주성분분석(Principal Component Analysis)
6. 회귀분석(Regression Analysis)
7. 로지스틱 회귀분석(Logistic Regression Analysis)
8. 시계열분석(Time-series Analysis)

II. 통계분석기법

1 연관규칙분석(Association Rule Analysis)

연관분석

대량의 트랜잭션 정보로부터 개별 데이터(변수) 사이에서 연관규칙(x면 y가 발생)을 찾는 것을 말한다. 가령 슈퍼마켓의 구매내역에서 특정 물건의 판매 발생 빈도를 기반으로 'A물건을 구매하는 사람들은 B물건을 구매하는 경향이 있다.'라는 규칙을 찾을 수 있다. 다른 말로 장바구니 분석(Market Basket Analysis)이라 한다.



〈연관규칙분석〉

우유(x)를 구매한 고객이 식빵(y) 또는 계란(y) 또는 휴지(y)를 함께 구매하는 일종의 조건부 확률을 구한다.

연관규칙

조건 결과의 빈도수를 기반으로 표현되기 때문에 비교적 결과를 쉽게 이해할 수 있다. 구매내역의 자료구조를 가지기 때문에 특별한 전처리 과정을 필요로 하지 않는다. 그러나 품목의 개수가 늘어남에 따라 분석에 필요한 계산의 수가 기하급수적으로 증가하는 단점이 있다.

넷플릭스(Netflex)도 연관규칙을 추천 알고리즘에 적용했다. A영화에 대한 시청 결과가 B나 C영화를 선택할 가능성에 얼마나 영향을 미치는지 계산하는 조건부 확률로 콘텐츠 추천의 모델을 만들었다.

신뢰도(Confidence)

항목 A를 포함한 거래 중에서 항목 A와 항목 B가 같이 포함될 확률을 구한다. 즉 우유를 구매했을 때 식빵이 장바구니로 함께 들어갈 확률이 바로 신뢰도인 것이다. 간단한 수식이므로 알아두고 실생활에서 활용해보자.

$$\frac{P(A \cap B)}{P(A)} = \frac{A\text{와 } B\text{가 동시에 포함된 거래 수}}{A\text{를 포함하는 거래 수}}$$

지지도(Support)

전체 거래 중 항목 A와 B를 동시에 포함하는 거래의 비율이다. 장을 본 목록을 확인했을 때 우유와 식빵이 꼭 함께 있을 확률이다.

$$P(A \cap B) = \frac{A\text{와 } B\text{가 동시에 포함된 거래 수}}{\text{전체 거래 수}}$$

향상도(Lift)

A가 주어지지 않은 상태에서 B의 확률에 대하여 A가 주어졌을 때 B의 확률 증가비율이다. 만일 A에 대해 B가 등장할 경우와 A에 대해 C가 등장할 경우의 두 신뢰도가 같다면 어떻게 해야 할까? 대개 이럴 때 향상도 지표가 활용된다. A에 대해 B가 등장 가능성이 높은지 C가 등장 가능성이 더 높은지를 확인하는 지표로 향상도가 있다.

$$\frac{P(B|A)}{P(B)} = \frac{A\text{를 포함하는 거래 중 } B\text{를 포함하는 거래의 수(지지도)}}{\text{전체 거래 중 } B\text{를 포함하는 거래 수(신뢰도)}}$$

분석 연습

마트에서 구매하는 품목을 바탕으로 연관분석의 개념을 이해하기 위한 목적으로 분석 코딩을 실시하였다.
간단한 구매내역 데이터를 직접 작성하여 어떤 항목과 항목이 서로 연관성이 있는지를 확인한다.

1) 연관규칙분석에 필요한 라이브러리와 모듈을 불러온다.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

pandas는 파이썬에서 사용하는 데이터 분석 라이브러리로, 행과 열로 이루어진 데이터 객체를 만들어 사용하며, 대용량의 데이터를 처리하는데 매우 편리한 도구이다.
mlxtend는 통계분석 기능을 지원해주는 파이썬 라이브러리이다.
연관규칙(Association Rule) 적용을 위해서는 각 항목들이 dataset 안에서 어떤 빈도로 나타났는지 또는 어떤 항목과 함께 나왔는지를 파악하는 것이 필수적이다. 하지만 dataset이 큰 경우 이를 모든 항목들에 대해 검사하는 것은 매우 비효율적이다. 이를 해결하기 위해 사용되는 연관규칙분석의 대표적인 알고리즘이 Apriori 알고리즘이다.

2) dataset이 아래와 같이 구성되어 있을 때 연관규칙을 확인해보자.
dataset은 5가지의 장바구니에 담긴 품목들을 나타내고 있다.

```
dataset = [['Milk', 'Onion', 'Nutmeg', 'Eggs', 'Yogurt'],
           ['Onion', 'Nutmeg', 'Eggs', 'Yogurt'],
           ['Milk', 'Apple', 'Eggs'],
           ['Milk', 'Unicorn', 'Corn', 'Yogurt'],
           ['Corn', 'Onion', 'Onion', 'Ice cream', 'Eggs']]

te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)
```

TransactionEncoder 객체를 사용하여 이 dataset을 일반적인 기계 학습에 적합한 배열 형식으로 변환할 수 있다.
fit 함수를 통해 dataset은 고유한 라벨을 갖게 되고, transform 함수를 통해 파이썬 리스트를 원-핫 인코딩(One-hot encoding) 된 numpy 배열로 변환할 수 있다.
원-핫 인코딩이란 어떤 변수의 값을 0과 1로만 표현하는 방식이다.
여기서는 False와 True를 통하여 품목이 없는 경우와 있는 경우를 표현하였다.

3) 원-핫 인코딩의 결과를 중간 과정에서 확인해보자.
0번 바구니부터 4번 바구니까지 총 5개의 장바구니 요약표가 작성됐다.
해당 품목이 있으면 True, 없으면 False이다.

	Apple	Corn	Eggs	Ice cream	Milk	Nutmeg	Onion	Unicorn	Yogurt
0	False	False	True	False	True	True	True	False	True
1	False	False	True	False	False	True	True	False	True
2	True	False	True	False	True	False	False	False	False
3	False	True	False	False	True	False	False	True	True
4	False	True	True	True	False	False	True	False	False

〈One-hot encoding 변환〉

4) 관심항목 대상으로 지지도, 신뢰도, 향상도를 확인한다.
아래 결과표는 Pandas에서 제공하는 association_rules 함수를 사용해 얻은 결과이다.

```
association_rules(frequent_itemsets, metric="lift", min_threshold=1)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Eggs)	(Nutmeg)	0.8	0.4	0.4	0.500000	1.250000	0.08	1.2
1	(Nutmeg)	(Eggs)	0.4	0.8	0.4	1.000000	1.250000	0.08	inf
2	(Eggs)	(Onion)	0.8	0.6	0.6	0.750000	1.250000	0.12	1.6
3	(Onion)	(Eggs)	0.6	0.8	0.6	1.000000	1.250000	0.12	inf
4	(Milk)	(Yogurt)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
5	(Yogurt)	(Milk)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
6	(Onion)	(Nutmeg)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
7	(Nutmeg)	(Onion)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
8	(Nutmeg)	(Yogurt)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
9	(Yogurt)	(Nutmeg)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
10	(Onion)	(Yogurt)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
11	(Yogurt)	(Onion)	0.6	0.6	0.4	0.666667	1.111111	0.04	1.2
12	(Eggs, Onion)	(Nutmeg)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
13	(Eggs, Nutmeg)	(Onion)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
14	(Onion, Nutmeg)	(Eggs)	0.4	0.8	0.4	1.000000	1.250000	0.08	inf
15	(Eggs)	(Onion, Nutmeg)	0.8	0.4	0.4	0.500000	1.250000	0.08	1.2
16	(Onion)	(Eggs, Nutmeg)	0.6	0.4	0.4	0.666667	1.666667	0.16	1.8
17	(Nutmeg)	(Eggs, Onion)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf
18	(Eggs, Nutmeg)	(Yogurt)	0.4	0.6	0.4	1.000000	1.666667	0.16	inf

〈연관규칙분석 결과〉

달걀과 양파 두 품목 모두를 구매할 확률(지지도)은 0.6이고 달걀 품목을 구매했을 때 양파 품목까지 함께 구매할 가능성(신뢰도)은 0.75, 양파를 구매했을 때 달걀을 구매할 가능성은 1(=100%)로 나타났다.

연관분석에 대한 결과 해석은 어느 하나의 지표만 보고 판단하지 않는다. 지지도와 신뢰도, 향상도 외에도 필요한 지표들을 종합하여 결론을 내린다. 여기서는 달걀과 양파가 연관성이 높은 편임을 알 수 있다. 특히 지지도와 신뢰도 그리고 향상도는 위와 같이 품목별 관계성을 직관적으로 보여주므로 품목 간 관계 정도의 서열을 확인하는 데 유용하다.

❷ 교차분석(Cross-tabulation Analysis)

교차분석(Cross-tabulation Analysis)

비교 대상이 되는 항목들의 빈도를 이용하여 자료 간 관계의 유의성을 파악할 때 사용한다. 주로 '범주형' 자료 간의 관계를 확인하는 데 쓰이며, 전체 비율을 통해 예상빈도를 구하여 실제빈도와의 차이를 대조하는 방식이다.

아래는 당뇨 환자 25명과 당뇨가 없는 정상인 75명의 인원 총 100명의 비만 유무를 조사한 결과이다. 100명 가운데 비만(20)과 정상(80) 체중의 구성비가 1:4이므로 당뇨환자군 안에서 비만과 정상의 비율 역시 1:4, 비환자군 안에서도 1:4의 비율로 환자 수가 도출될 것을 예상해 볼 수 있다. 이를 기대빈도(예상빈도)라 한다.

		당뇨환자	비환자(=정상)	전체 (비율)
기대빈도	비만체중	5	15	20 (20%)
	정상체중	20	60	80 (80%)
	전체 (비율)	25 (100%)	75 (100%)	100 (100%)

		당뇨환자	비환자(=정상)	전체 (비율)
관측빈도	비만체중	10	10	20 (20%)
	정상체중	15	65	80 (80%)
	전체 (비율)	25 (100%)	75 (100%)	100 (100%)

〈당뇨와 체중 간 교차표〉

표上은 비만과 정상 체중의 전체 구성비인 1:4에 따라 기대빈도를 설정했고,
표下는 실제빈도를 정리한 교차표(분할표)다.

당연히 실제로 빈도수는 예상과 다를 것이다. 관측빈도 부분을 보면 당뇨환자 25명 중 비만인 사람은 10명, 정상체중인 사람은 15명으로 실제는 2:3의 구성비를 나타냈고, 당뇨병이 없는 정상 범주의 인원 75명 중 비만인 사람은 10명이고 정상체중인 사람은 65명으로 약 1:6.5 비율이 확인됐다.

여기서 한 가지 유념해야 할 내용은 바로 변수의 개수이다. 얼핏 변수의 개수가 4개(2×2)로 보일 수 있지만 여기서 변수는 2개다. 첫 번째 변수는 당뇨환자와 정상인의 범주 정보가 되고, 두 번째 변수는 비만체중과 정상체중의 범주 정보가 되는 것이다. 변수의 개수는 가설검정 시 자유도 선택에 영향을 주므로 확실히 해야 한다.

비만과 체중의 교차분석에 대한 카이제곱 검정

범주형의 자료를 바탕으로 교차분석을 실시했을 때 우리는 '카이제곱(x^2) 검정'을 사용해 분석 결과가 통계적으로 유의미한 것인지 그저 우연인 것인지 확인해 볼 수 있다.

귀무가설 H0	두 변수는 연관성이 없다
대립가설 H1	두 변수는 연관성이 있다

〈비만과 당뇨병 관계 간 가설〉

비만과 당뇨병 간 연관성이 실제로 어떠한지를 검정하기 위하여 가설을 수립했다.

분포표를 확인하면, 자유도가 1일 때 카이제곱 검정 통계량이 3.84보다 작아야 유의수준(P-value)이 0.05이하가 되어 귀무가설을 기각(=대립가설을 채택)하게 된다.

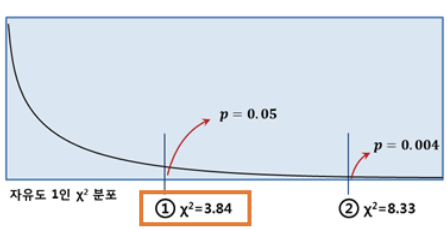
Degree of Freedom	Probability of Exceeding the Critical Value								
	0.99	0.95	0.90	0.75	0.50	0.25	0.10	0.05	0.01
1	0.000	0.004	0.016	0.102	0.455	1.32	2.71	3.84	6.63
2	0.020	0.103	0.211	0.575	1.386	2.77	4.61	5.99	9.21
3	0.115	0.352	0.584	1.212	2.366	4.11	6.25	7.81	11.34
4	0.297	0.711	1.064	1.923	3.357	5.39	7.78	9.49	13.28
.....									
40	22.164	26.509	29.051	33.660	39.335	45.62	51.80	55.76	63.69
50	27.707	34.764	37.689	42.942	49.335	56.33	63.17	67.50	76.15
60	37.485	43.188	46.459	52.294	59.335	66.98	74.40	79.08	88.38
	Not Significant							*Significant*	

〈카이제곱 분포표〉

유의수준이 0.05와 자유도가 1일 때 카이제곱 통계량(값)이 검정의 기준(값)이 된다.

카이제곱 검정 통계량은 아래 계산식에서 8.33이고, 유의수준이 0.004로 확인됐다. 통계량이 거의 0에 가까우므로 앞선 교차분석에 사용된 두 변수 간 유의성은 상당하다고 할 수 있다.

$$x^2 = \sum \frac{(\text{관측빈도} - \text{기대빈도})^2}{\text{기대빈도}}$$
$$= \frac{(5)^2}{5} + \frac{(-5)^2}{20} + \frac{(-5)^2}{15} + \frac{(5)^2}{60} = 8.33$$



〈카이제곱 분포 그래프〉

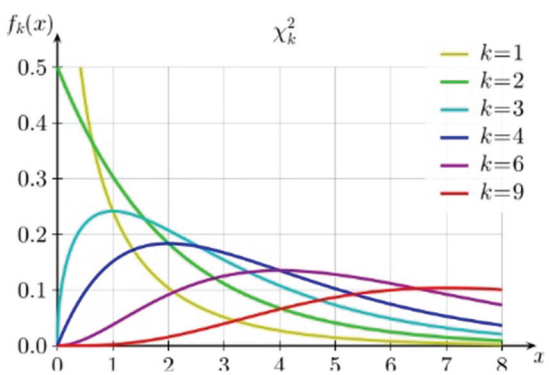
분포표에서 6.63보다 큰 값인 8.33의 위치를 확인하면 우연의 확률이 매우 희박(0.01 보다 작음)하다는 것을 확인할 수 있다.

정리하면, 카이제곱 분포표를 보는 방법은 다음과 같다. 분포표에서 자유도와 유의수준을 기준으로 우리가 확인할 기준값(Threshold)을 찾는다. 본 예제에선 관계를 밝힐 변수의 수가 2개이므로 자유도가 1(=(2-1)*(2-1))이고 유의수준을 0.05로 설정했으므로 기준값은 3.84가 된다.

더불어 카이제곱 검정 통계량은 공식에 따라 8.33이 나왔다. 그러므로 유의수준 0.05에서 귀무가설을 기각할 수 있다고 결론낼 수 있다. 더 나아가 8.33은 6.63의 P-값이 0.01인 경우보다 훨씬 큰 값이다. 유의수준이 거의 0에 수렴한다는 사실을 알 수 있다. 따라서 두 변수는 서로 독립적이지 않고 서로 밀접한 영향을 미친다고 해석할 수 있다.

카이제곱 통계량과 분산의 개념, 결국 평균으로부터의 거리를 구한 것!

앞서 관측빈돗값에서 기대빈돗값을 빼고 제곱한 값을 모두 더한 공식을 통해서 우리는 앞서 배운 분산을 떠올려야 한다(뒤에서 다룰 최소제곱법도 함께 이해하면 좋다). 분산의 공식을 다시 확인하면, '편차 제곱의 합'이다. 해당 수식을 통해서 분산은 데이터의 변동성을 설명했다. 자료의 개수로 평균으로부터의 거리 합을 나누었으니 데이터가 얼마나 퍼져있는지를 뜻하는 것이다.



〈카이제곱 그래프〉
자유도에 따라 그래프의 영역이 달라진다.

같은 방식의 카이제곱 검정 통계량은 어떤가. 기대빈도수를 기준으로 실젯값과 기댓값의 차이를 양수적(제곱)으로 표현하겠다는 의도를 수식 안에 넣어 놓았다. 관측빈도는 실 데이터의 값이고 기대빈도는 전체 비율에 근사하게 얻은 일종의 평균이다. 더불어 관측빈도와 기대빈도의 차이를 양수화(제곱)하고 그것을 평균으로 다시 나눴다. 그 의미는 기대빈도로부터 관측 결과가 얼마나 떨어져있는가(=거리)를 확인하고 있다. 그렇다면 분자의 값이 크면 클수록 거리가 멀다는 의미와 같고 거리가 멀다는 것은 곧 예측치와 관측치의 차이가 크다는 것을 의미한다. 결국 카이제곱 통계량은 교차 확인한 변수들이 서로 얼마나 먼지 또는 가까운지를 설명하는 수치인 셈이다.

분석 연습

슬관절염으로 청구된 연령별 분포를 가지고 교차분석을 실시한다. 분포 연령대는 20년 등간(0~10, 20~30, 40~50, 60~70, 80대~) 구분을 두었다. 2020년 12월 연령대별 전체 청구명세서 건수와 그 가운데 슬관절염이 주·부상병인 청구명세서 건수를 분석 대상으로 하고 있다.

DATASET

2020년 12월, 연령대별 전체 청구명세서 건수
2020년 12월, 연령대별 슬관절염이 주·부상병인 청구명세서 건수

1) 청구명세서 건수를 5개의 연령별 범주로 나누어 각 연령대별로 슬관절염이 얼마만큼의 비중을 갖는지 교차표를 작성한다. 젊은 연령일수록 슬관절염 관련 청구량 비중이 매우 낮다는 사실을 확인할 수 있다.

구분	슬관절염	슬관절염 제외	전체	비중
0~10대	1,672	4,121,352	4,123,024	***1 : 2465***
20~30대	13,999	5,163,671	5,177,670	***1 : 369***
40~50대	181,006	8,486,913	8,667,919	***1 : 47***
60~70대	687,827	7,672,114	8,359,941	***1 : 11***
80대 이상	1,093,949	25,222,785	26,316,734	***1 : 23***
계	1,978,453	50,666,835	52,645,288	***1 : 25***

〈슬관절염 청구명세서 교차표 작성〉
전체 청구명세서에서 슬관절염과 슬관절 제외 건에 대한 전체 비율을 구하면
약 1:25 수준이지만 실제 결과는 연령대별로 상이하다.

2) 전체 청구명세서 가운데 슬관절염의 발병 실제빈도가 **얼마나 유의한 것인지(=우연한 것이 아닌지)** 확인하기 위해 귀무가설과 대립가설을 설정한다.

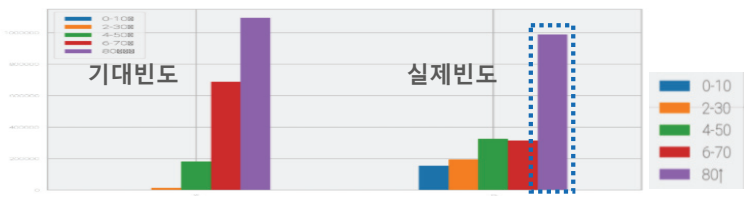
귀무가설	대립가설
슬관절염과 연령은 관련이 없다.	슬관절염과 연령은 관련이 있다.

〈가설검정〉
교차표를 확인한 뒤 위와 같은 연구가설을 세울 수 있다.

3) 전체 청구명세서 중에서 슬관절염으로 청구한 명세서 건수의 비중을 파악한다. 슬관절염 관련 청구빈도를 바탕으로 연령별 기대빈도를 확인한다. 전체 슬관절염 건수를 각 연령마다 곱해 기댓값을 구하였다.

```
DataFrame([ all, obseved, all_per_list, expected]
          , columns = ['0-10', '2-30'  ,'4-50', '6-70' , '80↑']
          , index   = ['전체 청구명세서 건 수','슬관절염 건 수',  '비중' , '기대수준'])
```

	0-10	2-30	4-50	6-70	80↑
전체 청구명세서 건 수	4123024.00	5177670.0	8667919.00	8359941.00	26316734.0
슬관절염 건 수	1672.00	13999.0	181006.00	687827.00	1093949.0
비중	0.08	0.1	0.16	0.16	0.5
기대수준	154947.00	194581.0	325747.00	314173.00	989004.0



〈기대빈도와 실제빈도 확인〉

슬관절염 건수 실제 데이터는 변수(observed)에 연령 구간별로 담겨있다.

4-1) 80대 이상 연령대의 슬관절염 건수 기대수준과 실제빈도가 가장 높다. 해당 값이 얼마나 통계적으로 유의한 값인지 실제 데이터로 카이제곱 검정을 통해 확인한다. (범주형 자료를 분석한 결과에 대한 검정의 기준값은 카이제곱 분포(표)를 참조한다.)빈도를 바탕으로 연령별 기대빈도를 확인한다. 전체 슬관절염 건수를 각 연령마다 곱해 기댓값을 구하였다.

Degree of Freedom	Probability of Exceeding the Critical Value								
	0.99	0.95	0.90	0.75	0.50	0.25	0.10	0.05	0.01
1	0.000	0.004	0.016	0.102	0.455	1.32	2.71	3.84	6.63
2	0.020	0.103	0.211	0.575	1.386	2.77	4.61	5.99	9.21
3	0.115	0.352	0.584	1.212	2.366	4.11	6.25	7.81	11.34
4	0.297	0.711	1.064	1.923	3.357	5.39	7.78	9.49	13.28
5	0.554	1.145	1.610	2.675	4.351	6.63	9.24	11.07	15.09

〈자유도가 4인 카이제곱 분포 기준값(Threshold)〉

9.49라는 기준을 찾는다.

4-2) **자유도 확인** : 자료 범주를 연령대별 5개 구간을 설정하였으므로 변수 x의 수는 5개가 되고 자유도는 여기서 5-1 한 '4'가 된다.

4-3) **기준값 확인** : 자유도 4, 유의수준 0.05를 기준으로 빈돗값의 유의성을 판단할 기준을 찾는다.

4-4) 관측 데이터를 카이제곱 검정 함수에 넣어 반환되는 통계량을 확인한다.

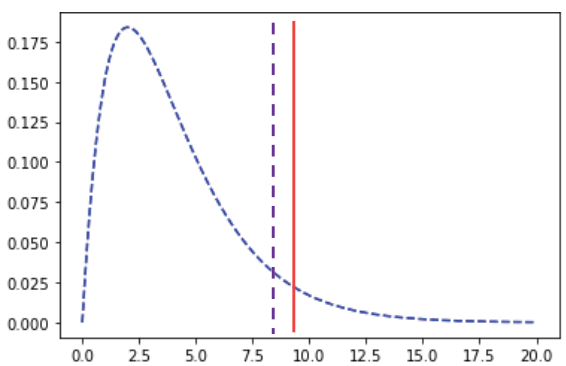
```
from scipy.stats import chisquare
result = chisquare(obseved, f_exp=expected)
result
```

```
Power_divergenceResult(statistic=839056.9559962321, pvalue=0.0)
```

〈통계량 확인〉

슬관절염 건수 실제 데이터(observed)를 카이제곱 검정 함수(chi-square)에 실어 보내
약 839,056(건) 통계량을 반환받은 결과다.

5) 마지막으로 앞서 확인했던 기준값인 9.49 이내로 해당 통계량이 들어오는지 확인한다. 기준을 넘지 않으면 대립가설을 채택하여 슬관절염 청구 건은 연령과 관계가 있다고 결론 내린다.




```
Out[3]: Text(839056.9559962321, 0.4, 'statistic 839056.956')
```

critical value = 9.4877 statistic 839056.956

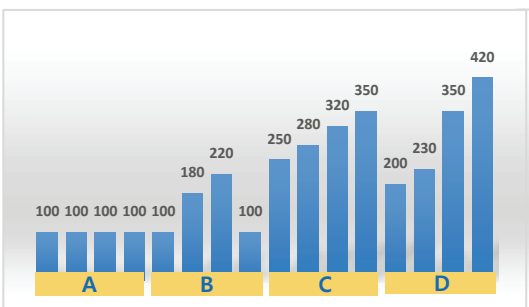
〈자유도 4인 카이제곱 그래프〉

보라색 점선인 통계량 위치가 붉은 기준선 안쪽(좌측)에 분포하므로
통계적으로 유의하다.

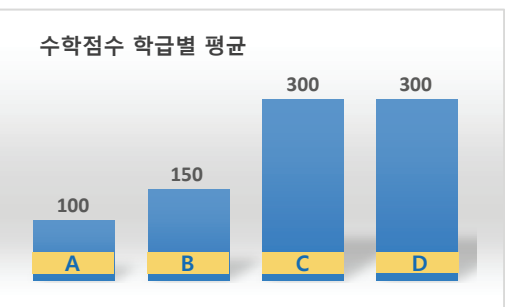
❸ 분산분석(Analysis of Variance, ANOVA)

분산분석(=변량분석)

자료 간의 차이를 대조시키는 분석기법이다. 여기서 분산은 평균을 중심으로 데이터가 얼마나 퍼져있는지를 표현하는 통계량이다. 만약 데이터의 퍼짐이 없고 모든 개별 데이터값들이 동일하다면, 편차가 없으므로 분산은 '0'이 될 것이다.



〈수학 과목 학급 대표 학생별 점수〉
500점 만점



〈수학점수 학급별 평균〉
C학급과 D학급은 평균이 같으나 분산은 크게 다르다.

A학급 경우가 바로 분산이 0이 되는 경우다. 각 대표 학생 4명의 점수가 모두 100점으로 평균이 100이라 편차가 없기 때문이다. 편차가 없다는 의미는 데이터가 좌표상에서 모두 한 점에 찍힌다는 말과 같다.

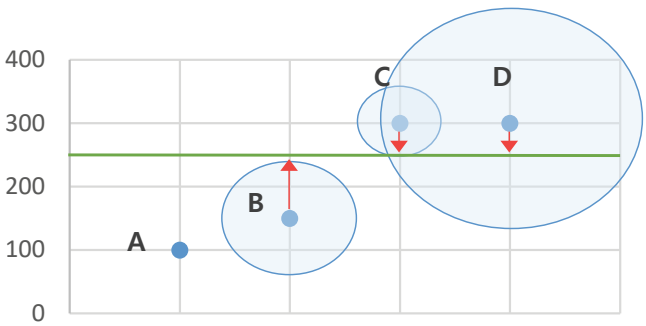
C학급과 D학급은 일단 평균이 같다. 그런데 학생별 점수 데이터를 확인해보면 C학급에 비해 D학급의 대표 학생별 점수 편차가 더 크다는 것을 확인할 수 있다. 실제로 공식에 대입해 편차를 구하고 분산을 계산해보면, C학급은 1,933, D학급은 10,600으로 확연한 차이를 보인다.

$$s^2 = \frac{\sum(X-\bar{X})^2}{n-1}$$

표본분산(s^2)은 관측값에서 표본평균을 빼고 제곱한 값을 모두 더한 후 전체 데이터 개수(n) - 1로 나눠서 구한다.

D학급의 학생별 점수 편차가 커 분산의 차이도 매우 크게 나타났다. 좌표상에서 데이터 간 거리를 표현하기 위해 양수화(=제곱)를 한 것에 영향이다. 분산이란 통계량을 통해서 우리는 C학급과 D학급의 평균이 서로 같다 하더라도 데이터의 분포(형태)는 서로 다르다는 사실을 확인한 것이다.

평균을 중심으로 각 학급별 점수 분포의 대략적인 범위를 표준편차를 지름 삼아 원 그래프로 표현하면 아래 그림과 같다. 평균과는 달리 **분산(표준편차 제곱) 통계량은 C와 D가 명백하게 다른 자료임을 말해준다.**



〈Between Variance〉

각 학급별 평균과 전체학급의 평균 사이의 거리를 구한다.

첫 번째 분산, BV(Between Variance)

BV란 결국 '그룹 간'의 차이로 각 학급의 평균과 학교 전체평균의 거리비율인 것이다. A를 제외하고 B~D 학급 각 평균의 거리(붉은 화살표)를 이용해 세 종류 이상의 그룹 간의 유사성을 분석할 수 있다. 계산은 전체평균-그룹평균의 제곱합을 n-1의 자유도(여기선, 3-1)로 나눈 값이다.

두 번째 분산, WV(Within Variance)와 F-value

WV란 '그룹 내'의 분산을 말한다. 이해를 돕기 위한 예제이므로 검정은 제외하고 F-값을 구하는데 의의를 두자. 계산은 그룹평균-샘플데이터의 제곱을 모두 합하여 전체 데이터 수-그룹 수(두 번째 자유도, 여기선 8-3)로 나눈 값이다. F-값은 BV를 WV로 나누어 얻는다.

$$F = \frac{BV}{WV} = \frac{Mean\ Squared\ Tretment}{Mean\ Squared\ Error}$$

학급	평균(M)	편차제곱×샘플수	샘플 그룹별 분산	F-value
~~A~~	~~100~~	~~0~~	~~0~~	
B	150	60	3,600	
C	300	43.9	1,933	3.10
D	300	102.9	10,600	
전체	전체평균(GM)	분산(WV)	분산(BV)	
A~D	250	9,680	30,000	

〈F-value의 계산〉

A학급을 제외하고 계산한다. 두 가지 평균과 두 가지 분산으로 F-값을 얻는다.
그룹별 분산은 F-값과 무관하다.

자유도에 따라 달리 봐야 할 것, F-분포표

ANOVA는 주로 3종 이상의 자료를 비교할 때 주로 사용된다. 따라서 t-검정과는 달리 F-검정은 비교할 자료의 수(=그룹 수)에 따라 참조할 분포표도 달라진다. 보통 그룹 수에서 −1한 경우의 F-분포표를 참조하면 된다. 아래는 그룹 간 분산(BV)의 자유도가 1일 때 그리고 그룹 내 분산(WV)의 자유도가 2일 때 참조할 분포표이다.

df_2 \ df_1	1	2	3	4	5	6	7	8	9	10	12	15	20	24	30	40	60	120	∞
1	4052	5000	5403	5625	5764	5859	5928	5981	6022	6056	6106	6157	6209	6235	6261	6287	6313	6339	6366
2	98.50	99.00	99.17	99.25	99.30	99.33	99.36	99.37	99.39	99.40	99.42	99.43	99.45	99.46	99.47	99.47	99.48	99.49	99.50
3	34.12	30.82	29.46	28.71	28.24	27.91	27.67	27.49	27.35	27.23	27.05	26.87	26.69	26.60	26.50	26.41	26.32	26.22	26.13
4	21.20	18.00	16.69	15.98	15.52	15.21	14.98	14.80	14.66	14.55	14.37	14.20	14.02	13.93	13.84	13.75	13.65	13.56	13.46
5	16.26	13.27	12.06	11.39	10.97	10.67	10.46	10.29	10.16	10.05	9.89	9.72	9.55	9.47	9.38	9.29	9.20	9.11	9.02
6	13.75	10.92	9.78	9.15	8.75	8.47	8.26	8.10	7.98	7.87	7.72	7.56	7.40	7.31	7.23	7.14	7.06	6.97	6.88
7	12.25	9.55	8.45	7.85	7.46	7.19	6.99	6.84	6.72	6.62	6.47	6.31	6.16	6.07	5.99	5.91	5.82	5.74	5.65
8	11.26	8.65	7.59	7.01	6.63	6.37	6.18	6.03	5.91	5.81	5.67	5.52	5.36	5.28	5.20	5.12	5.03	4.95	4.86
9	10.56	8.02	6.99	6.42	6.06	5.80	5.61	5.47	5.35	5.26	5.11	4.96	4.81	4.73	4.65	4.57	4.48	4.40	4.31
10	10.04	7.56	6.55	5.99	5.64	5.39	5.20	5.06	4.94	4.85	4.71	4.56	4.41	4.33	4.25	4.17	4.08	4.00	3.91
11	9.65	7.21	6.22	5.67	5.32	5.07	4.89	4.74	4.63	4.54	4.40	4.25	4.10	4.02	3.94	3.86	3.78	3.69	3.60
12	9.33	6.93	5.95	5.41	5.06	4.82	4.64	4.50	4.39	4.30	4.16	4.01	3.86	3.78	3.70	3.62	3.54	3.45	3.36
13	9.07	6.70	5.74	5.21	4.86	4.62	4.44	4.30	4.19	4.10	3.96	3.82	3.66	3.59	3.51	3.43	3.34	3.25	3.17
14	8.86	6.51	5.56	5.04	4.69	4.46	4.28	4.14	4.03	3.94	3.80	3.66	3.51	3.43	3.35	3.27	3.18	3.09	3.00
15	8.68	6.36	5.42	4.89	4.56	4.32	4.14	4.00	3.89	3.80	3.67	3.52	3.37	3.29	3.21	3.13	3.05	2.96	2.87
16	8.53	6.23	5.29	4.77	4.44	4.20	4.03	3.89	3.78	3.69	3.55	3.41	3.26	3.18	3.10	3.02	2.93	2.84	2.75
17	8.40	6.11	5.18	4.67	4.34	4.10	3.93	3.79	3.68	3.59	3.46	3.31	3.16	3.08	3.00	2.92	2.83	2.75	2.65
18	8.29	6.01	5.09	4.58	4.25	4.01	3.84	3.71	3.60	3.51	3.37	3.23	3.08	3.00	2.92	2.84	2.75	2.66	2.57
19	8.18	5.93	5.01	4.50	4.17	3.94	3.77	3.63	3.52	3.43	3.30	3.15	3.00	2.92	2.84	2.76	2.67	2.58	2.49
20	8.10	5.85	4.94	4.43	4.10	3.87	3.70	3.56	3.46	3.37	3.23	3.09	2.94	2.86	2.78	2.69	2.61	2.52	2.42
21	8.02	5.78	4.87	4.37	4.04	3.81	3.64	3.51	3.40	3.31	3.17	3.03	2.88	2.80	2.72	2.64	2.55	2.46	2.36
22	7.95	5.72	4.82	4.31	3.99	3.76	3.59	3.45	3.35	3.26	3.12	2.98	2.83	2.75	2.67	2.58	2.50	2.40	2.31
23	7.88	5.66	4.76	4.26	3.94	3.71	3.54	3.41	3.30	3.21	3.07	2.93	2.78	2.70	2.62	2.54	2.45	2.35	2.26
24	7.82	5.61	4.72	4.22	3.90	3.67	3.50	3.36	3.26	3.17	3.03	2.89	2.74	2.66	2.58	2.49	2.40	2.31	2.21
25	7.77	5.57	4.68	4.18	3.85	3.63	3.46	3.32	3.22	3.13	2.99	2.85	2.70	2.62	2.54	2.45	2.36	2.27	2.17
26	7.72	5.53	4.64	4.14	3.82	3.59	3.42	3.29	3.18	3.09	2.96	2.81	2.66	2.58	2.50	2.42	2.33	2.23	2.13
27	7.68	5.49	4.60	4.11	3.78	3.56	3.39	3.26	3.15	3.06	2.93	2.78	2.63	2.55	2.47	2.38	2.29	2.20	2.10
28	7.64	5.45	4.57	4.07	3.75	3.53	3.36	3.23	3.12	3.03	2.90	2.75	2.60	2.52	2.44	2.35	2.26	2.17	2.06
29	7.60	5.42	4.54	4.04	3.73	3.50	3.33	3.20	3.09	3.00	2.87	2.73	2.57	2.49	2.41	2.33	2.23	2.14	2.03
30	7.56	5.39	4.51	4.02	3.70	3.47	3.30	3.17	3.07	2.98	2.84	2.70	2.55	2.47	2.39	2.30	2.21	2.11	2.01
40	7.31	5.18	4.31	3.83	3.51	3.29	3.12	2.99	2.89	2.80	2.66	2.52	2.37	2.29	2.20	2.11	2.02	1.92	1.80
60	7.08	4.98	4.13	3.65	3.34	3.12	2.95	2.82	2.72	2.63	2.50	2.35	2.20	2.12	2.03	1.94	1.84	1.73	1.60
120	6.85	4.79	3.95	3.48	3.17	2.96	2.79	2.66	2.56	2.47	2.34	2.19	2.03	1.95	1.86	1.76	1.66	1.53	1.38
∞	6.63	4.61	3.78	3.32	3.02	2.80	2.64	2.51	2.41	2.32	2.18	2.04	1.88	1.79	1.70	1.59	1.47	1.32	1.00

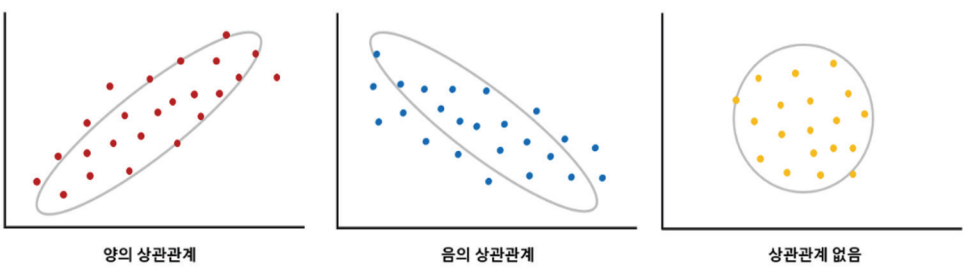
〈F-분포표〉

해당 분포표는 세 그룹을 비교하여 두 가지 분산을 구할 때 사용할 수 있는 분포표이다.
비교하는 그룹이 추가되면 자유도 dfn도 추가된다.

4 상관분석(Correlation Analysis)

상관분석

상관분석은 x와 y변수 간에 관계가 어떤 선형적인 관계를 갖고 있는지를 파악한다. 두 변수 간의 관계의 강도도 계산할 수 있다. 두 변수가 변하는 패턴이 얼마나 비슷한가를 확인하는 과정이 상관분석이다. 상관관계에 따른 산포도는 아래 그림과 같이 나타낼 수 있으며, x축과 y축으로 구성하여 흩어진 정도를 표현할 수 있다.



〈상관관계〉

범주형 자료에 한하여 가로축 x가 커짐에 따라 세로축 y가 증가할 때 양의 상관관계가 있다고 본다. 반대의 경우는 음의 상관관계이다.

공분산(Covariance), X축 분산과 Y축 분산의 평균으로 방향을 알다

상관관계를 표현하는 통계량의 일종으로 x의 분산과 y의 분산을 곱한 것의 기댓값이다. x의 편차와 y의 편차를 서로 곱한 개념임을 수식을 통해 알 수 있다. 만약 첫 번째 항이 (+)이고 두 번째 항이 (+)라면 공분산값은 (+)로 계산된다. 두 항 모두 (−)라도 (+)로 공분산이 나오게 된다. 그러므로 공분산은 그 값이 커질수록 양의 상관관계를 갖는다는 사실을 알 수 있다. 반대로 두 항의 부호가 다르면 그 결과가 (−)이므로 음의 관계를 표현하는 공분산값은 작아진다.

$$\boldsymbol{COV(x, y)=E(X-\mu_x)(Y-\mu_y)}$$

단, 공분산은 x와 y의 구간 범위(=Scale)에 영향을 많이 받는다. 따라서 같은 데이터라도 구간의 크기가 10단위냐 1단위냐에 따라 공분산값은 커지거나 작아질 수 있다. 이때 데이터 산포의 구간 범위를 통일하는 과정을 표준화라고 하며 이 과정에 얻어지는 통계량을 상관계수라 한다.

상관계수(Correlation coefficient), 모상관계수(p) 표본상관계수(r)

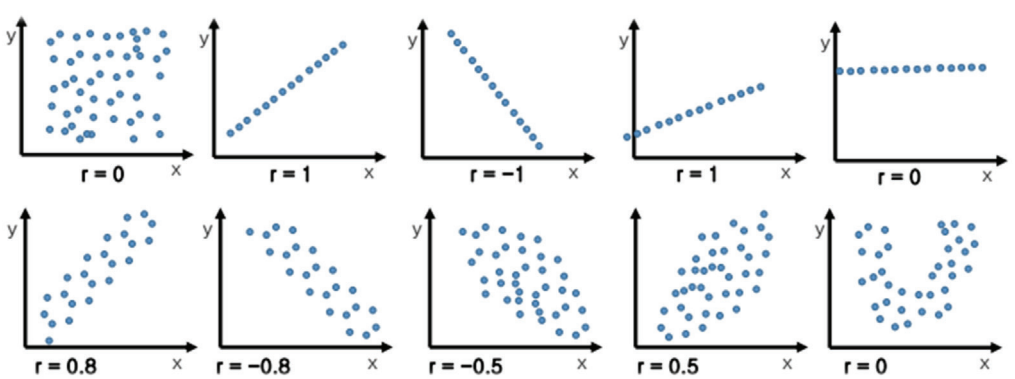
상관계수는 독립/종속변수 간 관계 정도를 확인할 수 있는 값으로 공분산 표준화의 결과이다.

$$p = Corr(x,y) = \frac{E(X-\mu_x)(Y-\mu_y)}{\sqrt{E(X-\mu_x)^2}\sqrt{E(Y-\mu_y)^2}}, \quad r = \frac{\sum(X-\bar{X})(Y-\bar{Y})}{\sqrt{\sum(X-\bar{X})^2(Y-\bar{Y})^2}}$$

상관계수는 확률변수의 절대 크기(=구간)에 영향을 받지 않도록 각 확률변수에 표준편차를 나누어 표준화시킨 것이다. 공분산이 있음에도 불구하고 상관계수가 필요한 이유는 앞서 말했듯 공분산에는 Scale(단위)이 표준화되어 있지 않아서다. 이러한 한계를 극복하기 위해서 표준화를 통한 상관계수 비교가 필요하다. 요컨대, 상관계수는 독립변수 x와 종속변수 y간 연관성을 숫자로 대표한다. 양의 상관관계와 비슷할수록 1로, 음의 상관관계와 비슷할수록 −1로 수치가 표현된다.

상관계수의 범위

상관계수(r)값의 범위는 −1부터 +1까지다. 표준편차의 총합으로 편차의 총합을 나누기 때문에 −1과 +1의 범위로 표현된다. r이 1에 가까울수록 강한 양의 상관관계를 나타내고, −1에 가까울수록 강한 음의 상관관계가 뚜렷해지게 된다. 여기서 상관계수 r은 보통 피어슨 상관계수(Pearson linear correlation coefficient, PLCC)를 의미하는데, 이는 선형적으로 두 변수 간의 관계를 표준화해 설명한 것이다.



〈상관계수에 따른 관계 예시〉

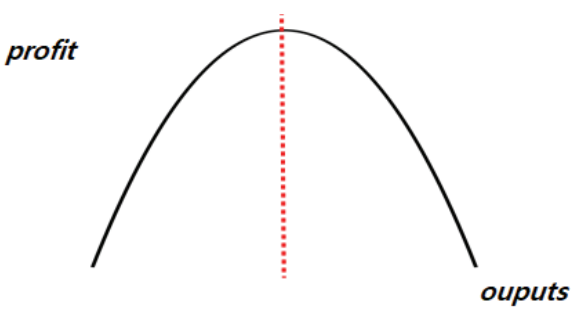
상관계수의 값이 1이면 완전한 양의 관계를 보여주며 −1은 그 반대다.
물론 기울기는 다를 수 있다. 뒤에서 다루겠지만 같은 과정 후 기울기를 찾는 과정이 더해지면 회귀분석이 된다.

상관계수의 활용, '만능계수'가 아님을 유념하자

상관관계를 설명하는 상관계수는 만능으로 적용되어 활용될 수 없다. 먼저 상관계수는 수학적인 관계를 설명하는 지표이다. 양/음의 관계를 설명하는 수치로 활용돼야 확대해석을 피할 수 있다.

가령, 아이스크림 판매량과 강력범죄율 데이터를 가지고 상관관계를 분석한 결과로 상관계수가 0.75라면 '아이스크림 판매가 많아질수록 강력범죄가 증가한다'고 해석할 수 있을까? 이는 상식적으로 보아도 뭔가 이상하다. 사실 여름 기간 동안 높아진 불쾌지수가 강력범죄에 영향을 준 것이다. 그런데 불쾌지수 대신 아이스크림 판매량이 그 자리를 대신하면 이와 같은 엉뚱한 해석을 하게 될 수 있다.

또한 상관계수는 선형의 관계를 설명할 때만 쓰인다. 비선형 관계를 상관분석을 통해 설명할 수는 없다. 예를 들어, 생산량과 이익 간 상관관계를 분석하고자 할 때 아래 그림과 같은 상관관계 곡선이 비선형적으로 그려졌다. 피어슨 상관계수(r)를 구하면 (좌우대칭이므로) 0값이 나오게 될 것이다. 그렇다고 두 변수는 상관관계가 없는 것일까? 그렇지는 않다. 비선형적으로 그 관계성이 명확히 보이고 있다.

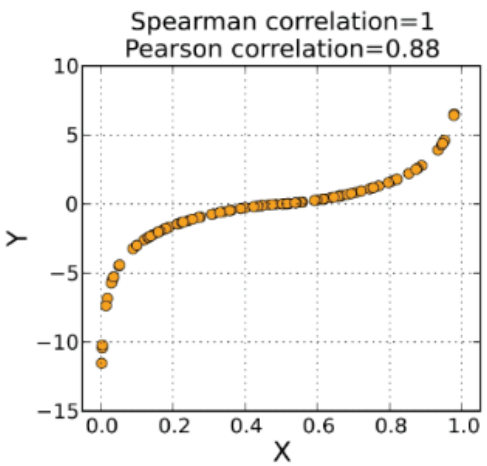


〈비선형 상관관계 예시〉
생산량이 증가할수록 이익이 증가하다가 어느 기점에서
이익이 감소함을 설명하는 그래프 모형이다.

생산량이 증가할수록 이익도 증가하다가 재고관리 비용 증가 등의 이유로 어느 기점에서 이익이 떨어지게 되는 2차 함수적인 관계가 분명히 존재한다. 단순하게 상관계수만으로 두 변수 간의 관계를 결론낸다면 올바른 분석(주성분, 회귀)을 수행하기 어렵다는 점을 알아두어야 한다.

스피어만 상관계수(Spearman rank-order correlation coefficient, SROCC)

변숫값 대신 순위를 이용하여 상관계수를 구하는 경우 사용된다. 데이터를 작은 것부터 차례로 순위를 매겨 서열 순서로 바꾼 뒤 순위를 이용해 상관계수를 구한다. 두 변수 안의 순위가 완전히 일치하면 +1이고, 두 변수의 순위가 완전히 반대이면 −1이 된다.



〈스피어만 그래프 형태〉
계숫값이 1인 경우에 S자 모양이다.
그림출처 : Wiki

구분	척도	특징
Pearson (피어슨)	간격/비율	– 상관분석에서 기본적으로 사용하는 상관계수 – 연속형 변수의 상관관계 측정 (신장, 몸무게) – 모수 검정 (parametric test)
Spearman (스피어만)	서열	– 변숫값 대신 순위를 이용하여 상관계수를 구함 (학교등급, 졸업학위 level) – 비모수 검정 (non-parametric test)

〈두 상관계수의 특징〉
두 상관계수의 범위는 공통적으로 −1~1이다.

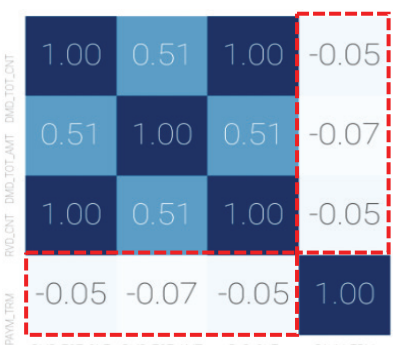
분석 연습

2020년 **A**병원이 청구한 접수내역 4,266건의 데이터 바탕의 각 4가지 항목 간 상관분석을 실시하고 양과 음의 상관관계를 도출한다. 테이블에 있는 미사용 컬럼 제거와 다중공선성 문제 해결까지 데이터전처리 과정도 포함해 진행하였다.

1) 추출한 데이터 세트에서 불필요한 정보(요양기관기호, 지원코드, 처리상태 등)는 제거하는 전처리를 수행하고 청구총금액(DMD_TOT_AMT), 청구총건수(DMD_TOT_CNT), 심사결정건수(RVD_CNT), 지급기간(PAYM_TRM)의 분석 대상 항목을 선정하였다.

2) 각 항목 간의 상관관계를 파악하기 위해서 상관분석을 실시(4×4)하고 동시에 상관계수를 구한다 (상관계수가 0보다 크면 양의 상관관계가 있고 0보다 작으면 음의 상관관계를 갖는다).

```
#상관계수 계산
a = x.corr(method = 'pearson')
print(a)
#HeatMap 시각화
heatmap = sns.heatmap(a, cbar=True, annot=True,
                      annot_kws={'size':16}, fmt='.2f', square=True, cmap='Blues')
```



	DMD_TOT_CNT	DMD_TOT_AMT	RVD_CNT	PAYM_TRM
DMD_TOT_CNT	1.000000	0.512112	0.999998	-0.045381
DMD_TOT_AMT	0.512112	1.000000	0.510707	-0.071655
RVD_CNT	0.999998	0.510707	1.000000	-0.045255
PAYM_TRM	-0.045381	-0.071655	-0.045255	1.000000

3) 붉은색 점선 영역에 표기된 상관계수를 확인하면, 지급기간을 기준으로 나머지 3종류의 계숫값이 −0.1 ~ 0.1 사이로 거의 0에 가까워 관계성이 매우 약하고 동시에 음의 관계성을 갖는다는 것을 확인하였다. 지급기간이 길다고 꼭 청구총금액과 건수 그리고 심결건수가 큰 것은 아님을 확인한 것이다.

4) **다중공선성의 해소** : 서로 다른 독립변수가 종속변수와 추세(또는 분류)선을 공유할 가능성(또는 성질)이 다중공선성이다. 여기서는 청구총건수(x_1)와 심사결정건수(x_2)가 서로 지급기간(y)과 매우 유사한 관계성을 보이므로 둘 중 하나를 제거하지 않으면 다중공선성 문제를 야기할 수 있는 것이다.

산포도 그래프를 그리기 전 다중공선성 분석을 통해 독립변수 중 청구총건수와 심사결정건수가 서로 영향을 강하게 미치고 있음을 알 수 있었다. 이는 둘 중에 하나의 독립변수를 제거하여도 후속 분석 작업으로 주성분분석이나 회귀분석을 수행했을 때 큰 영향을 끼치지 않음을 의미한다. 이번 경우에는 심사결정건수를 제거하였다. 일반적으로 모든 변수의 VIF 수치가 10미만이 되어야 다중공선성 문제가 해결되었다고 해석할 수 있다.

	VIF Factor	features
0	893450.635009	DMD_TOT_CNT
1	-0.045450	DMD_TOT_AMT
2	891640.887013	RVD_CNT
3	1.012362	PAYM_TRM

	VIF Factor	features
0	1.371967	DMD_TOT_CNT
1	-0.013208	DMD_TOT_AMT
2	1.009370	PAYM_TRM

〈VIF Factor〉

다중공선성 문제를 예방하기 위한 사전 조치로 VIF 수치를 확인한다.
수치가 높다면 비슷한 설명력을 가지기 때문에 심결건수(RVD_CNT)를 독립변수에서 제거한다.

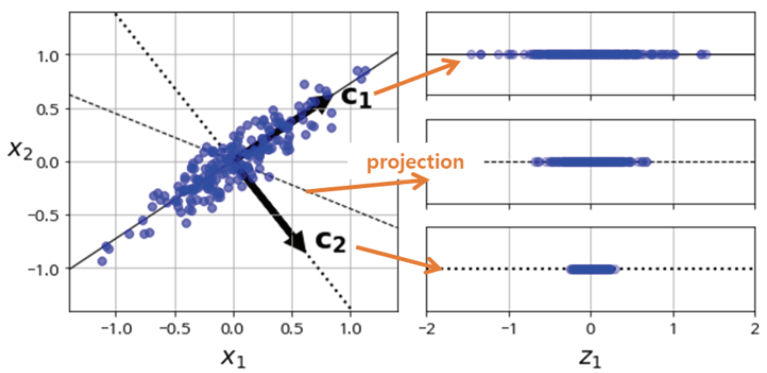
다중공선성의 문제, 다른 관계의 선형이 서로 매우 유사할 때

대개 상관분석은 변수들 간의 상관관계를 선형적으로 설명하기 위한 분석의 중간 과정으로 많이 활용된다. 앞선 분석 연습에서처럼 사실 변수 4종의 상관계숫값을 얻는 과정 중에는 16(=4^2)개의 2차원 그래프로의 변환이 선행된다. 이때 16개의 그래프 가운데 양/음의 관계를 선형으로 설명할 수 있는 그래프가 4종이었다고 한다면, 4개의 선이 서로 닮을 수도 있고 그렇지 않을 수도 있을 것이다. '다중공선성'이란 바로 이 선들이 서로 많이 닮았다는 것을 뜻한다.

5 주성분분석(Principal Component Analysis)

주성분분석

여러 특성(feature)가운데 대표 특성을 찾아 분석하는 방식으로, 대표 특성의 선별은 자료의 차원을 고차원에서 하위 차원으로 축소하는 (차원축소)기법을 활용한다. 차원축소기법에 대한 이해가 주성분 분석의 시작이자 끝이며 여기서는 2차원을 1차원으로 축소하는 범위로 한정해 설명하고자 한다.

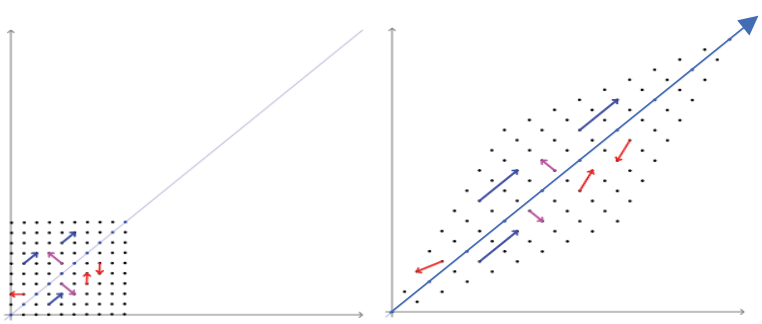


〈차원축소〉

정사영(projection)으로 2차원을 1차원으로 축소한 모습이다. 여기선 C_1이 첫 번째 주성분이다.
그림출처 : Excelsior

분산, 차원축소를 위한 주성분의 선택 기준

위 그림과 같이 차원축소를 위한 정사영의 시작은 무엇을 기준으로 선택되는 것일까? 선택에 따라 데이터의 실제 특성을 보존할 수도 있고 반대로 잃을 수도 있다. C_1을 참고하면 우린 자연스럽게 데이터 간 거리가 가장 큰 쪽이 가장 강력한 데이터 변화 방향이란 사실을 직관적으로 이해할 수 있다. 결국 주성분 선택에 있어 최초로 고려되는 요소는 **분산이 가장 큰 하나의 데이터 선**(2차원으로 축소 시는 면)이 된다.

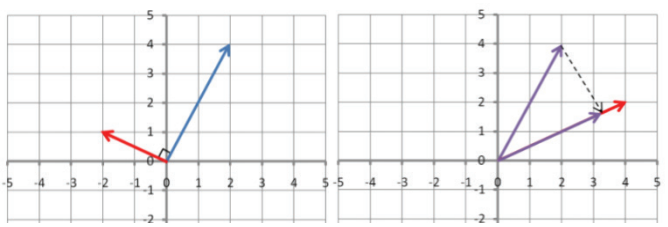


〈가장 센 방향의 선택〉

좌측보다 우측 데이터의 변화 방향(성)이 더 뚜렷함을 확인할 수 있다. 파란색 선을 방향 벡터라 부른다.
그림출처 : Data Science school

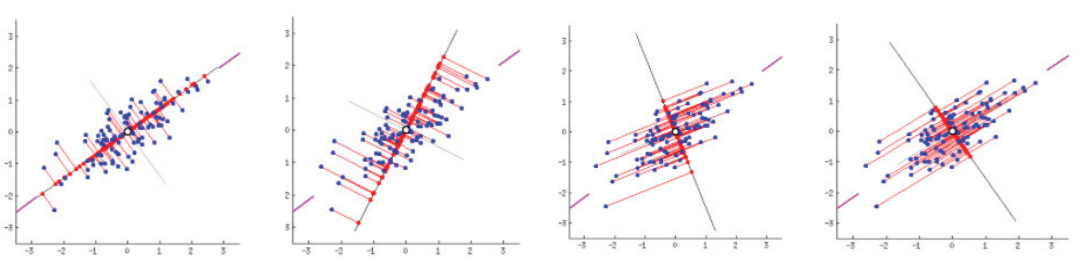
직교(Orthogonality), 그 다음 주성분을 찾는 기준

그렇다면 다음 주성분은 어떻게 찾는 것일까? 두 번째 주성분은 첫 번째 주성분과 '직교'하는 또 하나의 선(또는 면)이다. 다음 주성분은 첫 번째 선택 방향(=방향벡터)과 직교하면서 첫 번째로 분산이 큰 쪽이 선택된다. 두 선이 직교하고 있다면 하나의 선과 다른 하나의 선은 서로 가장 독립적인 상태라고 말할 수 있는 상태다. 이를 내적(Inner Product)이 0인 상태라고 하는데, 좌표상에 두 선이 수직(90°)을 이루며 교차함을 뜻한다.



〈직교의 의미〉
직교는 두 선이 가장 독립적임을 말해준다.
그림출처 : ST Lab

A라는 선형이 180° 회전하게 되더라도 그 방향벡터의 크기와 방향은 동일하기 때문에, 서로 가장 닮지 않는 다른 하나의 선형은 A선형과의 관계가 수직(90°)이다. 가령 x_i값이 증가함에 따라 y_i값이 증가한다면 우리는 그 관계가 양의 상관관계로 '우상향'하는 방향임을 알 수 있다. 반대로 x_i값은 증가하는데 y_i값이 감소하면 '우하향'한다는 사실도 함께 확인 가능하다. 즉, x_i가 고정일 때 y_i값이 완전 반대의 방향을 갖는 경우의 선형의 결과는 결과적으로 90°가 되는 것이다.



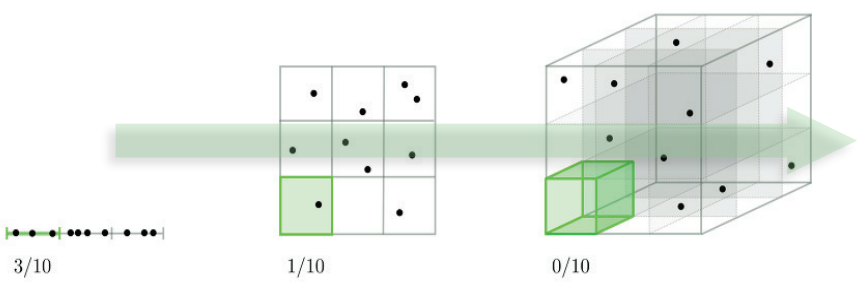
〈주성분을 찾는 과정〉
분산을 기준으로 첫 번째 주성분을 찾아낸다. 분산이 가장 큰 경우가
데이터의 변동 방향을 가장 잘 설명하는 첫 번째 주성분값이 되며
그것에 직교하는 (회색)선이 두 번째 주성분이 된다.

차원축소의 3가지 순기능

우리가 주성분 분석을 사용하는 이유는 데이터가 가진 특성의 수가 지나치게 많을 때, 그 수를 적절하게 줄임으로써 얻는 이점이 있기 때문이다. 특성의 수를 줄일 때 우리는 크게 3가지 순기능을 기대해 볼 수 있다.

먼저 차원이 낮아지면 대상에 대한 이해가 보다 쉬워지게 된다. 공간보다는 면, 면보다는 선, 선보다는 점을 이해하는 것이 보다 용이한 것과 같은 맥락이다. 다음으로 얻을 수 있는 장점은 연산속도가 개선된다는 점이다. 분산값을 유지하면서 정보의 크기 자체를 줄이기 때문에, 데이터의 특성을 훼손시키지 않고도 보다 빠른 연산을 기대할 수 있게 된다.

마지막으로 차원축소는 '차원의 저주'를 해결하는 열쇠가 된다. 선보다는 면, 면보다는 육면체에 데이터가 위치할 공간이 훨씬 크다는 사실은 누구나 쉽게 이해할 수 있을 것이다. 만약 데이터의 양이 동일한 경우에 보다 상위 차원 속에 데이터를 위치시키면 어떨까. 그 결과는 아래 그림처럼 서로 간의 거리가 더욱 멀어진 모습으로 보여질 것이다.



〈차원확대와 데이터 수량과의 관계〉
상위 차원으로 갈수록 면적(부피)당 요구되는 데이터의 수(밀도)가
떨어진다는 사실을 확인할 수 있다.

이런 경우에 발생하는 문제를 차원의 저주라고 한다. 차원 증가에 따라 요구되는 데이터의 양이 기하급수적으로 늘어나기 때문에 우리는 차원축소를 통해 이와 같은 문제를 해결할 수 있다. 높아진 차원을 고려한 데이터 증량이 없다면 우리는 고차원의 데이터를 학습시키는 과정에서 '과적합(Overfitting)'의 문제를 겪을 것이다. 차원축소는 이처럼 데이터가 부족한 상태에서 과적합을 예방하는 전처리 기법으로 가능하게 된다.

분석 연습

위스콘신 암센터 569명의 유방암 진료 환자 샘플 데이터를 PCA에 활용하였다. 암 진단에 필요한 속성 정보와 양성(Benign)/악성(malignant) 진단 결과로 분류된 환자 데이터이다. 해당 데이터 세트를 검토하여 유방암 진단에 설명력이 높은 주성분들을 찾아본다.

1) Breast Cancer Wisconsin Diagnostic 데이터 세트 구성의 총 속성을 확인한다.

```
from sklearn.datasets import load_breast_cancer
import numpy as np

cancer = load_breast_cancer()
print(cancer.DESCR)
```

```
:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
    - radius (mean):                        Min    Max
    - texture (mean):                       ======  ======
    - perimeter (mean):                     6.981   28.11
    - area (mean):                          9.71    39.28
    - smoothness (mean):                    43.79   188.5
    - compactness (mean):                   143.5   2501.0
    - concavity (mean):                     0.053   0.163
    - concave points (mean):                0.019   0.345
    - symmetry (mean):                      0.0     0.427
    - fractal dimension (mean):             0.0     0.201
    - radius (standard error):              0.106   0.304
    - texture (standard error):             0.05    0.097
                                            0.112   2.873
                                            0.36    4.885
                                                .....
                                            fractal dimension (standard error):  0.001  0.03
                                            radius (worst):                      7.93   36.04
                                            texture (worst):                     12.02  49.54
                                            perimeter (worst):                   50.41  251.2
                                            area (worst):                        185.2  4254.0
                                            smoothness (worst):                  0.071  0.223
                                            compactness (worst):                 0.027  1.058
                                            concavity (worst):                   0.0    1.252
                                            concave points (worst):              0.0    0.291
                                            symmetry (worst):                    0.156  0.664
                                            fractal dimension (worst):           0.055  0.208
                                            ====================================== ====== ======
```

〈데이터 세트 요약통계 확인〉

10가지 속성 정보를 mean, standard error, worst 3종으로 구분해 계산된 내용이다

2) 569명의 환자를 기준으로 속성 정보를 표기하는 데이터 프레임을 작성하고 데이터값들을 확인한다.

```
df = pd.DataFrame(cancer.data, columns=cancer.feature_names)
sy = pd.Series(cancer.target, dtype='category')
cancer.target_names
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
0	17.99	10.38	122.8	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...
1	20.57	17.77	132.9	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...
2	19.69	21.25	130.0	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...

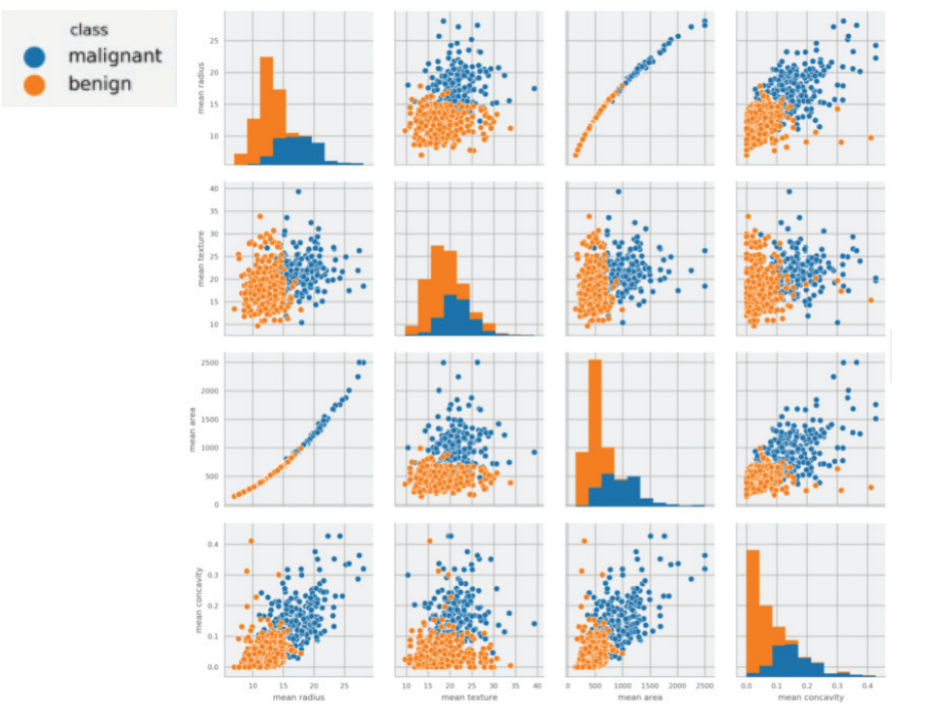
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...

〈전체 데이터 확인〉

암 진단 속성 정보를 정렬하여 표 형태로 확인하였다.

3) 양성/악성 정보 바탕의 진단 결과를 속성별 관계로 플로팅하여 양성 진단과 악성 진단에 대한 환자 분포의 분명한 차이를 시각적으로 확인한다. 시각화된 그래프만을 가지고도 양성인지 악성인지를 판단할 때, 각 속성 값이 분명한 차이를 보였다는 사실을 알 수 있다.

```
> sns.pairplot(vars=[  "mean radius", "mean texture"
                     , "mean area", "mean concavity"]
                  , hue="class", data=df)
```



〈악성과 양성 환자 분포〉

속성 간 교차 시각화 결과이다. 악성 종양 진단 환자와 양성 환자의 극명히 양분된 분포 형태를 확인한다. 여기서는 4가지 속성만을 사용해 플로팅(4×4)했다.

4) 다음으로 데이터 표준화 또는 정규화 작업이 필요하다. 속성마다 데이터값의 범위가 다르므로 이를 주성분 선별 전에 전처리해야 각 속성의 영향도를 같은 선상에서 비교할 수 있다. 여기서 표준화는 사이킷런(scikit-learn) 전처리 모듈 가운데 Standard Scaler함수를 사용해 수행했다.

4-1) 사이킷런(scikit-learn) 라이브러리 PCA모듈을 활용해 공분산(Covariance)과 상관계수를 구한다 (※ 상관분석 '공분산' 참고).

```
> pca = PCA(n_components=2)
> pca.fit(x_scaled)
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
> x_pca = pca.fit_transform(x_scaled)
> pca.explained_variance_ratio_
array([0.44272026, 0.18971182])
> a= pca.explained_variance_ratio_
```

```
> pca = PCA(n_components=30)
> pca.fit(x_scaled)
PCA(copy=True, iterated_power='auto', n_components=30, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
> x_pca = pca.fit_transform(x_scaled)
> pca.explained_variance_ratio_
array([4.42720256e-01, 1.89711820e-01, 9.39316326e-02, 6.60213492e-02,
       5.49576849e-02, 4.02452204e-02, 2.25073371e-02, 1.58872380e-02,
       1.38964937e-02, 1.16897819e-02, 9.79718988e-03, 8.70537901e-03,
       8.04524987e-03, 5.23365745e-03, 3.13783217e-03, 2.66209337e-03,
       1.97996793e-03, 1.75395945e-03, 1.64925306e-03, 1.03864675e-03,
       9.99096464e-04, 9.14646751e-04, 8.11361259e-04, 6.01833567e-04,
       5.16042379e-04, 2.72587995e-04, 2.30015463e-04, 5.29779290e-05,
       2.49601032e-05, 4.43482743e-06])
```

〈PCA함수를 통한 차원축소〉

n-components 설정에 따라 산출된 계수의 수가 달라진다.
유방암 진단을 위한 속성 서른 가지를 차원값으로 놓았다.

PCA함수에서 n_component가 설정하는 정수가 차원을 설정하는 값이다. 앞 페이지 〈악성과 양성 환자 분포〉그림에서 확인했듯, 두 속성별 관계를 2차원 환자 산점도로 표현할 수 있었다. 속성 하나가 곧 하나의 차원이며 변수이므로, 여기서는 차원설정값을 30으로 설정하여 PCA함수에서 서른 가지 특성에 대한 공분산을 반환받은 것이다.

데이터 표준화와 정규화의 차이

데이터전처리를 하기 위해 흔히 표준화와 정규화라는 방법을 많이 사용하게 된다. 혼동하기 쉬운 용어이나 의미와 사용목적의 차이가 있으므로 적절한 전처리 방법을 선택해 분석해야 한다.

구분	표준화(Standardization)	정규화(Normalization)
공식	$z = \frac{x-\bar{x}}{\sigma}$ (분모가 표준편차)	$x_{new} = \frac{x-x_{min}}{x_{max}-x_{min}}$ (분모가 최댓값)
설명	데이터가 평균으로부터 얼마나 떨어져 있는지 나타내는 값으로 특별히 큰 값들은 이상치(outlier)로 해석한다.	데이터의 상대적 크기에 대해 영향을 일반화 시켜놓은 것. 값의 범위가 0~1로 변환된다.

4-2) 각 속성별 Scale을 통일하는 표준화 과정을 수행한다. transform함수로 데이터 전체의 표준화를 진행했다(※ 상관분석 '상관계수' 참고).

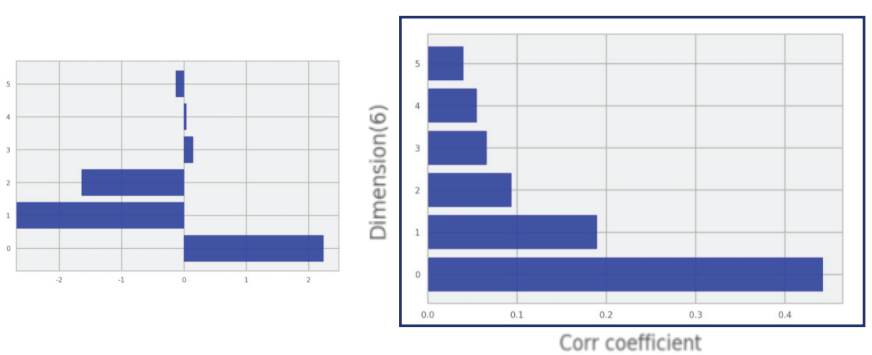
```
scaler = StandardScaler()
scaler.fit(cancer.data)
x_scaled = scaler.transform(cancer.data)
df = pd.DataFrame(x_scaled, columns=cancer.feature_names)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588	0.105777	-0.809117	-0.895587	...
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944	2.658866	2.137194	1.043695	...
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873	-1.261820	-0.820070	-0.561032	...

〈데이터전처리〉

암 진단 569명 환자별 속성값들을 표준화하였다.
-2.0733은 앞에서 초기 전체 데이터를 확인했을 때 10.38이었다.

4-3) 전체 데이터를 표준화했으니 주성분으로써 기능을 할 수 있는 속성 간 비교가 가능하다. 상관계수의 순위 막대그래프는 6가지 주성분을 시각화한 Bar차트이다. 결과적으로 30개 변수 중 상관관계가 높은 변수끼리 선형 결합을 했을 때 전체 변화를 설명하는 데 유력한 변수가 6가지란 의미이다.



〈양성 진단의 6가지 주성분〉

암 진단에 유력한 주성분 6가지를 찾았다.
좌측은 6번 환자에 대한 동일한 성분의 영향력이다.

PCA분석을 수행하기 전에 상관계수를 통한 데이터 표준화가 필요하다는 사실 이해와 더불어 전처리 이후 가장 계숫값이 높은 속성별로 서열을 확인하여 환자 양성 및 악성 진단에 대한 영향력이 큰 최적의 차원 수를 확인하였다.

주성분분석과 혼동하지 말자, 요인분석(Factor Analysis)

결과에 대한 결정적인 인자(원인요소)를 찾아내는 방법이다. 주성분분석과 차이를 대조하여 이해하는 것이 좋은데 그 차이를 이해하기 위해서 먼저 분산을 다시 한번 확인한다.

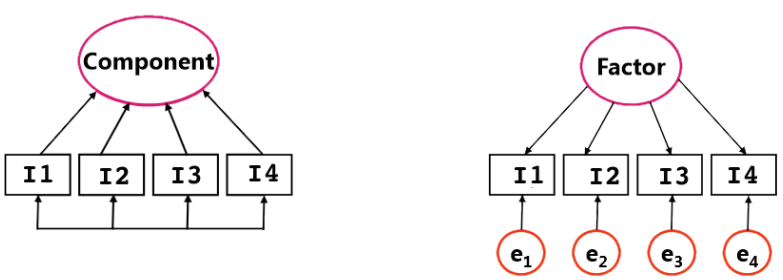
〈분산의 종류〉

1장에서 분산은 데이터의 퍼짐을 나타내는 수치로 배웠고 2장 분산분석에서 분산은 데이터를 분포 측면에서 특성을 말해준다고 배웠다. 이번에는 분산을 좀 더 세부적으로 나누어 알아보고 그 차이를 확인한다.

주성분(Principle Components)과 요인(Factor)의 차이

주성분분석에서 차원축소를 위해 사용한 분산은 공분산이었다. 여러 변수 간의 상관관계를 공통의 분산으로 밝히고 일정한 기준(=상관계수)으로 우선순위를 설정하는 과정, 즉 종속변수 y에 영향도가 높은 1개 이상의 독립변수 x_n를 찾는 것이 바로 주성분분석이다.($x_1 \cdots x_n$의 총합은 1이다. 여기서 1은 완전한 설명력으로 이해해도 좋다.)

이와 달리 요인분석에서는 Factor를 찾기 위해서 고유분산을 이용한다. 예를 들어 국어, 영어, 수학 점수를 통해 '언어능력'을 측정한다는 가정에서, 각 점수에 영향을 끼친 고유한 원인을 찾는 과정이 요인분석이다. 각 변수마다 독립적인 특성을 확인해야 하므로 고유분산을 사용한다.



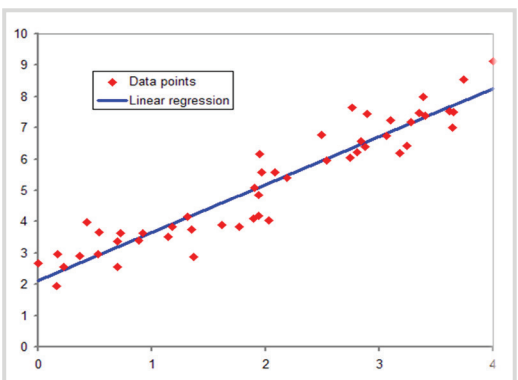
〈성분과 요인의 차이〉

각 변수가 하나의 종속변수를 바라보는 형태가 주성분분석,
변수마다 영향을 미치는 고유한 값을 찾는 형태가 요인분석이다.

6 회귀분석(Regression Analysis)

회귀분석

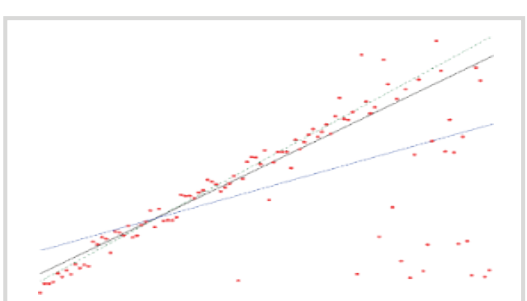
일반적으로 예측을 목표하는 통계 분석이다. 예측을 하는 방법에 핵심이 되는 개념이 바로 '추세선'이다. 좌표상에서 데이터의 분포와 앞으로의 변화를 가장 잘 설명할 수 있는 하나의 선을 그려내는 것이 회귀분석의 궁극적인 목적이 된다.



〈직선의 추세선〉
데이터 산포를 통해 가장 예측력이 뛰어난 선을 그린다.
출처 : Wiki

추세선의 의미

직선의 추세선을 수식으로 표현하면, y=ax+b와 같은 1차 방정식이 된다. 여기서 x와 y는 이미 알고 있는 데이터값이다. 여기서 수식을 활용해 데이터 변화의 추세를 확인하는 방법은 x와 y에 들어오는 값을 기준으로 해당 수식을 충족시키는 a(=기울기), b(=절편)값을 찾는 것이다. 즉 추세선을 통한 회귀적 예측이란 곧 a, b를 구하는 과정을 말하는 것이다. 여기서 a와 b를 '회귀계수'라 한다.



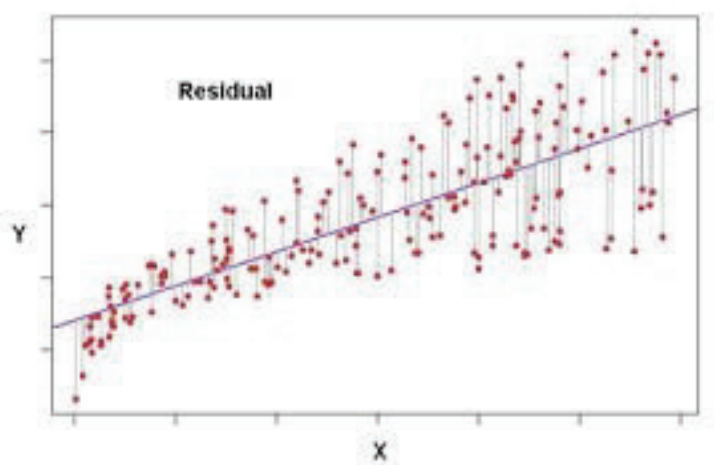
〈이상치와 추세선〉
이상치를 제외하면 최적의 회귀계수를 찾아
더욱 높은 예측력을 가지는 회귀선을 그릴 수 있다.

회귀의 시작, 최소제곱법(=최소자승법 Ordinary Least Squares)

그렇다면 최적의 회귀계수 a와 b를 어떻게 찾을 수 있을까. 이때 사용되는 개념이 최소제곱법이다. 최초엔 임의의 추세선(=가설초기화)을 기준으로 분포한 x, y의 좌푯값의 차이(=잔차)를 제곱하여 모두 더한 값이 최소가 되는 지점들을 연결하는 방법이다.

잔차가 최소인 부분을 찾는 이유는 잔차가 곧 '오차'를 의미하기 때문이다. 추세선을 기준으로 실제 분포한 좌푯값과의 거리가 크다면 오차가 큰 것이다. 오차는 추세선보다 클 수(+)도 있고 작을 수(−)도 있다. 값의 크기와 관계없이 차이에 대한 절댓값(=거리)을 필요로 하므로 오차마다 제곱을 하게 되고 그 값들의 합이 최소가 될 때 비로소 추세선과 데이터값과의 오차의 합이 가장 작아지게 된다.

따라서 OLS는 잔차들의 제곱합 값이 가장 작은 값의 선분을 찾아감으로써 평균으로 '회귀'하는 성질을 가지며, 우리는 이를 활용해 회귀적인 분석(예측)을 할 수 있다.

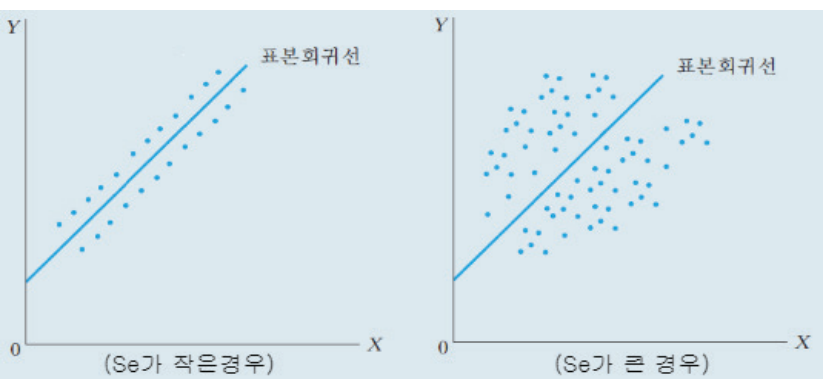


〈잔차와 추세선〉
잔차란 곧 데이터값과 추세선과의 오차이고 각 오차들의 제곱합이 최소가 될 때 비로소 최적의 추세선을 찾을 수 있다.

추세선은 얼마나 정확한 것일까, 표준오차와 회귀계수

데이터가 다르더라도 동일한 방정식과 회귀계수를 가질 수 있다. 이런 경우 표준오차의 차이를 확인하여 구분해 볼 수 있다. 표준오차란 회귀선(=추세선)과 데이터 간 차이의 표준값이다. 표준편차는 데이터와 평균 간 표준 거리 차였고, 표준오차는 회귀(직)선을 기준으로 데이터가 얼마나 잘 모여있거나 퍼져있는지를 표현한 개념이다. 따라서, 아래 그림과 같이 표준오차(SE)가 작다면 회귀선 주변에 가깝게 데이터가 분포하고 있음을 뜻하고, 반대라면 회귀선으로부터 데이터가 멀리 퍼져있음을 의미한다.

정리하면, 회귀선을 기준으로 데이터가 잘 모여 있다는 말은 다른 말로 해당 추세선이 데이터를 잘 설명해 준다는 말이며 이는 곧 표준오차가 작다는 말과 동치이다. 그러므로 회귀계수(y=ax+b)를 구했을 때 표준오차가 큰 것보다 작은 것이 회귀계수의 설명력을 높이게 된다(=회귀계수의 우연성(p-value)을 떨어뜨린다). a와 b값이 정확할수록 x에 대한 그 다음 y값을 보다 더 정확하게 예측해낼 수 있는 것처럼 말이다. 결국 회귀선을 기준으로 데이터의 분포가 조밀할 때, 우리는 표준오차는 작으면서 회귀계수와 추세선은 보다 정확할 것이라고 생각할 수 있다.



〈표준오차와 추세선〉

표준오차(SE)가 크면 y값을 추정하는 회귀계수(=추세선)의 우연정도(P-value)가 커지므로 해당 추세선(=회귀식)의 설명력은 떨어질 수밖에 없다.

추세선은 얼마나 정확한 것일까, 회귀계수와 t-value

회귀분석은 기본적으로 독립변수와 종속변수 사이 양 또는 음의 상관관계가 전제돼야 한다. 그런데 만약 두 변수 간의 관계가 없다면, 다시 말해 기울기가 '0'이라면 애써 회귀분석을 실시한다 하더라도 아무런 의미가 없다. 회귀계수를 구하고 표준오차가 상대적으로 작다 할지라도 기울기(≒변화)가 없다면, 예측할 근거가 전혀 없는 것과 같기 때문이다.

이쯤에서 다시 한번 상기해야 할 점은 회귀선(=추세선)이 평균을 지나간다는 사실이다. 평균을 지나는 추세선의 기울기를 검정하는 방법으로 t-test를 활용하는데, 우린 1장에서 t-test는 두 자료의 차이를 평균과 표준편차의 비교로 설명한다는 내용을 배웠다. 그 가운데 데이터의 변동성은 표준편차(분산 참고)를 통해 설명 가능하므로 데이터 변화 정도를 t-분포표를 참조하여 t-value를 확인(독립변수가 1개이므로 자유도는 '1'로 설정)하면 기울기가 '0'이 아닌 것을 확인함과 동시에 두 변수 관계 정도까지 확인할 수 있다.

회귀분석 결과표의 이해

수행한 회귀분석이 유의한 것이었는지 확인할 수 있는 분석 결과표다. 앞서 살펴보았듯 표준오차와 p값은 작을수록 추세선을 결정하는 회귀계수의 우연성이 낮아(=예측력이 높아)지며, t값은 커질수록 두 변수 간의 관계가 유의미하다는 것을 말해준다.

Model Coefficients

Predictor	Estimate	SE	t	p
Intercept	−43581	4402.69	−9.90	< .001
sqft_living	281	1.94	144.92	< .001

계수[a]

모형		비표준화 계수		표준화 계수		
		B	표준오차	베타	t	유의확률
1	(상수)	-43580.743	4402.690		-9.899	.000
	sqft_living	280.624	1.936	.702	144.920	.000

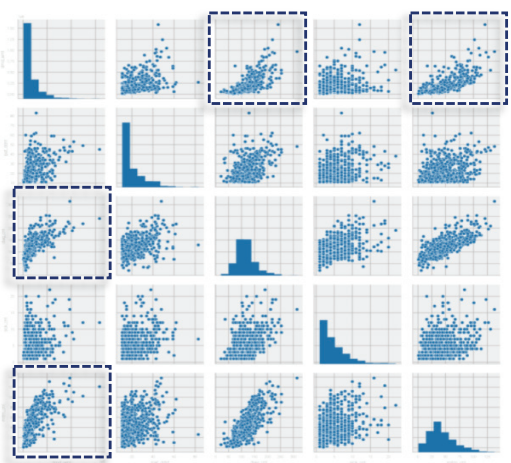
〈분석결과표〉

(좌)JAMOVI와 (우)SPSS를 통해 회귀분석의 결과(회귀계수)를 표 형태로 확인할 수 있다.
여기서 t값은 표준오차(SE)로 회귀계수(Estimate)를 나눈 값이다.

분석 연습

회귀분석은 보통 상관분석부터 시작된다. 상관관계를 계수로 확인하고 계숫값이 선형을 뜻할 때(1 또는 −1에 가까울 때) 회귀분석이 의미가 있다. 여기서는 청구금액(DMD_AMT)을 종속변수(y)로 설정하여 진료건수(DIAG_CNT), 입원일수(IPAT_DDNT), 상병건수(SICK_CNT), 약품목수(MITM_CNT)의 4가지 독립변수 후보 간 관계를 이해하고 회귀분석 시 가장 설명력이 높은 변수를 찾는다.

1) 771건의 샘플 데이터를 전처리한 뒤 선형의 관계를 찾기 위한 첫 번째 과정으로 그래프 대칭행렬의 시각화를 실시하여 분포를 확인한다. 점선 박스 처리된 그래프를 통해 선형적으로 설명될 수 있는 변수를 대략적으로 확인 가능하다.

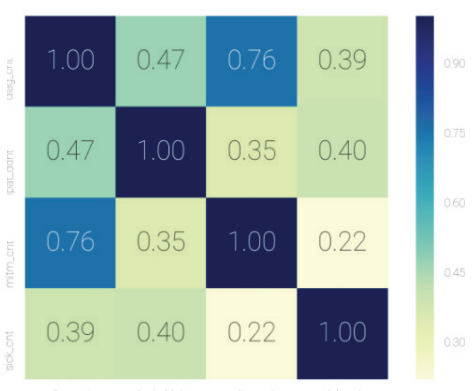


〈독립변수 후보 변수의 Pair-plot(5×5)〉

청구금액(y) 변화에 가장 큰 영향을 주는 독립변수가 무엇인지 찾는다.

2) 어떤 그래프가 선형적으로 관계 설명이 원활할지 그렇지 않은지를 독립변수 간 상관계수를 통해 더 정확히 알 수 있다.

```
> a = x.corr(method = 'pearson')
> print(a)
            diag_cnt  ipat_ddnt  mitm_cnt  sick_cnt
diag_cnt    1.000000   0.469707  0.758451  0.389193
ipat_ddnt   0.469707   1.000000  0.352814  0.400782
mitm_cnt    0.758451   0.352814  1.000000  0.218686
sick_cnt    0.389193   0.400782  0.218686  1.000000
```



〈독립변수 간 상관계수(4×4)〉

가장 양의 상관관계가 뚜렷한 두 변수는 약품목수와 진료건수로 상관계수가 0.76이다.

3) Statsmodels에서 OLS함수를 실행하여 데이터 요약을 확인한다. 요약표 R-squared, P값을 통해 모델의 예측력과 유의수준을 확인한다. 상병건수가 유의수준이 0.6을 넘어가므로 모집단과 표본 간 평균이 다를 수 있다. 따라서 해당 변수는 제거해야 한다.

```
# 최소자승법(Ordinary Least Squares)
# 잔차제곱합(Residual Sum of Squares)을 최소로
Model = sm.OLS(y, x)
fittedModel = Model.fit()
fittedModel.summary()
print(fittedModel.summary())
```

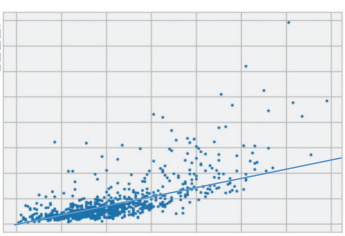
```
                                 OLS Regression Results
================================================================================================
Dep. Variable:                 dmd_amt   R-squared (uncentered):                   0.769
Model:                             OLS   Adj. R-squared (uncentered):              0.768
Method:                  Least Squares   F-statistic:                              639.9
Date:                 Thu, 15 Jul 2021   Prob (F-statistic):                    1.25e-242
Time:                         01:26:43   Log-Likelihood:                         -13637.
No. Observations:                 771   AIC:                                    2.728e+04
Df Residuals:                     767   BIC:                                    2.730e+04
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
diag_cnt   -3.948e+04   1.44e+04     -2.744      0.006   -6.77e+04   -1.12e+04
ipat_ddnt   3.379e+05    4.7e+04      7.187      0.000    2.46e+05    4.3e+05
mitm_cnt    3.88e+05   2.63e+04     14.742      0.000    3.36e+05    4.4e+05
sick_cnt   -7.162e+04   1.39e+05     -0.516      0.606   -3.44e+05    2.01e+05
==============================================================================
Omnibus:                      580.266   Durbin-Watson:                   1.594
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11024.602
Skew:                           3.248   Prob(JB):                         0.00
Kurtosis:                      20.349   Cond. No.                         44.0
==============================================================================
```

R^2(결정계수) 선형회귀 추세선의 설명력을 표현한다.

우연성이 큰 변수임으로 분석 대상에서 제외한다.

4) 상병건수를 제외한 나머지 세 변수와 청구금액과의 관계를 산점도와 함께 OLS 결과표로 확인한다. 3가지 변수 가운데 약품건수가 청구금액과의 상관계수를 비롯하여 추세선의 신뢰도(=설명력)가 가장 높아 청구금액의 크기에 가장 영향력 높은 변수로 확인됐다.

```
==============================================================================
Dep. Variable:                 dmd_amt   R-squared:                       0.504
Model:                             OLS   Adj. R-squared:                  0.504
Method:                  Least Squares   F-statistic:                     782.5

==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept  -4.513e+06   8.92e+05     -5.059      0.000   -6.26e+06   -2.76e+06
mitm_cnt    5.024e+05    1.8e+04     27.973      0.000    4.67e+05    5.38e+05
==============================================================================
```



〈약품건수와 청구금액〉

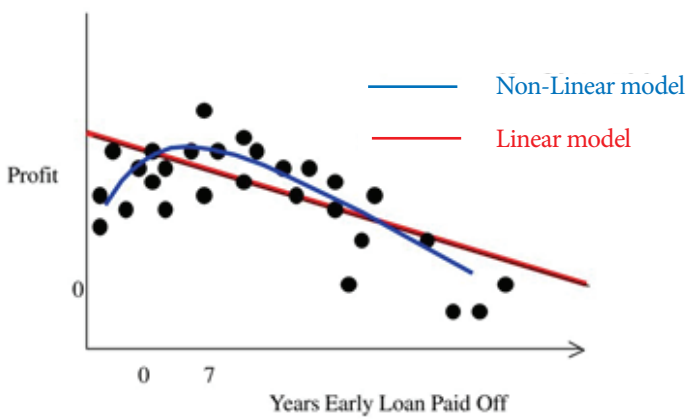
약품건수와의 청구금액(y)과의 상관관계와 설명력(0.50) 모두 가장 높았다.
(입원일수: 0.27, 진료건수: 0.48)

비선형회귀분석(Non-linear regression)

비선형회귀란 직선의 회귀선을 곡선으로 변환해 보다 더 정확하게 데이터 변화를 예측하는 데 목적이 있다. 현실에서의 데이터는 대개 선형보다 비선형적으로 더 상세하게 표현될 수 밖에 없다. 아래 그림에서 붉은색 선은 앞서 보았던 OLS방식으로 데이터 산포를 회귀직선으로 설명한 것인데, 직선의 회귀선으로 y축의 이익이 감소 추세라는 사실은 설명할 수 있지만, 더욱 정확하게 데이터 변화를 예측하긴 어렵다.

그 이유는 직선의 회귀선이 곡선에 비해 '잔차'가 더 크기 때문이다. 잔차가 크다는 것은 실측과 추세선 사이의 표준오차가 크다는 의미를 내포하기 때문에 보다 정밀한 예측에 한계가 있다. 앞에서 우리는 표준오차가 클수록 회귀계수의 우연성이 크다고 배웠다. 비선형 회귀식은 상대적으로 더 작은 잔차를 이용해 데이터 변화의 추세를 더욱 정교하게 설명할 수 있게 된다.

비선형은 현실과 맞닿아 있는 만큼 그 종류와 활용 케이스가 매우 다양하다. 따라서 본 책에서는 여러 비선형 모델 가운데 분류에 주로 사용되는 로지스틱 회귀분석을 비선형 기법을 대표해 살펴보고자 한다.



〈두 가지 회귀선〉

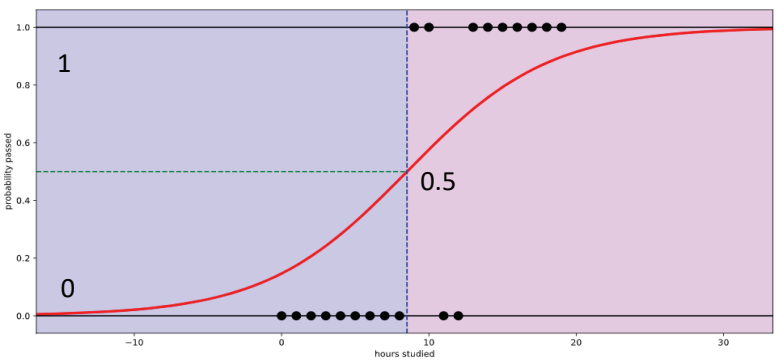
두 선 가운데 데이터의 추세를 보다 더 정확하게 설명하는 선은 무엇일까. 비선형 회귀선을 찾는 모델은 보다 복잡한 패턴을 가지고 있는 자료에 대하여 비교적 정확한 모델링을 할 수 있다는 장점이 있다. 선형 모델에 비해 잔차(실측치-예측치)가 더 작은 형태의 모델을 찾는 것이 가능하기 때문이다.

7 로지스틱 회귀분석(Logistic Regression Analysis)

로지스틱 회귀

데이터가 어떤 범주에 속할 확률을 0에서 1 사이의 값으로 예측하고, 그 확률에 따라 가능성이 더 높은 범주에 속하는 것으로 분류하는 기법이다.

0.5 보다 크면 어떤 사건이 일어난다.(성공확률)
0.5 보다 작으면 어떤 사건이 일어나지 않는다.(실패확률)



〈로지스틱 회귀〉
0과 1사이의 확률값을 확인한다.

합격/불합격, 성공/실패, 생존/사망, 진실/거짓 등 이분법적인 결과를 도출하기 위해 주로 사용되는 회귀분석 방식으로 예측을 주목적으로 하는 회귀분석과 차이가 있다.

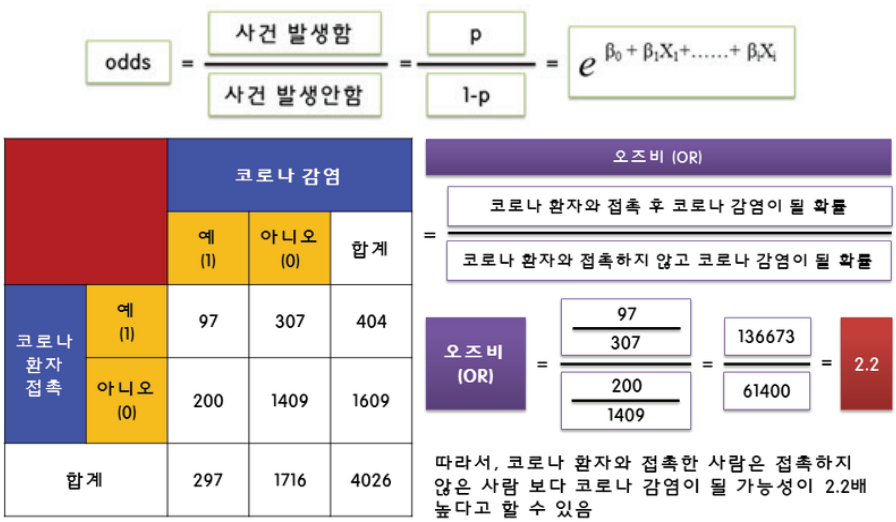
로지스틱 회귀분석 방법

로지스틱 회귀 분석은 이진 분류를 수행하는 데 사용된다. 즉, 데이터 샘플을 양성(1) 또는 음성(0) 클래스 둘 중 어디에 속하는지 예측한다. 각 속성(feature)들의 계수 log-odds를 구한 후 시그모이드 함수를 적용하여 실제로 데이터가 해당 클래스에 속할 확률을 0과 1사이의 값으로 나타낸다.

오즈비(Odds Ratio)

오즈비는 사건이 발생할 확률이 발생하지 않을 확률에 비해 몇 배 더 높은가를 설명하는 개념이다.

P(성공확률) = 사건 발생함, 1-P(실패확률) = 사건 발생 X



*이미지 출처: 네이버 블로그(도시마법사)

A : 97명 – 코로나 환자와 접촉 후, 코로나에 감염됨

B : 307명 – 코로나 환자와 접촉 후, 코로나에 감염되지 않음

C : 200명 – 코로나 환자와 접촉하지 않았지만, 코로나에 감염됨

D : 1409명 – 코로나 환자와 접촉하지 않았고, 코로나에 감염되지 않음

Odds Ratio(OR) = 136673 / 61400 = 2.2로 계산된다.

따라서, 코로나 환자와 접촉한 사람은 접촉하지 않은 사람보다 코로나 감염이 될 가능성이 2.2배 높다고 할 수 있다.

로짓(logit) 변환: 오즈비에 log 함수 적용한 것으로 로짓을 대상으로 회귀분석을 적용한 것이 로지스틱 회귀분석이 된다.

$$logit = log(p / 1-p)$$

문제정의

N대학교의 대학원 입학처는 본교 대학원에 입학하는 학생이 어떠한 특성을 가지는지 파악하고 대학원에 입학할 가능성이 있는 수험생을 예측하고자 한다. 입시점수와 학점이 합격여부에 어떤 영향이 있는지를 확인한다.

DATASET : 데이터 등 관련 자료와 설명

합격여부, 입시점수 및 학점에 대한 정보를 test.csv 파일로 저장한다.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import matplotlib.pyplot as plt

## 사이킷런(sklearn) 패키지 이용
from sklearn import datasets
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, roc_auc_score, roc_curve

data = pd.read_csv('./Logistic Regression/test.csv', encoding = 'cp949')
data.head()
```

	합격여부	입시점수	학점
0	0	380	3.61
1	1	660	3.67
2	1	800	4.00
3	1	640	3.19
4	0	520	2.93

분석

독립변수와 종속변수를 지정해주고 훈련용 데이터와 검증용 데이터를 7:3으로 분할해준다.

```
##변수 지정
x = data[['입시점수', '학점']]   #독립변수
y = data[['합격여부']]          #종속변수

##학습데이터와 평가(test)데이터 분리
##random_state는 난수 생성 프로그램을 사용할 때 사용한다. 기본값은 없다.
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
from sklearn.linear_model import LogisticRegression

##로지스틱 회귀분석 모델 생성
model = LogisticRegression()
```

```
##모델의 정확도 확인
model.fit(x_train, np.ravel(y_train))
print('학습용 데이터 세트 정확도 : %.2f' % model.score(x_train, y_train))
print('검증용 데이터 세트 정확도 : %.2f' % model.score(x_test, y_test))
```

학습용 데이터 세트 정확도 : 0.68
검증용 데이터 세트정확도 : 0.68

조금 더 자세한 평가를 위해 classification_report 모듈을 이용한다. 이 모듈은 모형 성능평가를 위한 모듈로 정밀도(precision), 재현율(recall), F1-score, support를 구해준다.

```
from sklearn.metrics import classification_report
y_pred=log.predict(x_test)
print(classification_report(y_test, y_pred))
```

	precision	*recall*	*f1-score*	*support*
0	*0.68*	*1.00*	*0.81*	*82*
1	*0.00*	*0.00*	*0.00*	*38*
avg / total	*0.47*	*0.68*	*0.55*	*120*

이 모델의 정밀도의 수치는 0.47, 재현율의 수치는 0.68로 나타났다

이제 모델의 회귀계수와 오즈비를 구해 독립변수가 분류 결정에 미치는 영향의 정도를 알아보기 위해 다른 방식의 로지스틱 회귀분석을 진행해보자. logit이란 변수에 합격 여부를 종속변수로 하는 데이터를 넣고, 앞서 넣은 독립변수 x값도 입력한 뒤 적합 시켜준다.

```
##로지스틱 회귀분석 학습
trains_cols = data.columns[1:] ## train_cols는 독립(설명)변수
logit = sm.Logit(data[['합격여부']],x)

##result = logit.fit()
##로지스틱 회귀모델을 적합 시킬 때에 사용하는 최적화 기법 지정
result = logit.fit(method = 'newton')
```

Optimization terminated successfully.
Current function value: 0.629448
Iterations 5

정밀도 : 모델이 True라고 분류한 것 중에서 실제 True인 것의 비율이다.
예) 날씨 예측 모델이 맑다로 예측했는데, 실제 날씨가 맑았는지를 살펴보는 지표

재현율 : 실제 True인 것 중에서 모델이 True라고 예측한 것의 비율이다.
예) 실제 날씨가 맑은 날 중에서 모델이 맑다고 예측한 비율을 나타낸 지표

```
## 회귀계수 확인
result.params
```

```
입시점수    0.001563
학점       -0.482319
dtype: float64
```

```
##summary 함수로 결과 확인
result.summary2()
```

Model:	Logit	No. Iterations:	5
Dependent Variable:	합격여부	Pseudo R-squared:	-0.007
Date:	2019-02-28 13:31	AIC:	507.5584
No. Observations:	400	BIC:	515.5414
Df Model:	1	Log-Likelihood:	-251.78
Df Residuals:	398	LL-Null:	-249.99
Converged:	1	Scale:	1

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
입시점수	0.0016	0.001	1.5825	0.1135	-0.0004	0.0035
학점	-0.4823	0.1752	-2.7522	0.0059	-0.8257	-0.1388

각 회귀계수가 합격 여부에 어떻게 영향을 미치는지 확인한다. 먼저, 유의확률(P>|z|) 즉 P-value를 본다. P-value의 기본값인 0.05 이하인 경우 유의미한 값으로 보기 때문에 학점 변수가 종속변수에 영향을 주는 유의미한 변수임을 알 수 있다. 입시점수를 제외하고 계속해서 해석을 진행한다. 편회귀계수(Coef.)의 부호를 통해 종속변수에 미치는 영향의 방향을 파악할 수 있다. 편회귀계수(Coef.)의 값이 양수라면 합격 여부가 '1'일 확률이 높아진다는 뜻이다. 반대로 음수라면 합격 여부의 값이 '0'일 확률이 높아진다는 뜻이다. 따라서 모형의 회귀계수 분석결과는 다음과 같다. 통계량에 나와있는 편회귀계수의 값 자체만으로는 각 변수들이 종속변수에 얼마나 영향을 주는지는 파악할 수 없다.

유의확률 : 유의확률(p-value, p값)은 귀무가설이 진실일 때 적어도 그 정도의 극단적인 표본값이 나올 확률, 즉 귀무가설이 참임에도 이를 기각할 확률
유의수준 : 유의수준은 귀무가설의 기각여부를 결정하는데 사용하는 기준이 되는 확률
귀무가설 : 설정한 가설이 진실일 확률이 극히 적어 처음부터 버릴 것이 예상되는 가설
편회귀계수 : 회귀식에 포함되는 다른 변수의 영향을 제거한 후 독립변수가 종속변수에 주는 영향을 나타낸다. 원인(독립변수)이 여러 개인 다중회귀분석에서의 회귀계수를 의미

종속변수에 미치는 정도는 오즈비(Odds Ratio, 승산비)를 통해 파악할 수 있다.
다음과 같은 코드를 작성해서 오즈비를 구해보자.
np.exp(result.params)

```
입시점수    1.001563
학점        0.617389
dtype:      float64
```

오즈비가 1을 기준으로 큰지 작은지를 파악하여 종속변수에 미치는 영향의 방향을 파악할 수 있다. 독립변수가 두개 이상 있을 때는 다른 독립변수를 일정한 값으로 고정한 경우의 오즈비로 해석된다. 아무런 관계가 없을 때 오즈비는 1이다. 1에서 멀리 떨어질수록 종속변수와의 관계가 강하다는 뜻이다. 즉, 종속변수 여부에 큰 영향을 준다는 뜻이다.

오즈비는 1을 기준으로 영향을 판단하므로, 오즈비가 10인 경우와 0.1인 경우는 종속변수에 영향을 주는 강도가 같다. 입시점수 변수의 경우, 극도로 1에 가까운 값으로 나타난다. 따라서, 입시점수는 합격여부에 별다른 영향을 주지 않았음(관계 없음)을 알 수 있다. 반면, 학점의 경우는 각 오즈비가 0.61로 1과 멀어져 있으므로 합격여부에 영향을 미쳤음을 알 수 있다. 독립변수가 수치형일 경우 또 다르게 오즈비를 해석할 수 있다. 학점이 1단위 증가하면 합격할 확률이 0.61배 증가한다는 뜻으로도 해석할 수 있다. 따라서, 대학원 입학에는 입시점수보다 학점이 중요한 것을 알 수 있었다.

성능 확인

```
##cut_off 함수를 생성하여 임계치(threshold) 설정
##로지스틱은 확률값을 표현하기 때문에 임계치를 정해줘야 함
##일반적으로 임계치를 0.5로 정하여, 0.5 이상이면 1로, 아니면 0으로 판단
def cut_off(y, threshold):
  Y = y.copy()   #대문자 Y를 새로운 변수로 지정하여 기존의 y값에 영향을 주지 않도록 함
  Y[Y 〉 threshold] = 1
  Y[Y 〈= threshold] = 0
  return(Y.astype(int))
pred_Y = cut_off(pred_y, 0.5)
##confusion_matrix 함수로 혼동행렬을 뽑는다
cfmat = confusion_matrix(y_test, pred_Y)
print(cfmat)
```

```
[[ 8  4]
 [ 4 14]]
```

```
##혼동행렬의 결괏값으로 정확도를 구한다.
(cfmat[0,0] + cfmat[1,1]) / np.sum(cfmat) ##accuracy
```

```
0.7333333333333333
```

```
def acc(cfmat):
  acc = (cfmat[0,0] + cfmat[1,1]) / np.sum(cfmat)
  return(acc)

##cut_off 함수에 따른 성능지표 비교
##임계치(threshold) 따라 성능이 달라질 수 있다.
##조금씩 바꾸면서 성능이 어떻게 바뀌는지 확인
threshold = np.arange(0,1,0.1)
table = pd.DataFrame(columns = ['ACC'])

for i in threshold:
  pred_Y = cut_off(pred_y, i)
  cfmat = confusion_matrix(y_test, pred_Y)
  table.loc[i] = acc(cfmat)

table.index_name = 'threshold'
table.columns.name = 'preformance'
table
```

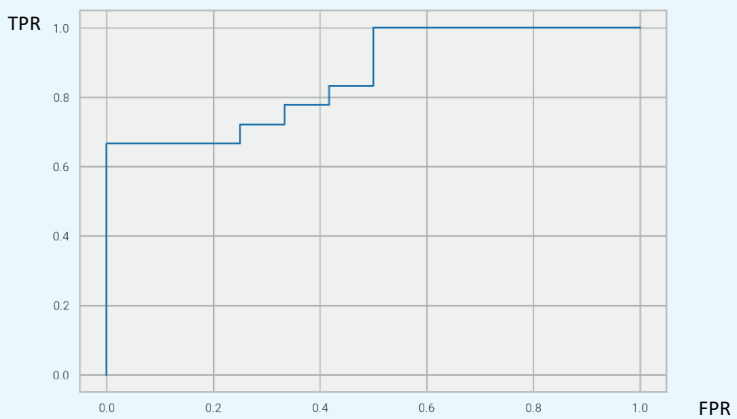
preformance	ACC
0.0	0.600000
0.1	0.766667
0.2	0.766667
0.3	0.733333
0.4	0.733333
0.5	0.733333
0.6	0.733333
0.7	0.733333
0.8	0.800000
0.9	0.700000

정확도(ACC)값이 가장 높은 부분의 임계치(threshold)가 성능이 가장 좋은 것으로 0.8 정도가 가장 성능이 좋게 나온 것을 알 수 있다.

Confusion_matrix(혼동행렬)

특정 분류 모델의 성능을 평가하는 지표로, 실젯값과 모델이 예측한 예측값을 한 눈에 알아볼 수 있게 배열한 행렬이다. 이진 분류의 경우 실젯값과 예측값을 참/거짓에 따라 분류한다.

```
##ROC, AUC로 해당 모델 검증
##sklearn ROC 패키지 이용
fpr, tpr, thresholds = metrics.roc_curve(y_test, pred_y, pos_label = 1)
##Print ROC curve
plt.plot(fpr, tpr)
```




```
##Print AUC
auc = np.trapz(tpr, fpr)
print('AUC:', auc)
```

AUC: 0.861111111111111

8 시계열분석(Time-series Analysis)

시계열분석

시간에 따라 변화되는 자료의 패턴을 밝혀 가까운 미래를 예측하는 방법이다. 시계열분석을 위해서는 시계열 데이터가 준비돼야 한다. 시간의 경과만 한 축(x)을 구성하는 것이 아니라 시간 경과가 일정한 시차로 정돈되어 있을 때 이를 시계열 데이터로 본다.

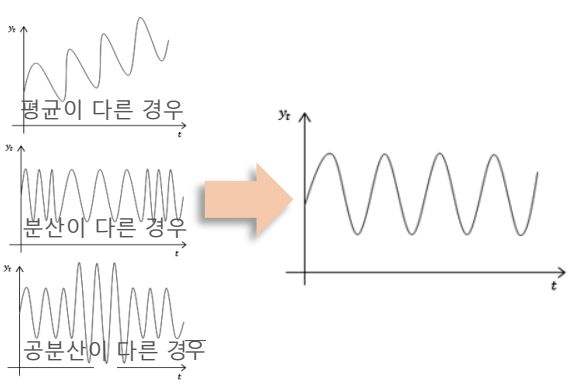
시계열 데이터의 필수 조건, 정상성(Stationary)

정상성이란 '데이터 변동의 안정성'으로 달리 표현할 수 있다. 시간의 흐름에 따라 관측된 결과에서 세로축(**y**) 값의 변동이 지나치게 크다면 그 다음 예측에 관한 정확도가 높을 수 있을까? 그렇지 않을 것이다.

회귀분석에서 살펴보았듯이, 데이터의 분포가 추세선을 기준으로 잘 모여 있을 때(=표준오차가 작을 때), 해당 추세선이 보다 예측력이 높다고 배웠다. 마찬가지로 어떤 시계열 자료가 정확한 간격으로 데이터를 관측했다 할지라도 변동성이 너무 크게 나타난다면 다음 관측 결과에 대한 예측의 의미가 퇴색될 것이다. 따라서 올바른 시계열 분석을 위해 '비정상'시계열 자료를 '정상'시계열 자료로 전처리하는 과정이 분석 만큼이나 중요하다.

정상성을 확보하는 방법, 차분(Difference)과 변환(Transformation)

정상 시계열로 조정하는 방법이다. 일반적으로 평균 변동이 크면 직전 값에서 현재 값을 빼는 차분의 과정을 거치고, 분산이 시점에 따라 다르면 변환 과정을 거쳐 정상성을 충족시킨다. 변환을 통해 정상성을 높이는 방법에는 이동평균법, 지수평활법 등이 있다.



〈비정상자료의 정상성 확보〉
차분과 변환을 통해 평균이 다른 경우, 분산이 다른 경우
그리고 공분산이 다른 경우에 정상성을 확보한다.

시계열분석 모형

시계열분석은 기본적으로 선형 예측을 전제로 한다. 시간을 가로축에 놓은 회귀분석이라고 할 수 있다. 뚜렷한 상관관계를 바탕으로 하는 선형 회귀분석은 미래는 과거를 닮는다는 전제를 바탕으로 시계열분석으로 응용되었다.

종류	설명
AR(*Auto-regressive*)	과거 데이터에 기반하여 미래를 예측하는 모형으로 추세선을 통한 예측 과정은 선형 회귀모델과 동일하다.
MA(*Moving-average*)	직전 데이터와 현재 데이터의 평균인 '이동평균'을 이용한 시계열 분석법이다. 전체 자료 가운데 다른 하위 데이터에 대한 이동평균을 따로 생성해 활용하는 것도 가능하다.
ARIMA (*AR+MA+Momentom*)	결과적으로 회귀분석의 한 형태다. 단, 정상 시계열 자료로만 회귀 예측을 한다는 차이가 있다. MA에 대한 수 차례의 차분을 통해 정상성을 높이고 AR을 수행한다.

〈시계열분석 모형의 대표적인 종류〉

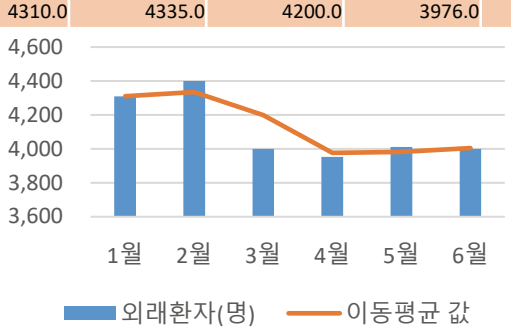
알고 보면 시계열분석도 회귀분석에 한 형태이다.

직전 데이터와 지금 데이터의 평균, MA(Moving-average)

이동평균은 직전 데이터와 지금 데이터의 평균을 의미한다. 아래 표는 서울의 W 의료기관의 외래 환자를 월 단위로 정리하고 월별 이동평균을 구해 그래프로 나타낸 내용이다. 이동평균은 급격한 변동을 보다 완화해 표현하는 특징이 있다.

$$F_{n+1} = \frac{1}{m}(z_n + z_{n+1} + \ldots + z_{n-m})$$

구분	1월	2월	3월	4월	5월	6월
외래 환자(명)	4,310	4,400	4,000	3,952	4,011	4,000
이동평균 계산	= 4310	= (4310+4400)÷2	= (4400+4000) ÷2	= (4000+3952) ÷2	= (3952+4011) ÷2	= (4011+4000) ÷2
이동평균(MA)	4310.0	4335.0	4200.0	3976.0	3981.5	4005.5

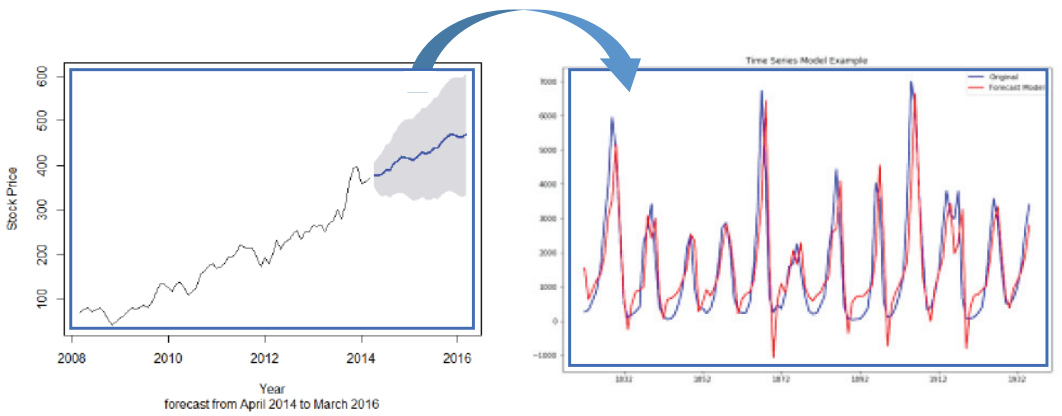


〈*W* 의료기관 상반기 외래 환자 수와 이동평균〉

2월에서 3월의 환자 수가 급감하지만 이동평균값은 이를 '평활화'해(=이동평균법) 나타낸다.

시계열 변동 요인

W 의료기관 상반기 외래 환자 수와 이동평균 그래프를 참고하면, 3월 외래환자 수가 급감한 현황을 확인할 수 있다. 여기서 우리는 겨울에서 봄으로 계절이 변화하면서 감기 환자 수가 감소했고 그 결과 3월부터 외래 환자 수가 감소한다는 예상을 해 볼 수 있다. 만일 매년 계절 변화로 큼지막한 환자 수 변동이 발생한다면 우리는 이를 시계열 자료에 내재된 계절변동(Seasonal Variation)요인이라 말한다(그 외 추세변동, 순환변동, 불규칙변동을 더해 총 4가지 변동 요인이 있다).



〈ARIMA와 SARIMA 결과〉

1년 단위 시계열 자료에서 계절변동 요인을 제거(=계절성을 반영)하면
우측과 같이 더욱 뚜렷해진 정상성을 확인할 수 있다.

지수평활법

모든 시계열 자료를 사용해 평균을 구하고 시간 흐름에 따라 **최근 시계열에 더 높은 가중치를 부여**해 추세를 계산한다.

$$F_{n+1} = \alpha Z_n + \alpha(1-\alpha)Z_{n-1} + \alpha(1-\alpha)^2 Z_{n-2} + \alpha(1-\alpha)^3 Z_{n-3} + \dots$$

차분과 ARIMA(Auto-regressive Integrated Moving-average)

이동평균을 활용해 월별 데이터의 평활화를 실시했지만, 아직 정상 시계열 자료로서 정상성이 높다고 보기 어렵다. 이동평균값을 일정 수준으로 계속 차분(Differencing)하여 회귀분석(≒추세선 예측)하는 것이 ARIMA이다.

II. AI분석모형

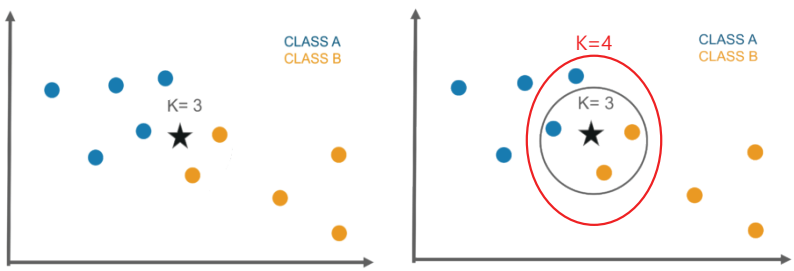
1. 최근접이웃(K-Nearest Neighbors)
2. 의사결정나무(Decision Tree)
3. 랜덤포레스트(Random Forest)
4. 앙상블(Ensemble)
5. 서포트벡터머신(SVM)
6. 군집분석(Clustering Analysis)

III. AI분석모형

1 최근접 이웃(K-Nearest Neighbors)

최근접 이웃

알고리즘은 우리가 예측하려고 하는 임의의 데이터와 가장 가까운 거리의 데이터 K개를 찾아 다수결에 의해 데이터를 예측하는 방법이다.



〈K-근접법 예시〉

새롭게 추가된 별 모양 데이터 주변에 Class B 데이터가 더 많으므로
별 모양 데이터는 Class B로 분류된다.

위 그림과 같이 두 그룹의 데이터가 있을 때 주어진 임의의 데이터가 어느 그룹에 속할 것인지를 K-NN알고리즘으로 풀 수 있다. 별 데이터로부터 가장 가까운 K개 데이터들이 더 많이 속한 그룹으로 별 모양 데이터를 분류하는 알고리즘이다. K가 3이라면 3개의 데이터를 확인해 더 많은 데이터가 있는 B그룹으로 주어진 임의의 데이터를 분류한다. K의 선택 기준은 학습의 난이도와 데이터의 개수이며, 보통 훈련 데이터 개수의 제곱근으로 설정한다.

K와 별 모양 데이터 간 좌표상 거리도 고려해야 한다. K를 4로 설정했다면 별 모양 데이터를 기준으로 4개의 데이터 간 떨어진 거리가 다르다. 별 모양 데이터를 기준으로 파란 데이터 중 1개 데이터의 거리가 가장 멀어 가중치가 가장 낮으므로 B그룹으로 분류된다.

K-NN의 장단점

K-NN알고리즘의 장점은 추가된 데이터의 처리가 쉽고, 예측 결과에 대한 해석도 쉽다는 것이다. 사용이 간단하여 훈련 데이터에 대한 훈련 과정이 별도로 필요하지 않으며, 범주를 나눈 기준을 알지 못하더라도 데이터를 분류할 수 있다.

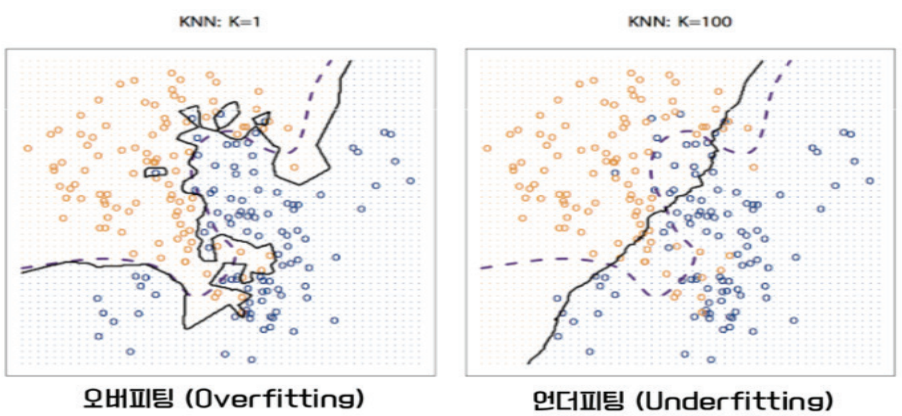
반면 훈련 데이터 세트의 크기가 너무 크거나 특성(feature)의 개수가 너무 많으면 연산속도가 느려지는 단점이 있다. 비수치 데이터의 유사도를 정의하기가 어렵고 이상치의 영향을 크게 받는다. 서로 다른 특성 값들의 비율을 일정하게 하지 않을 경우 성능이 떨어지기 때문에 같은 스케일(규모)을 갖도록 전처리 하는 과정이 필수적이다.

최근접 이웃을 활용한 회귀분석 사례

지도학습 방법으로 클래스가 정해진 데이터에 대해 분류 모델을 만들어 분류예측에 사용하거나 최근접 이웃 회귀분석에도 사용할 수 있다.

K값이 너무 작으면 거리가 가장 가까운 데이터 1개를 기준 삼아 새로운 데이터를 분류하게 되므로 분류 기준이 매우 엄격해진다. 훈련 데이터 세트에 대해서는 높은 정확도를 기대할 수 있지만 임의의 테스트 데이터 세트에는 높은 정확도를 기대하기 어려워진다. 결과적으로 과대적합(overfitting) 문제가 발생할 수 있다.

K값이 너무 크면 거리가 가장 가까운 K개의 모든 데이터를 기준으로 데이터를 분류하기 때문에 모델의 분류 기준이 매우 모호해진다. K가 극단적으로 커질 경우 훈련 데이터에 대해서 정확도 성능이 나오지 않는 과소적합(underfitting) 문제가 생길 수 있다.



〈K값 1과 100 차이〉

K값에 따른 차이점을 명확히 보여준다.

따라서 최근접 이웃 알고리즘을 사용할 때는 적절한 K의 개수를 정하는 것이 매우 중요하다. 일반적으로 K-NN 분류 모델을 훈련시킬 때 일정 범주를 주고 학습을 시키며, 교차검증이나 리샘플링 기법을 이용하여 최적의 성능을 내는 K 값을 구한다.

분석연습

K-NN알고리즘을 이용한 당뇨병 예측

DATASET

National Institute of Diabetes and Digestive and Kidney Diseases에서 제공된 당뇨병 진단 장치로부터 나온 데이터와 당뇨병의 여부를 보여준다. 샘플의 개수는 총 768개이고 21세 이상의 여성이다. 데이터 특성(feature)은 임신횟수, BMI, 인슐린 Level, 나이 등이 있으며, 결과는 당뇨병의 여부(0, 1)로 표현되어 있다.

1) 데이터 확인하기

```
df = pd.read_csv('./diabetes.csv')
df.head()
```

```
> df.head()
```

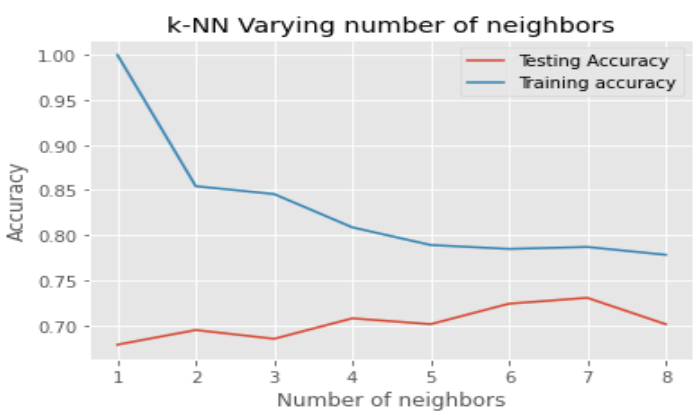
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

2) 훈련, 시험 데이터 분할 (6:4)

```
X_train,X_test,y_train,y_test = train_test_split(X_rescale,y,test_size=0.4,random_state=42, stratify=y)
```

3) K값을 변경하면서 알고리즘 정확도 측정

```
neighbors = np.arange(1,9)
train_accuracy =np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
    #Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)

    #Fit the model
    knn.fit(X_train, y_train)

    #Compute accuracy on the training set
    train_accuracy[i] = knn.score(X_train, y_train)

    #Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```



〈K값을 변경하며 정확도 측정을 했을 때 7에서 가장 차이가 적게 나타나고 있다.〉

4) 훈련 데이터와 시험 데이터 정확도의 차이가 가장 작은 K=7인 값을 이용하여 정확도 측정

```
confusion_matrix(y_test,y_pred)
```

```
pd.crosstab(y_test, y_pred, rownames=['실제'], colnames=['KNN모델'], margins=True)
```

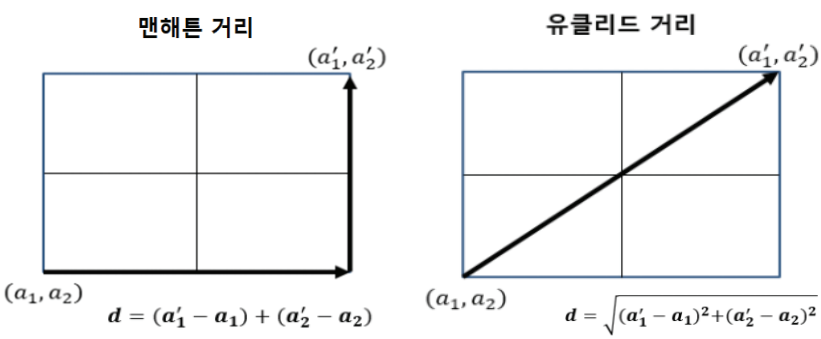
KNN모델	0	1	All
실제			
0	161	40	201
1	40	67	107
All	201	107	308

	precision	recall	f1-score	support
0	0.78	0.82	0.80	201
1	0.62	0.56	0.59	107
accuracy			0.73	308
macro avg	0.70	0.69	0.70	308
weighted avg	0.73	0.73	0.73	308

민코프스키 공식을 이용한 K-NN의 거리측정

```
knn = KNeighborsClassifier(n_neighbors=k, p=2, metric='minkowski')
```

최근접 이웃 알고리즘에서 사용되는 데이터 간 거리를 구하는 방식에는 '맨해튼'과 '유클리드' 측정 방식이 있다. 맨해튼 거리 측정 방식은 두 점 사이의 격자를 고려한 거리 측정 방식이고, 유클리드 거리 측정 방식은 최단거리를 측정한 것이다. 다음은 맨해튼과 유클리드 거리를 나타낸 그림과 공식이다.



$$d = \sqrt[p]{\sum_k (a_k' - a_k)^p}$$

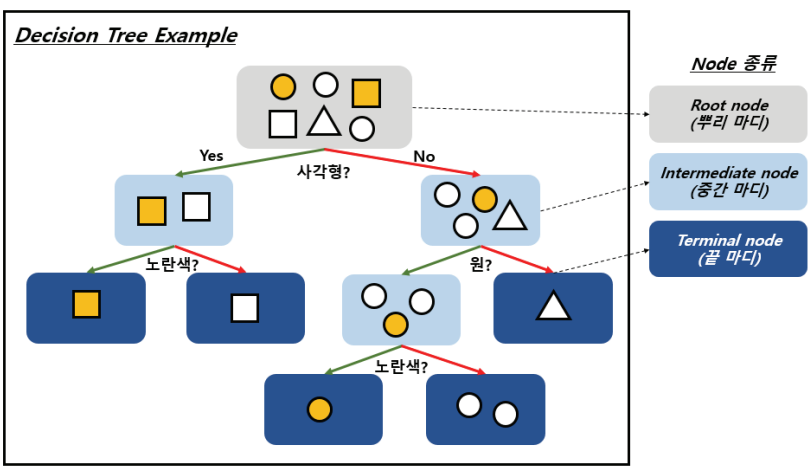
〈민코프스키 공식〉

민코프스키 공식은 거리를 구하는 공식을 일반화해 놓은 것이며, 위 식과 같다. 이때 P=1이면 맨해튼 거리, P=2일 때 유클리드 거리를 계산한다. 파이썬의 KNeighborsClassifier 함수에서 인자 값을 생략 할 경우, 민코프스키에 P=2(유클리드 거리)가 기본값으로 설정되어 있다.

2 의사결정나무(Decision Tree)

의사결정나무

일종의 분류기법이다. 전체 집단을 계속 양분하는 분류기법으로써 분기가 발생하는 포인트(=노드)에는 기준이 되는 질문이 있어 기준 질문에 부합하냐(YES), 부합하지 않느냐(NO)에 따라 노드 이동의 방향이 결정된다.



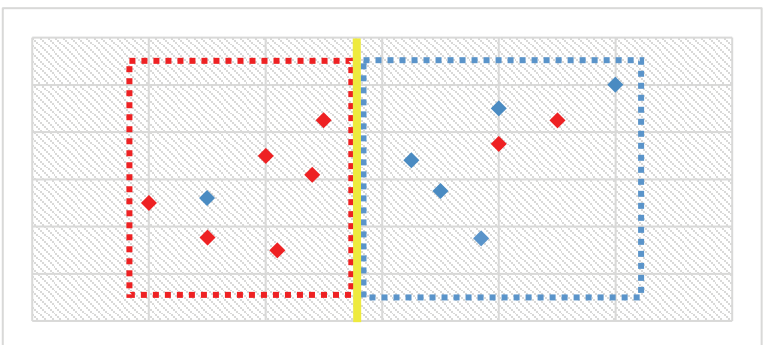
〈의사결정나무 예시〉
그림출처 : Wiki

의사결정나무 모형은 분류(classification)와 회귀예측(regression) 모두 가능한 알고리즘이다. 분류나무 모형은 불연속적(이산형 자료)인 값을 예측한다. 예를 들어 분류 모델은 다음과 같은 질문에 대한 답을 예측한다. '수신된 이메일이 스팸인가 아닌가?', '이 사진이 강아지인가 고양이인가 또는 햄스터인가?'와 같은 질문에 답을 내기 적합한 비지도 분류 알고리즘 모형이다.

한편 회귀분석의 한 갈래인 회귀나무 모형은 연속적인 값을 예측한다. 예를 들어 회귀 모델은 다음과 같은 질문에 답을 예측한다. '원주시 주택 가격은 얼마인가?', '사용자가 이 광고를 클릭할 확률이 얼마인가?'와 같은 질문에 답할 수 있다. 주로 의사결정 나무는 광고 인쇄물의 응답자 분석, 고객들의 신용점수화, 의학 연구, 시장분석, 품질관리, 주가, 환율 예측 등 다양한 분야에서 이용된다.

분할규칙

데이터 집단을 나눌 경우에는 분할 기준이 있다. 분할 기준은 분할 변수와 목표 변수를 통해 산포된 데이터들을 가장 잘 구분해 줄 수 있는 지점(기준)을 찾는 기준이 된다. 아래 그림에서처럼 분할 기준을 찾아가는 방법이 바로 분류나무이며, 이를 통해 나눠진 데이터 그룹은 그 특징을 보다 정확하게 설명할 수 있게 한다. 붉은 영역과 푸른 영역은 각각 다른 데이터를 포함하지만, 다수의 동일 데이터로 구성되어 있어 각 그룹을 구분 짓는 것이 가능하다.



〈데이터 분할 예시〉
노랑색 선을 기준으로 데이터를 구분하면 데이터의 성격을
이전보다 더 명확하게 이해할 수 있다.

순수도/불순도(Purity/Impurity)

위 그림처럼 데이터 분할이 이뤄졌을 때, 붉은 영역이 푸른 영역보다 순수도가 더 높다 또는 불순도(=혼잡도)가 더 낮다고 말할 수 있다. 분할점은 순수도가 최대(=불순도가 최소)가 되도록 설정된다. 불순도가 낮으면 불확실성도 감소하는데 이를 정보이론에서는 정보획득(Information Gain)이라고 한다.

불순도와 엔트로피(Entropy)

결정트리에서 불순도를 측정하는 지표로 엔트로피가 적용되어 있다. 물리학에서 엔트로피가 높다는 것은 무질서가 강하다는 의미다. 정보이론에서도 엔트로피가 높을수록 정보 내용의 기대수준이 떨어지는 현상을 정리했다. 데이터 혼잡도가 높을 경우 엔트로피값도 높게 나타난다.

결정트리분류기(DecisionTreeClassifier)

결정트리분류기(DecisionTreeClassifier)를 통해 아이리스 정보의 몇 가지 속성 가운데 잎사귀 길이(length)와 너비(width)를 기준으로 분류하는 과정을 이해할 수 있다. 아래는 잎사귀 크기 범위에 포함되느냐 아니냐에 따라 데이터를 분할하는 과정을 반복하는 중간중간의 결과들을 그래프로 하나씩 살펴본다.

참고 SOURCECODE

```
from sklearn.datasets import load_iris
data = load_iris()
y = data.target
X = data.data[:, 2:]

feature_names = data.feature_names[2:]

from sklearn.tree import DecisionTreeClassifier
tree1 = DecisionTreeClassifier(criterion='entropy', max_depth=1, random_state=0).fit(X, y)

import io
#import pydot
from IPython.core.display import Image
from sklearn.tree import export_graphviz

def draw_decision_tree(model):
    dot_buf = io.StringIO()
    export_graphviz(model, out_file=dot_buf, feature_names=feature_names)
    graph = pydot.graph_from_dot_data(dot_buf.getvalue())[0]
    image = graph.create_png()
    return Image(image)

print(Image)

def plot_decision_regions(X, y, model, title):
    resolution = 0.01
    markers = ('s', '^', 'o')
    colors = ('red', 'blue', 'lightgreen')
    cmap = mpl.colors.ListedColormap(colors)
    cmap

    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = model.predict(
        np.array([xx1.ravel(), xx2.ravel()]).T).reshape(xx1.shape)

    plt.contour(xx1, xx2, Z, cmap=mpl.colors.ListedColormap(['k']))
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1], alpha=0.8,
                    c=[cmap(idx)], marker=markers[idx], s=80, label=cl)

    plt.xlabel(data.feature_names[2])
    plt.ylabel(data.feature_names[3])
    plt.legend(loc='upper left')
    plt.title(title)

    return Z
```

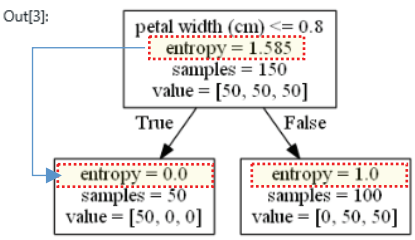
〈결정트리 분할 과정 예시〉

트리 함수가 반환한 결과를 Contour 형태로 시각화하여
몇 번의 분할을 어떻게 거쳤는지 확인한다.

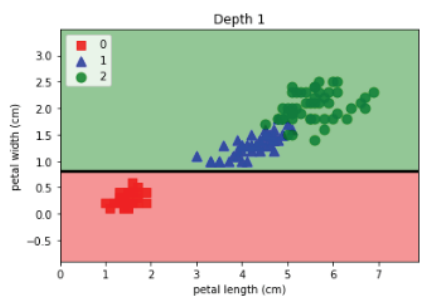
1) 먼저 잎사귀 길이(x)와 너비(y)값의 산포에서 붉은색 영역의 데이터 집단이 매우 독립적인 것을 확인할 수 있다. 여기서 가장 최적의 분류는 잎 너비 변수의 값이 0.8cm일 때이다.

```
In [3]: draw_decision_tree(tree1)
```

Out[3]:




```
In [4]: plot_decision_regions(X, y, tree1, "Depth 1")
        plt.show()
```



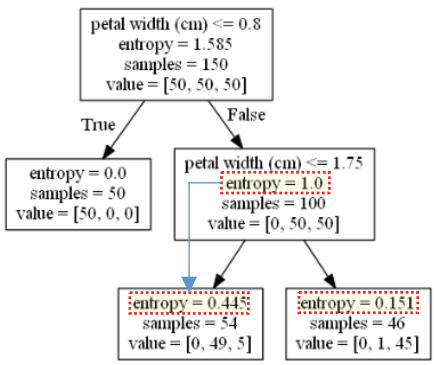
〈첫 번째 데이터 분할〉

데이터를 1:2 크기로 나눈 상황에서
엔트로피가 0인 클래스를 확인할 수 있다.

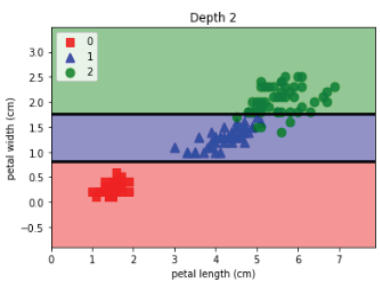
2-1) 다음은 엔트로피가 1인 데이터 그룹을 분할하는 과정이다. 데이터를 비슷한 규모로 양분한 최적의 분할 기준값은 잎사귀 너비 1.75cm이다.

```
In [7]: draw_decision_tree(tree2)
```

Out[7]:




```
In [8]: plot_decision_regions(X, y, tree2, "Depth 2")
        plt.show()
```



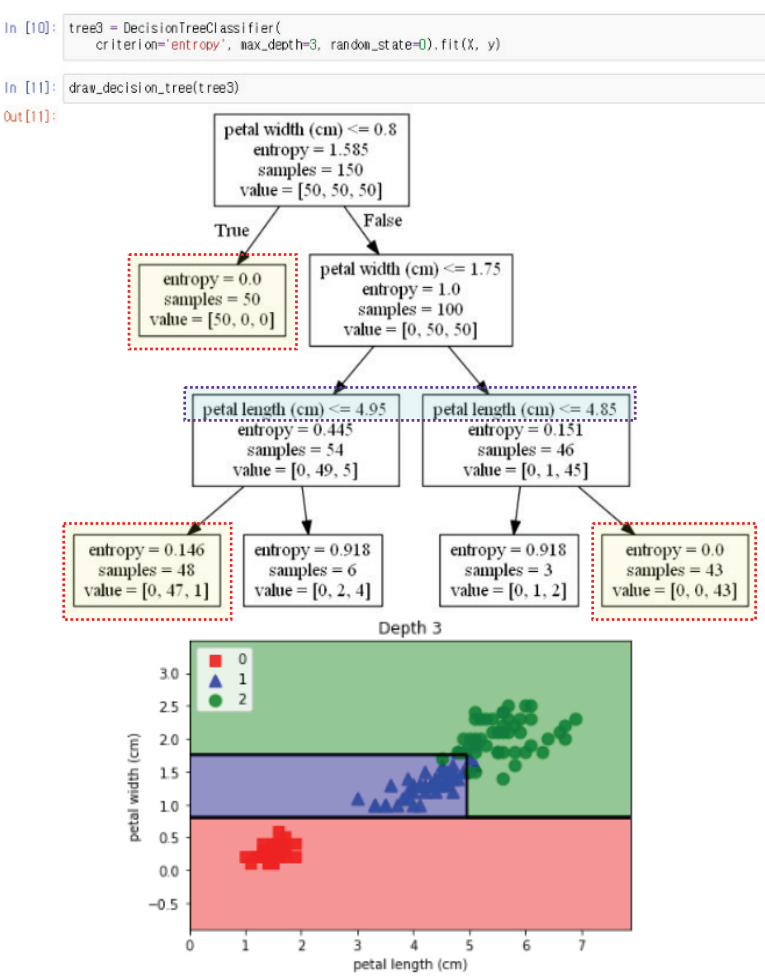
〈두 번째 데이터 분할〉

최적의 분할값을 찾아 분류하여도
엔트로피가 0이 아닐 수 있다.

2-2) 데이터를 비슷한 규모로 양분한 기준을 찾았음에도 파란색 영역에 녹색점이 5건 섞여 있다. 불순도가 1/10을 넘으므로 분할을 한 차례 추가한다.

3-1) 분할 기준을 다른 축에서 찾을 수도 있다. 이번엔 잎의 길이가 기준이 되어 4.95cm 이하와 4.85cm 이상을 나눈 결과를 확인한다. 결과적으로 잎 너비 1.75cm 이하, 길이 4.85cm 이하의 영역이 파란색으로 분류됐다.

3-2) 녹색 영역에 파란 점 1건이 존재하긴 하나 이를 제거하기 위해 트리가 한 차례 더 분류를 진행하는 건 무의미하다. 이미 잎 길이 4.95cm 분할 기준을 높이게 되면 파란색 영역의 불순도가 더 커지게 되기 때문이다(여러 분할 시뮬레이션 가운데 엔트로피 값이 가장 낮을 때(엔트로피 총합: 0.146) 분할 과정은 종료된다).



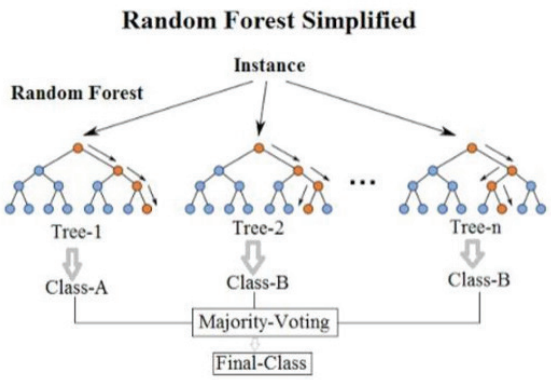
〈세 번째 분할〉

엔트로피가 0이 되는 또 다른 기준을 찾은 결과는
잎 너비 1.75 이하, 길이 4.85 이하의 범위(파란색 영역)이다.

❸ 랜덤포레스트(Random Forest)

랜덤포레스트(RF)

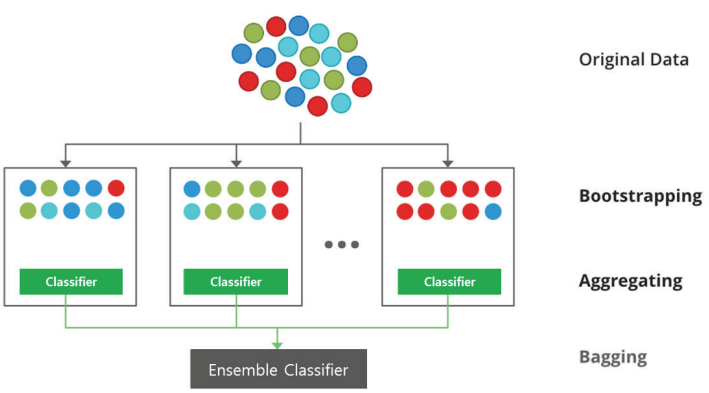
의사결정나무를 여러개 모아서 데이터 분류 및 예측을 수행하는 AI알고리즘이다. 어떤 데이터 집단에 대한 분류나 예측을 실시한다고 할 때, 하나의 결정트리를 사용하는 것보다 여러 트리를 결합해서 사용하면 보다 높은 성능의 알고리즘 모형을 만들 수 있다.



〈랜덤포레스트 구성 예시〉
여러 결정트리를 이용하여 답을 구한다. 하나의 트리만을 이용해 답을 찾는 방식보다 더 높은 성능을 기대할 수 있다.
그림출처 : Wiki

여러 나무와 하나의 포레스트, 배깅(Bagging, Bootstrap Aggregating)

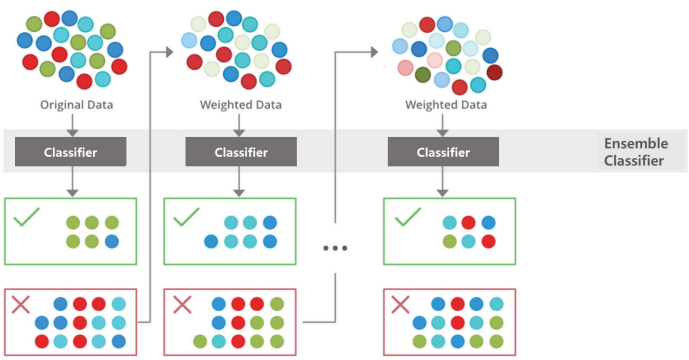
결정트리를 독립병렬 결합하는 방식을 배깅이라고 한다. 원 자료에서 하위 데이터 세트를 만드는 부트스트랩 과정을 포함한다. 데이터 건수는 동일하게 그리고 데이터 무작위 추출 시 중복은 허용(복원추출)하여 각 트리 모형들이 학습할 데이터 세트를 그 수에 맞게 준비해야 한다. 가령 결정트리 서른 개를 사용한다면, 전체 자료에서 데이터를 추출하여 30개의 학습 데이터 세트를 구성해야 한다.



트리마다 독립적으로 데이터 세트를 학습하고 결과를 낸다.
그림출처 : GeeksforGeeks

여러 나무와 하나의 포레스트, 부스팅(Boosting)

결정트리를 연결병렬 결합하는 방식을 부스팅이라고 한다. 첫 번째 결정트리의 결과가 그 다음 결정트리의 학습 데이터 구성에 영향을 미치는 순환구조이다. 무작위 샘플 복원추출과 여러 트리의 답을 최종 결정 시 다수결의 원칙에 맞춘다는 점은 배깅과 동일하다.



〈부스팅 과정 예시〉
이전에 트리의 결과 정보가 반영된 데이터 세트를 학습하고 결과를 낸다.
그림출처 : GeeksforGeeks

두 포레스트 유형 간 주요 차이 4가지

No	Bagging	Boosting
1.	동일 형태의 예측 조합	이종 형태의 예측 조합
2.	분산 감소 목적, 편향 그대로	편향 감소 목적, 분산 그대로
3.	모든 트리별 동일 가중치	트리별 성능 기반 차별적 가중치
4.	트리별 독립적인 학습 데이터 세트 구성	이전 트리에서 잘못 분류된 데이터를 다음 트리 학습 데이터 세트 구성에 포함

〈배깅과 부스팅의 주요 차이점〉
배깅은 병렬적인 결합으로 트리 모두 독립적으로 운용되며,
부스팅은 직렬적인 결합으로 트리별 전후 관계가 중요하다.

하나의 포레스트와 하나의 답, 다수결 원칙(Majority-Voting)

결정트리가 여러 개라면 각각의 결과 역시 여러 개일 것이다. 여러 결과를 모았을 때, 그 대표 결과를 선정하는 것은 다수결 원칙에 따른다. 예를 들어, 결정트리 3개 중 2개의 결과가 '사자'라면 나머지 하나의 트리에서 '고양이'라는 결과가 나오더라도 최종의 결과는 '사자'가 된다.

랜덤포레스트의 분류모형(RandomForestClassifier)

랜덤포레스트의 분류모형(RandomForestClassifier)을 활용하여 아이리스(iris)의 종 분류 시 RFC분석 과정을 알아본다. 파이썬 연습에 대표 예제인 아아리스 데이터 세트를 사용하여 모델이 스스로 아이리스의 종류를 나눈 결과를 대칭행렬(4 x 4)로 시각화하여 확인한다.

데이터 세트에는 아이리스의 속성 정보가 기록돼 있다. 잎사귀의 너비, 꽃잎의 색상 및 크기 등 총 4가지 주요 정보를 바탕으로 모델은 해당 속성 정보를 분석해 특징을 도출하고 그 특징을 기준으로 어떤 데이터가 어떤 종류의 아이리스인지를 분류하게 될 것이다.

참고 SOURCECODE

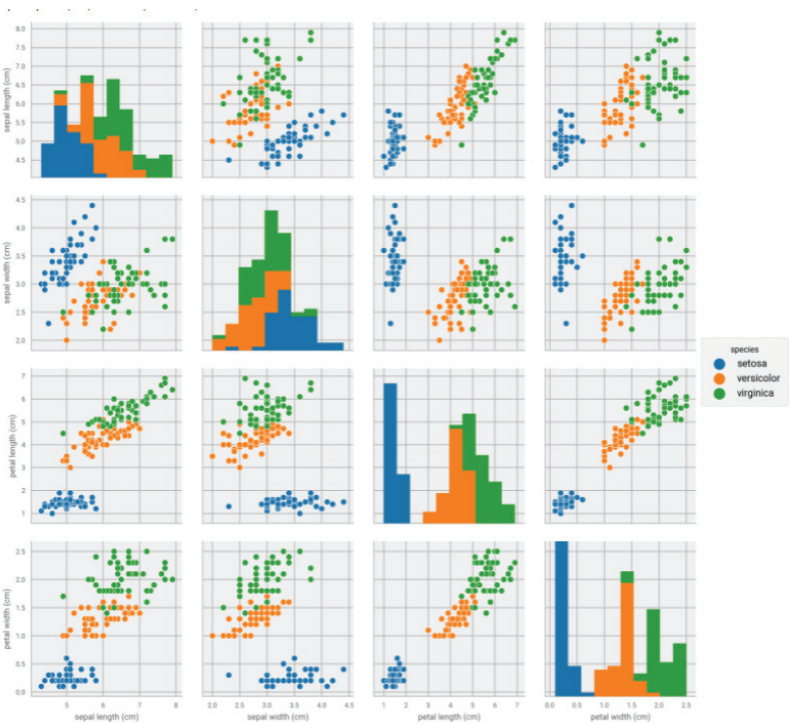
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn import datasets

iris = datasets.load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)
#print(df)
# sklearn provides the iris species as integer values since this is required for classification
# here we're just adding a column with the species names to the dataframe for visualisation
df['species'] = np.array([iris.target_names[i] for i in iris.target]) ---
#print(df['species'])

sns.pairplot(df, hue='species')
# ------------------------------------------------------------ 데이터 프레임으로 변경

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df[iris.feature_names], iris.target, test_size=0.25, s
# ------------------------------------------------------------ 학습/훈련데이터 구분
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, oob_score=True, random_state=9999, bootstrap=True)
rf.fit(X_train, y_train)

from sklearn.metrics import accuracy_score

predicted = rf.predict(X_test)
accuracy = accuracy_score(y_test, predicted)

print(f'Out-of-bag score estimate: {rf.oob_score_:.3}')
print(f'Mean accuracy score: {accuracy:.3}')

from sklearn.metrics import confusion_matrix

cm = pd.DataFrame(confusion_matrix(y_test, predicted), columns=iris.target_names, index=iris.target_names)
sns.heatmap(cm, annot=True)
```

〈랜덤포레스트 분류모형 소스코드 예시〉

scikit-learn 라이브러리의 랜덤포레스트 분류기를 통해
아이리스 속성 정보를 가지고 꽃의 종류를 나눈다.

의사결정나무에서 확인했던 모델이 아이리스 데이터를 언제까지 분할하는지를 상기한다. 데이터의 불순도가 낮아지는(엔트로피가 0에 가까운) 상태까지 모델이 데이터 분할을 시도할 것이다.

산포도를 중심으로 보면 파란색 데이터가 가장 먼저 분류가 이뤄졌을 것이란 예상이 가능하다. 변수 간 관계와 정밀도(=모여있는 정도)가 제일 명확하기 때문이다. 문제는 데이터 중첩이 많은 주황색과 녹색 자료인데, 불순도가 최소가 되는 지점에서 여러 결정트리의 분류 결과를 계산(다수결)해 최종적으로 두 자료를 나누게 된다. 결과적으로 총 3종으로 아이리스 데이터가 표현되었다.



〈아이리스 속성 4가지의 시각화 대칭행렬〉

속성별(4 x 4) 데이터 분포로 모델은 총 3종으로 아이리스 데이터 세트를 분류하였다.

scikit-learn 라이브러리에는 랜덤포레스트의 분류모형과 회귀모형 모두 있다. 랜덤포레스트 알고리즘은 이산형 자료 바탕의 '분류'와 연속형 자료 바탕의 '예측'을 목적으로 한다.

랜덤포레스트의 회귀모형(RandomForestRegressor)

랜덤포레스트의 회귀모형(RandomForestRegressor)을 활용하여 와인 맛 평가에 가장 중요한 요인이 무엇인지 확인한다. 여기서는 와인 성분이 정리된 정보표를 RFR모델에 학습시키고 해당 모델의 어떤 성분이 와인 종류를 나누는 데 가장 큰 영향을 미치는 성분인지를 예측하는 과정을 확인한다.

SOURCECODE와 설명

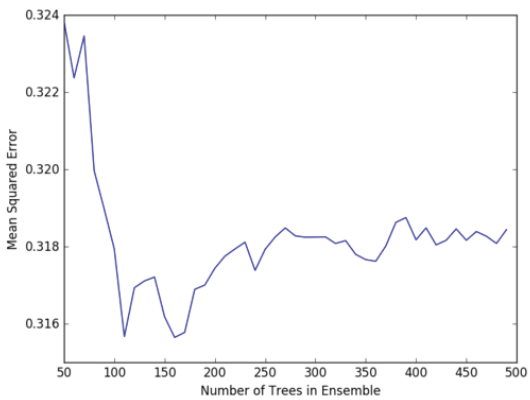
```
nrows = len(xList)
ncols = len(xList[0])
X = numpy.array(xList)
Y = numpy.array(labels)
wineNames = numpy.array(names)
#print(X);print(Y);print(wineNames)

#데이터 행의 30%로 고정된 홀드 아웃 세트 구성
xTrain, xTest, yTrain, yTest = train_test_split(X, Y, test_size = 0.3, random_state = 531
#print(xTrain);print(xTest);print(yTrain);print(yTest)

#MSE의 변화를 확인하기 위하여 앙상블의 크기 범위에서 랜덤포레스트 트레이닝
mseOos = []
nTreeList = range(50, 500, 10)
for iTrees in nTreeList:
    depth = None
    maxFeat = 4 #조정해 볼 것
    wineRFModel = ensemble.RandomForestRegressor(n_estimators = iTrees,
                        max_depth = depth, max_features = maxFeat,
                        oob_score = False, random_state = 531)
    wineRFModel.fit(xTrain, yTrain)

    #데이터 세트에 대한 MSE 누적
    prediction = wineRFModel.predict(xTest)
    mseOos.append(mean_squared_error(yTest, prediction))

print("MSE")
print(mseOos[-1])

#트레이닝 테스트 오차 대비 앙상블의 트리 개수 도표 그리기
plot.plot(nTreeList, mseOos)
plot.xlabel('Number of Trees in Ensemble')
plot.ylabel('Mean Squared Error')
#plot.ylim([0.0, 1.1*max(mseOob)])
plot.show()
```

〈랜덤포레스트 회귀모형 소스코드 예시〉

scikit-learn 라이브러리의 랜덤포레스트 회귀모형을 활용하여
적정한 트리 개수와 와인 맛 구성에 중요한 성분 값을 찾는다.

랜덤포레스트(Random Forest) : 실습

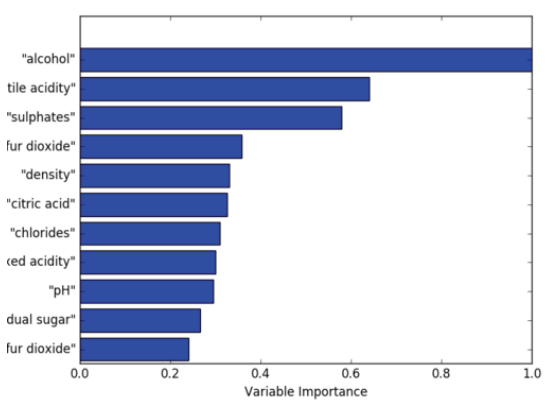
랜덤포레스트에서 결정트리의 수가 정확도를 항상 보장하진 않는다. 아래 그림에서 처럼 알고리즘은 결정트리의 적정한 수와 더불어 원 자료를 근거로 마련된 데이터 학습이 잘 선행되어야 일정 수준의 성능을 기대할 수 있다.



〈결정트리의 개수와 오류 발생의 관계 검정〉

결정트리도 최적의 수가 있다. 오분류율이 가장 낮은 점이 최적이다.

와인 종류 정보를 포함해 와인을 구성하는 성분표 데이터 세트를 **RF**회귀 모델에 학습시키고 해당 모델에서 어떤 성분이 와인 맛(=종류)을 결정하는 데 유의미하게 영향을 미쳤는지를 예측한 결과이다. 그래프를 통해 확인하면 알코올(alcohol)과 산도(tile acidity) 등의 성분이 와인 맛을 가르는 중요한 성분으로 예측됐다.



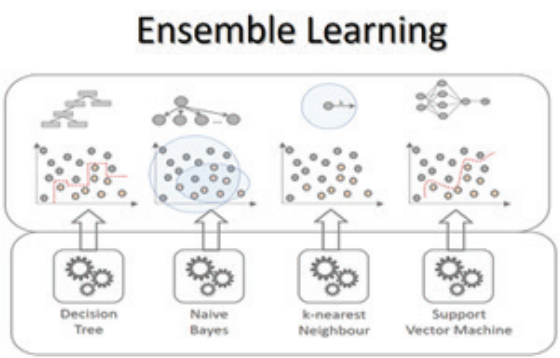
〈와인 맛 평가 주요 요인 분석 결과〉

와인을 구성하는 여러 성분 요소 가운데 알코올이
와인의 종류를 나누는 가장 강력한 성분임을 보여준다.

4 앙상블(Ensemble)

앙상블 학습(Ensemble Learning)

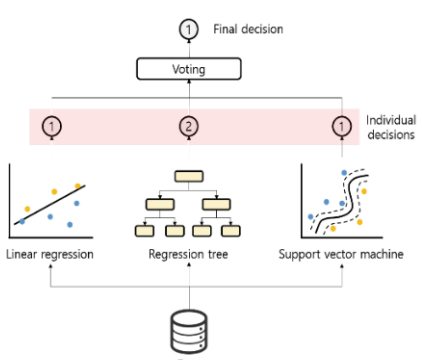
그동안 우리는 여러 기계학습 모델들을 살펴보았다. 이렇게 개별적으로 동작하는 모델들을 모아 종합적으로 의사결정을 한다면 어떨까? 앙상블은 프랑스어로 전체적인 어울림이나 통일을 의미하며 음악에서 2인 이상의 가창이나 합주를 의미하기도 한다. 예측력을 높이기 위해 여러 가지 학습알고리즘이 어울려 성능을 높이는 방식을 기계학습 분야에서 앙상블(Ensemble) 학습이라 한다.



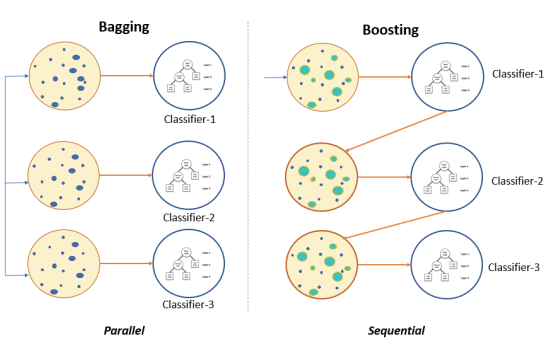
〈앙상블 학습〉

앙상블 종류

앙상블 학습 방식에는 크게 Voting , Bagging, Boosting의 3가지로 나눌 수 있다. Voting(투표) 방식은 여러 모델로부터 나온 후보 결과들 중 다수가 나온 결과를 채택하는 방식이다. Bagging은 같은 모델을 사용하지만 훈련세트에서 중복을 허용한 여러 부분 집합을 샘플링하여 학습하는 방식이며, 여기서 모델이 의사결정나무를 이용한 것을 랜덤포레스트라고 한다. 마지막으로 Boosting은 약한 개별의 모델들을 연결하여 이전 모델이 생성한 가중치 또는 오차를 기준으로 성능을 보완해 나가면서 학습하는 방법이다.



〈Voting〉

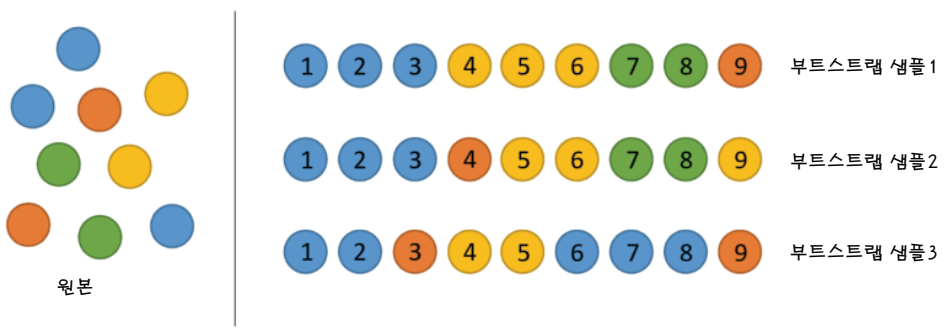


〈Bagging과 Boosting〉

앙상블의 장단점

일반적으로 앙상블 학습 방법을 사용하면, 개별 모델에서 나온 결과를 종합하는 과정에서 종합 결괏값인 분산이 줄어들게 된다. 즉 특정 훈련 데이터 세트에서만 정확도를 보장하는 과대적합(Overfitting)현상을 어느 정도 줄여 줄 수 있다. 다만 투표방식인 Voting의 경우 앙상블에 포함된 모델 사이의 독립성이 전제되는 경우 개별 분류기보다 정확한 예측이 가능하지만 그렇지 못한 경우 성능이 떨어질 수 있다.

배깅(Bagging) vs 페이스팅(Pasting)



배깅은 Bootstrap Aggregating의 줄임말로 중복을 허용한 무작위 표본 추출에 의존하는 어떤 시험이나 계측으로 훈련세트를 구성하고, 페이스팅은 중복을 허용하지 않고 데이터 세트를 구성하는 것을 말한다.

예제

단일 모델과 앙상블 기법(Voting, Bagging, Boosting)을 이용한 유방암 분류 예측 비교 분석

DATASET

유방암 조직에서 미세 바늘을 이용하여 세포를 획득한 후 이미지를 디지털화한 정보이다. 특성은 이미지 속 수치 정보 10개(둘레, 반지름, 질감 등)를 각각 평균, 표준편차, 가장 큰 3개 값의 평균으로 하여 총 30개이며 결과는 악성(Malignant), 양성(benign)이다.

1) 데이터 확인하기

```
cancer = load_breast_cancer()

df = pd.DataFrame(cancer.data, columns=cancer.feature_names)

df.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0

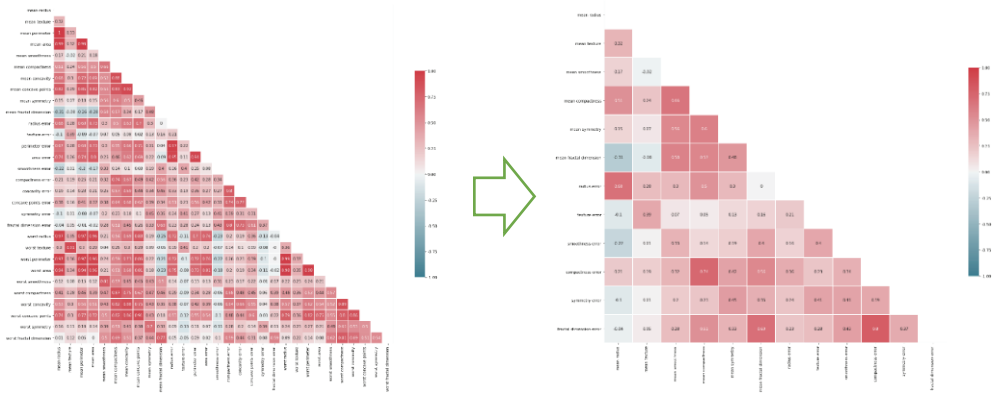
5 rows × 30 columns

2) 탐색적 데이터 분석 – 결측치 확인 및 공분산 매트릭스를 통한 변수 제거

```
# Generate and visualize the correlation matrix
corr = df.corr().round(2)

# Mask for the upper triangle
mask = np.zeros_like(corr, dtype=np.bool)

mask[np.triu_indices_from(mask)] = True

# Set figure size
f, ax = plt.subplots(figsize=(20, 20))

# Define custom colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap
sns.heatmap(corr, mask=mask, cmap=cmap, vmin=-1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)
```

가장 큰 3개 값의 평균, 둘레 넓이, Concavity, Concave point 데이터 간의 강한 상관관계가 발견되어 특성값 제거 (30개 ▶ 12개)

```
cols = ['worst radius', 'worst texture', 'worst perimeter', 'worst area',
        'worst smoothness', 'worst compactness','worst concavity',
        'worst concave points', 'worst symmetry', 'worst fractal dimension']
df = df.drop(cols, axis=1)

cols = ['mean perimeter', 'perimeter error', 'mean area','area error']
df = df.drop(cols, axis=1)
```

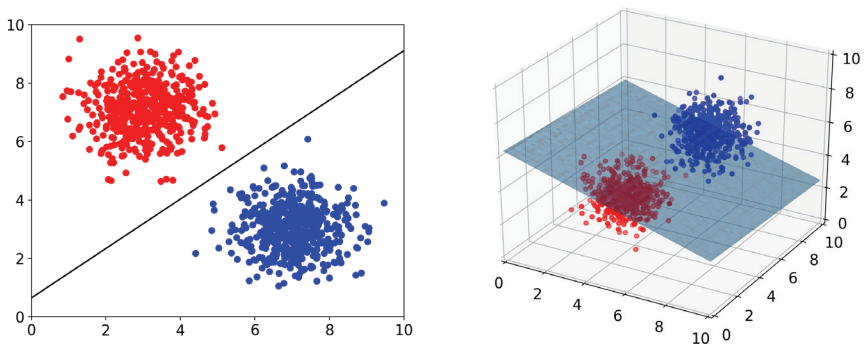


〈특성 30개〉 〈특성 12개〉

5 서포트벡터머신(SVM)

서포트벡터머신(SVM)

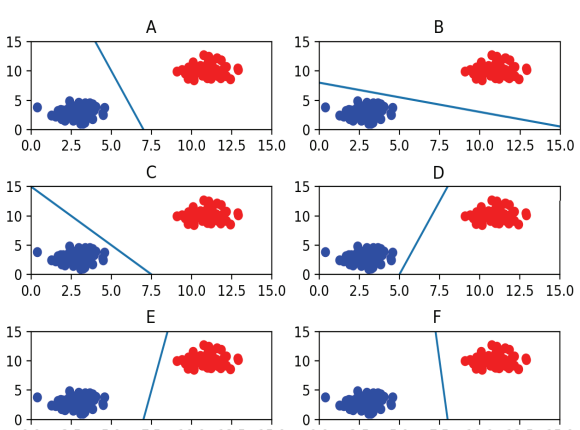
분류를 위한 기준선을 정의하는 모델이다. 분류되지 않은 새로운 점이 나타나면 어느 쪽에 속하는지 확인을 위해서 결정 경계를 정한다.



〈결정 경계(Decision Boundary) 이해〉
2차원과 3차원에서 두 그룹을 분류하는 최적의 경계를 찾는다.
(이하 내용 및 그림출처 : ojune575 블로그)

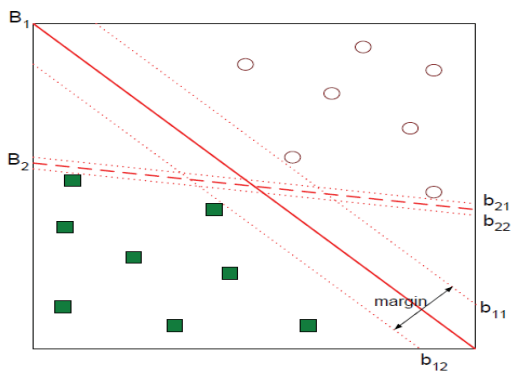
데이터에 2개 속성(feature)만 있다면 결정 경계는 간단한 선 형태가 된다. 속성이 3개가 되면 3차원으로 그려야 한다. 이때의 결정 경계는 '선'이 아닌 '평면'이 된다. 우리가 이렇게 시각적으로 인지할 수 있는 범위는 딱 3차원까지다. 차원, 즉 속성의 개수가 늘어날수록 당연히 복잡해질 것이다. 결정 경계도 단순한 평면이 아닌 고차원이 되는데 이를 '초평면(hyperplane)'이라고 부른다.

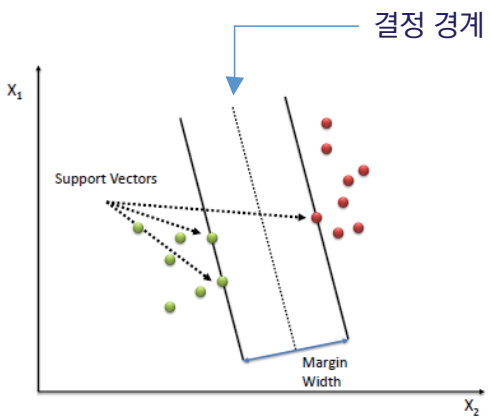
결정 경계는 여러 형태로 정할 수 있다. 그래프 C를 보면 경계선이 파란색 클래스(분류)와 너무 가까워 보인다. 결정 경계가 가장 적절한 것은 그래프 F다. 두 클래스 사이에서 거리가 가장 멀기 때문이다. 결정 경계는 데이터 군으로부터 최대한 멀리 떨어지도록 설정하는 것이 좋다.



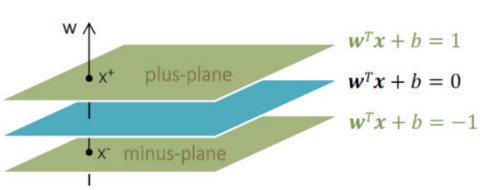
마진(Margin)의 이해

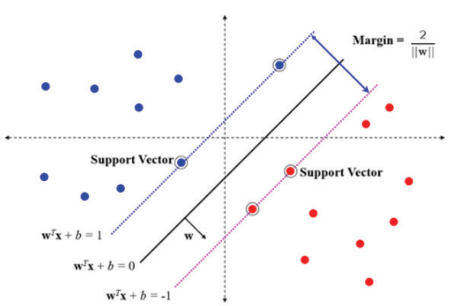
마진(Margin)은 결정 경계와 서포트 벡터 사이의 거리를 의미한다.





위의 왼쪽 그림에서 직선 B1와 B2 모두 두 클래스를 무난하게 분류하고 있음을 확인할 수 있다. 좀 더 나은 결정경계를 꼽으라면 B1일 것이다. 두 범주를 여유 있게 구분하고 있다. b11을 plus-plane, b12를 minus-plane, 이 둘 사이의 거리가 마진(margin)이다. 결정 경계에 가장 가까운 각 클래스의 점들(데이터 포인트들)을 서포트벡터(support vertors)라 한다. 마진의 결정에 영향을 끼치는 관측치들로써, 이 데이터들이 경계를 정의하는 결정적인 역할을 하는 셈이다. SVM은 이 마진을 최대화하는 분류 경계면을 찾는 기법이라 할 수 있다.

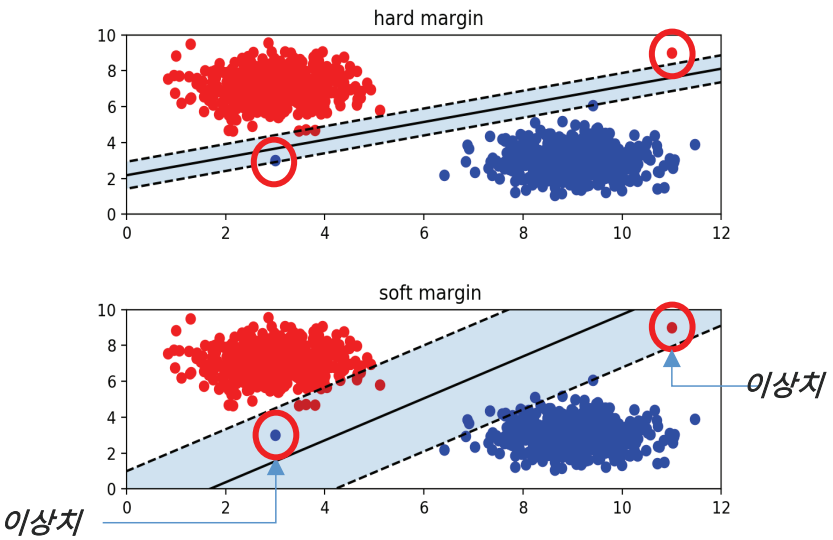




경계면을 도식적으로 나타내면 위의 왼쪽 그림과 같다. 도식화된 분류 경계면을 $w^T + b$ 이라고 할 때 벡터 w는 이 경계면과 수직인 법선벡터가 되며, 오른쪽 그림과 같이 마진의 폭은 $\frac{2}{||w||}$ 으로 정의된다. 마진의 폭이 최대가 되려면 벡터 **w**가 최소가 되어야 한다.

이상치(극단값, outlier)

이상치는 '패턴에서 벗어난 값' 또는 '중심에서 많이 벗어난 값'을 의미한다. 통계적 자료 분석의 결과를 왜곡시키거나 자료 분석의 적절성을 위협하는 변숫값을 뜻하기도 한다.



왼쪽에 혼자 떨어져 있는 파란 점과, 오른쪽에 혼자 떨어져 있는 빨간 점이 이상치이다.

위쪽 그림은 서포트 벡터와 결정 경계 사이의 거리가 매우 좁아져서 마진이 매우 작아진 상태이다. 이런 상태를 하드 마진(hard margin)이라고 한다. 이상치를 허용하지 않는 기준으로 결정 경계를 정해버리면 과대적합(overfitting) 문제가 발생할 수 있다.

아래쪽 그림은 이상치들이 마진 안에 어느 정도 포함되도록 기준을 잡았다. 서포트 벡터와 결정 경계 사이의 거리가 멀어져서 마진이 커진 상태이다. 이런 상태를 소프트 마진(soft margin)이라고 한다. 이 경우 과소적합(underfitting) 문제가 발생할 수 있다.

문제정의

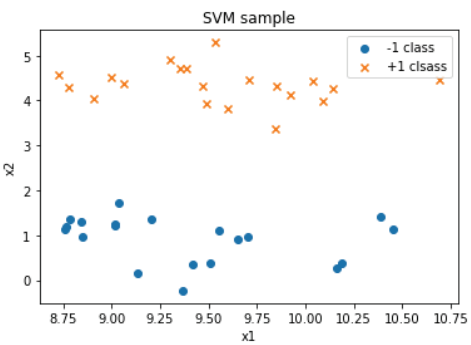
분리된 두 클래스를 선형으로 가장 적절하게 결정 경계를 정하도록 하자.

DATASET

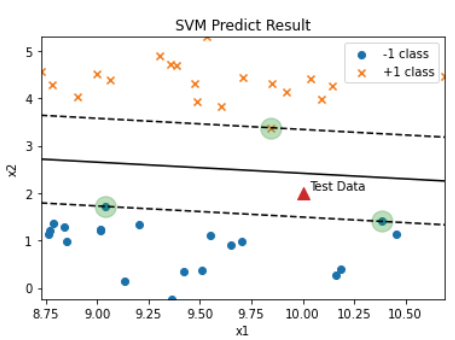
데이터 등 관련 자료와 설명

각 클래스를 구분하기 위해 'O'와 'X'로 데이터를 표시하고, 'O'는 '-1 class'로 'X'는 '+1 class'로 구분하여 표시한다. 샘플데이터는 각 클래스당 20개로 설정한다.

```
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=40, centers=2,
        cluster_std=0.5, random_state=4)
y = 2 * y - 1
plt.scatter(X[y == -1, 0], X[y == -1, 1],
  marker='o', label="-1 class")
plt.scatter(X[y == +1, 0], X[y == +1, 1],
  marker='x', label="+1 class")
plt.xlabel("x1")
plt.ylabel("x2")
plt.legend()
plt.title("SVM sample")
plt.show()
```




```
import numpy as np
from sklearn.svm import SVC
model = SVC(kernel='linear', C=1e10).fit(X, y)
xmin = X[:, 0].min()
xmax = X[:, 0].max()
ymin = X[:, 1].min()
ymax = X[:, 1].max()
xx = np.linspace(xmin, xmax, 10)
yy = np.linspace(ymin, ymax, 10)
...
plt.scatter(X[y == +1, 0], X[y == +1, 1],
marker='x', label="+1 class")
plt.contour(X1, X2, Z, levels, colors='k',
linestyles=linestyles)
plt.scatter(model.support_vectors_[:, 0],
model.support_vectors_[:, 1], s=300, alpha=0.3)
x_new = [10, 2]
plt.scatter(x_new[0], x_new[1], marker='^',
s=100)
plt.text(x_new[0] + 0.03, x_new[1] + 0.08, "Test
Data")
plt.xlabel("x1")
plt.ylabel("x2")
plt.legend()
plt.title("SVM Predict Result")
plt.show()
```



〈SVM 예측 결과〉

Scikit-Learn의 SVM 패키지는 서포트 벡터 머신 모형인 SVC(Support Vector Classifier) 클래스를 제공한다.

문제정의

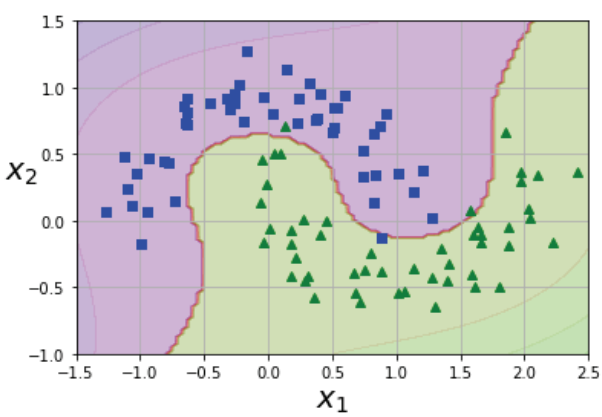
분리된 두 클래스가 비선형인 경우 결정 경계를 정하도록 하자.

DATASET

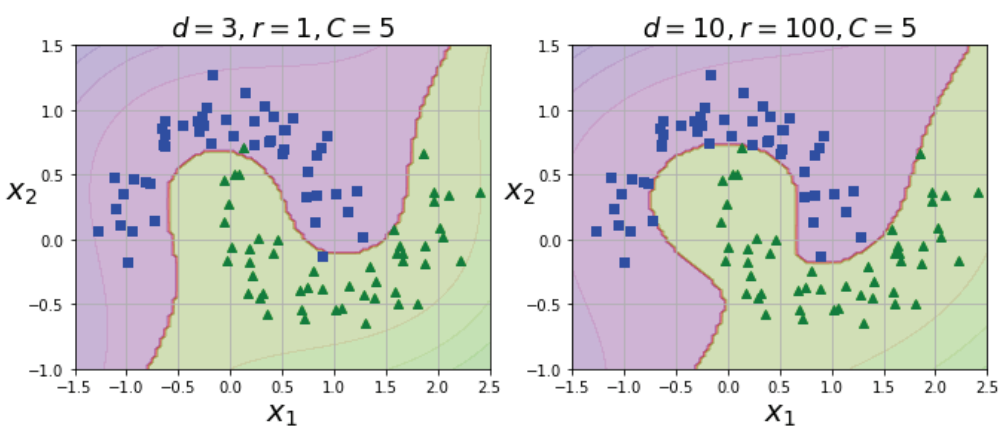
데이터 등 관련 자료와 설명

하나의 클래스는 '□' , 다른 클래스는 '△'로 구분하여 표시한다. 커널은 다항식(poly)을 사용한다.

```
import numpy as np
from matplotlib import pyplot as plt
def plot_dataset(X, y, axes):
    plt.plot(X[:, 0][y==0], X[:, 1][y==0], "bs")
    plt.plot(X[:, 0][y==1], X[:, 1][y==1], "g^")
    plt.axis(axes)
    plt.grid(True, which='both')
    plt.xlabel(r"$x_1$", fontsize=20)
    plt.ylabel(r"$x_2$", fontsize=20, rotation=0)
def plot_predictions(clf, axes):
    x0s = np.linspace(axes[0], axes[1], 100)#axes[0]부터 axes[1]까지 100개로 이루어진 숫자들
    x1s = np.linspace(axes[2], axes[3], 100)
    x0, x1 = np.meshgrid(x0s, x1s) #표현할 수 있는 모든 배열조합
    X = np.c_[x0.ravel(), x1.ravel()]# ravel 1차원 배열로 핀다.
    y_pred = clf.predict(X).reshape(x0.shape)
    y_decision = clf.decision_function(X).reshape(x0.shape)
    plt.contourf(x0, x1, y_pred, cmap=plt.cm.brg, alpha=0.2)
    plt.contourf(x0, x1, y_decision, cmap=plt.cm.brg, alpha=0.1)

# 다항식(polynomial) 커널을 사용하면 데이터를 더 높은 차원으로 변형하여 나타냄으로써 초
평면(hyperplane)의 결정 경계를 얻을 수 있다.
plot_predictions(polynomial_svm_clf, [-1.5, 2.5, -1, 1.5])

plot_dataset(X, Y, [-1.5, 2.5, -1, 1.5])
plt.show()
```



```
from sklearn.svm import SVC
poly_kernel_svm_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5))
])
#coef0는 모델이 높은 차수와 낮은 차수에 얼마나 영향을 끼치는지 정할 수 있다.
poly_kernel_svm_clf.fit(X, Y)

plt.figure(figsize=(11, 4))
plt.subplot(121)
plot_predictions(poly_kernel_svm_clf, [-1.5, 2.5, -1, 1.5])
plot_dataset(X, Y, [-1.5, 2.5, -1, 1.5])
plt.title(r"$d=3, r=1, C=5$", fontsize=18)

poly_kernel_svm100_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="poly", degree=10, coef0=100, C=5))
])
poly_kernel_svm100_clf.fit(X, Y)

plt.subplot(122)
plot_predictions(poly_kernel_svm100_clf, [-1.5, 2.5, -1, 1.5])
plot_dataset(X, Y, [-1.5, 2.5, -1, 1.5])
plt.title(r"$d=10, r=100, C=5$", fontsize=18)
plt.show()
```

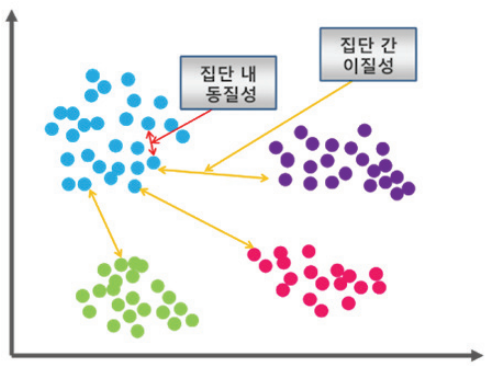


매개변수인 coef( ) 값을 1에서 100으로 변경할 경우 결정 경계가 오른쪽 그림과 같이 달라짐을 알 수 있다.

6 군집분석(Clustering Analysis)

군집분석

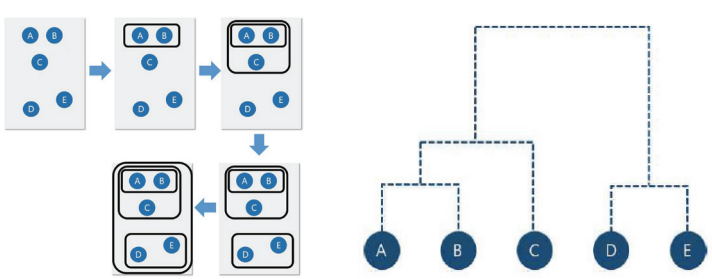
분류 기준이 없는 상태에서 데이터 속성을 고려해 스스로 전체 데이터를 N개의 소그룹으로 묶어내는(Clustering) 분석법이다. 유사성이 높은 대상 데이터를 묶고, 서로 다른 그룹에 속한 데이터 개체 간의 이질성을 계산하는 과정을 거친다.



〈군집분석 예시〉
군집화는 집단 내 동질성과 이질성을 기반으로 이뤄진다.

계층적 군집화(Hierarchical Clustering)

가장 유사한 개체를 묶어 나가는 과정을 반복하여 원하는 개수의 군집을 형성하는 방법이다. 계층적 군집을 형성하는 방법에는 작은 군집으로부터 출발하여 군집을 병합해 나가는 병합적 방법과 큰 군집으로부터 출발하여 군집을 분리해 나가는 분할적 방법이 있다.



왼쪽 그림은 병합적 방법으로 군집을 형성해 나가는 과정을 보여주고 있다.
계층적 군집의 결과는 오른쪽 그림과 같이 덴드로그램(dendrogram)의 형태로 표현된다.
이 그림을 통해 군집들 간의 구조적 관계를 파악할 수 있다. 항목 간의 거리, 군집 간의 거리를 알 수 있고, 군집 내의 항목 간 유사 정도를 파악함으로써 군집의 견고성을 해석할 수 있다.

계층적 군집 종류

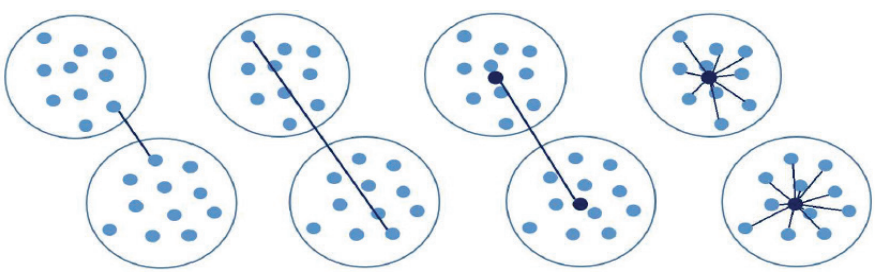
1) 최단 연결법(single linkage, nearest neighbor) :
한 군집의 점과 다른 군집의 점 사이에 가장 짧은 거리로 군집 형성

2) 최장 연결법(complete linkage, farthest neighbor) :
한 군집의 점과 다른 군집의 점 사이에 가장 긴 거리로 군집 형성

3) 중심 연결법(centroid linkage) : 두 군집의 중심 간의 거리로 군집 형성

4) 평균 연결법(average linkage) : 한 군집의 점과 다른 군집의 점 사이의 거리에 대한 평균을 사용하여 군집 형성

5) 와드 연결법(ward linkage) :
군집 간 거리에 기반하는 다른 연결법과는 달리, 군집 내 오차제곱합(ESS, error sum of square)을 고려하여 군집 형성(군집 간 정보의 손실을 최소화하기 위해 군집화)



〈최단 연결법〉 〈최장 연결법〉 〈중심 연결법〉 〈와드 연결법〉

비계층적 군집(Non-Hierarchical Clustering Method)

구하고자 하는 군집의 수를 정한 상태에서 설정된 군집의 중심에 가장 가까운 개체를 하나씩 포함해 가는 군집 형성 방법이다.

비계층적 군집 종류

1) K-평균 군집(K-means clustering) : 사전에 결정된 군집 수 K에 기초하여 전체 데이터를 상대적으로 유사한 K개의 군집으로 구분하는 방법이다. 가장 대표적인 방법이며, 순차적 군집 분석법(Sequential Threshold Method)이라고도 한다.

2) 동시 군집 분석법(Paralleled Threshold Method) : 사전에 지정된 값 안에 관측 대상이나 속성이 속하는 경우나 몇 개의 군집이 동시에 결정되는 경우

3) 최적 할당 군집 분석법(Optimizing Partitioning Method) : 사전에 주어진 군집의 수를 위한 군집 내 평균거리를 계산하는 최적화 기준에 의거하여 최초의 군집에서 다른 군집으로 다시 할당하는 방법

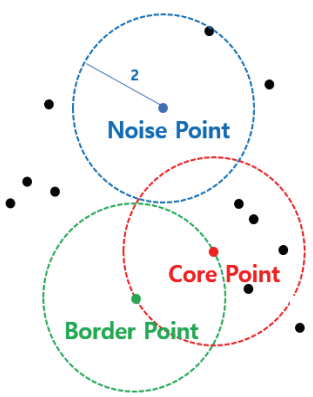
K-평균 군집(K-means clustering) 과정

1) 클러스터 개수 K값 결정
2) 데이터 공간에 클러스터 중심 K개 할당
3) 각 클러스터 중심을 해당 클러스터 데이터의 평균으로 조정
4) 클러스터 중심이 변하지 않을 때까지 반복

DBSCAN 클러스터링

DBSCAN의 경우 '밀도 기반'으로 군집을 할당한다. 여기서 밀도 기반이란, 어떤 데이터 포인트에서의 반지름 x내에 n개 이상의 포인트를 갖는 것을 하나의 군집으로 구분하는 것이다. 즉 K-Means와는 달리 군집의 개수 k를 미리 정의해놓을 필요는 없지만 반지름과 한 군집 내에 최소 n개의 포인트가 있어야 하므로 이 두 개를 사전에 정의해야 한다.

오른쪽 그림에서 반지름이 2이고, 최소 6개의 포인트를 가질 때 1개의 군집으로 할당하고자 한다. 이때 빨간 점은 해당 조건을 만족하여 코어 포인트가 된다. 보더 포인트란 코어 포인트의 반지름 내에 걸쳐서 '이웃'하고 있으나 최소 포인트 개수를 충족시키지 못하는 경우(초록색)를 말한다. 노이즈 포인트는 코어 포인트도 아니고 보더 포인트도 아닌 포인트를 말한다. 모든 데이터 포인트에서 해당 점이 코어 포인트가 되는지 확인하면서 군집을 할당하는 것이 DBSCAN 알고리즘의 핵심이다. K-Means와 달리 어느 군집에도 속하지 않는 노이즈 포인트는 제외하며 남아있는 포인트를 중심으로 군집을 할당하는 것이 특징이다. 노이즈 포인트를 버리기 때문에 노이즈에 취약하지 않으며, 군집의 모양과 사이즈가 다양하게 나올 수 있다.



DBSCAN에서 최적의 반지름과 최소 군집 개수 찾는 법

이제 최적의 반지름과 최소 군집 개수를 찾아보자. 이상적인 케이스를 생각하면, 하나의 군집 안에 있는 데이터 포인트의 거리는 가깝고, 노이즈 포인트와의 거리는 꽤 멀게 떨어져 있을 것이다. 어떤 군집 안에 있는 데이터 포인트들과 'k번째 인접한 이웃 데이터 포인트까지의 거리'를 계산해보며 반지름을 찾아나간다. 즉, 그 거리가 급격하게 늘어날 때까지의 점을 찾는다. 여기서 k는 최소 군집 개수이며, k번째 인접한 이웃 데이터 포인트까지의 거리는 반지름이 된다.

문제정의

계층적 군집 방법에 대해 알아보자.

DATASET

붓꽃 꽃받침 길이(Sepal Length), 꽃받침 폭(Sepal Width), 꽃잎 길이(Petal Length), 꽃잎 폭(Petal Width)

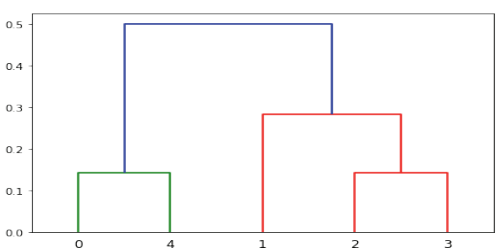
참고 SOURCECODE

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
#계층적 군집화
iris = load_iris()
#print(iris)
iris_df = pd.DataFrame(iris.data, columns =
iris.feature_names)
print(iris_df.head(5))
from scipy.spatial.distance import pdist, squareform
distmatrix = pdist(iris_df.loc[0:4, ["sepal length (cm)",
"sepal width (cm)"]],
                   metric = "euclidean")
#print("distmatrix: ", distmatrix)
#squareform을 이용해 거리 값을 matrix로 형태로 표현
row_dist = pd.DataFrame(squareform(distmatrix))
print("row_dist: ", row_dist)
```

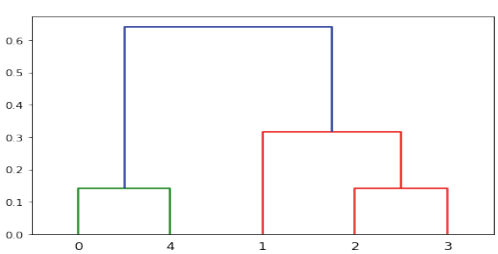
	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	0	1	2	3	4
0	0.000000	0.538516	0.500000	0.640312	0.141421
1	0.538516	0.000000	0.282843	0.316228	0.608276
2	0.500000	0.282843	0.000000	0.141421	0.500000
3	0.640312	0.316228	0.141421	0.000000	0.640312
4	0.141421	0.608276	0.500000	0.640312	0.000000

```
#단일(최단) 연결법
from scipy.cluster.hierarchy import linkage,
dendrogram
r_cluster = linkage(distmatrix, method = "single")
print("r_cluster: ", r_cluster)
df = pd.DataFrame(r_cluster, columns = ["id_1",
"id_2", "거리", "멤버 수"])
#print(df)
row_dend = dendrogram(r_cluster)
plt.tight_layout()
plt.show()
```




```
# 완전(최장) 연결법
from scipy.cluster.hierarchy import linkage,
dendrogram
r_cluster = linkage(distmatrix, method = "
complete")
print("r_cluster: ", r_cluster)
df = pd.DataFrame(r_cluster, columns = ["id_1",
"id_2", "거리", "멤버 수"])
#print(df)
row_dend = dendrogram(r_cluster)
plt.tight_layout()
plt.show()
```



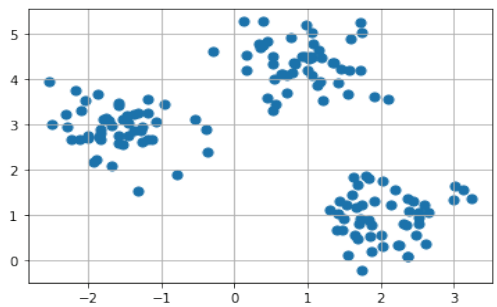
X축의 특성은 동일하게 군집하지만,
단일 연결법과 완전 연결법은 각각 Y축이 0.5일 때와
0.6일 때 연결되는 차이를 보임

문제정의

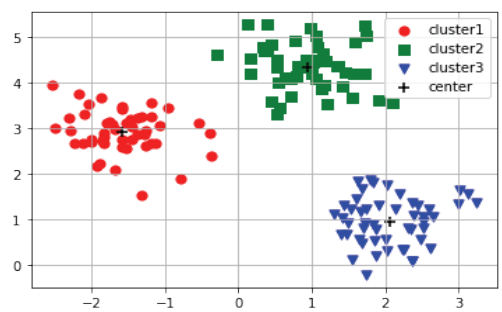
비계층적 군집 가운데 K-평균 군집(K-means Clustering)에 대해 알아보자.

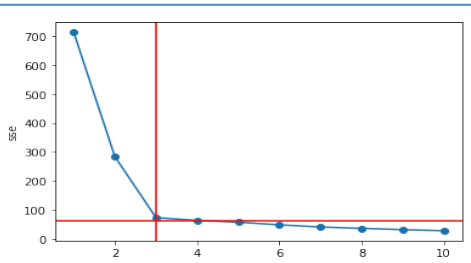
참고 SOURCECODE

```
#초기 Data 시각화
from sklearn.datasets import make_blobs
print(make_blobs)
x, y = make_blobs(n_samples = 150, n_features = 2,
centers = 3,
                  cluster_std = 0.5, shuffle = True,
random_state = 0)
print("x.shape: ", x.shape, ", y.shape: ", y.shape)

plt.scatter(x[:, 0], x[:,1], marker = "o", s = 50)
plt.grid()
plt.show()
```



```
#Kmeans Clustering
from sklearn.cluster import Kmeans
# 초기 클러스터 중심은 임의로 할당
init_centroid = "random"
#init_centroid = "k-means++" # 기본값
kmodel = KMeans(n_clusters = 3, init =
init_centroid, random_state = 0)
#print(kmodel)
pred = kmodel.fit_predict(x)
#print("pred: ", pred)
plt.scatter(x[pred == 0, 0], x[pred == 0, 1], marker
= "o", s = 50, c = "red", label = "cluster1")
plt.scatter(x[pred == 1, 0], x[pred == 1, 1], marker
= "s", s = 50, c = "green", label = "cluster2")
plt.scatter(x[pred == 2, 0], x[pred == 2, 1], marker
= "v", s = 50, c = "blue", label = "cluster3")
plt.scatter(kmodel.cluster_centers_[:,0],
kmodel.cluster_centers_[:,1],
            marker = "+", s = 80, c = "black", label =
"center")
plt.legend()
plt.grid()
plt.show()
```






```
#n-clusters를 알기위한 방법
#1 엘보우 기법: 클러스터 내 오차제곱합이 최소
가 되도록 클러스터의 중심을 결정해 나가는 방
법
def elbow(x):
  sse = []
  for i in range(1, 11):
    km = KMeans(n_clusters = i,
            init = "k-means++", random_state =
0)
    km.fit(x)
        sse.append(km.inertia_)
    print(sse[3])
    plt.plot(range(1, 11), sse, marker = "o")
    plt.axvline(x=3,color='r')
    plt.axhline(y=sse[3],color='r')
    plt.xlabel("Number of Cluster")
    plt.ylabel("sse")
    plt.show()
elbow(x)
# 클러스터의 개수가 3일 때 팔꿈치 부분이므로,
최적의 클러스터 개수는 3으로 결정됨
```

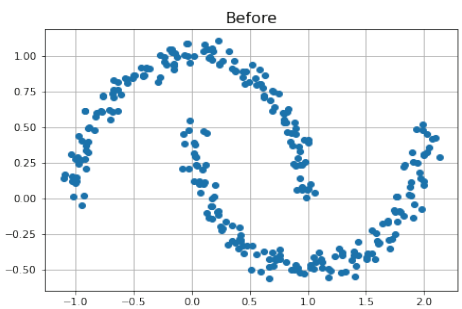
문제정의

계층적 군집 방법에 대해 알아보자.

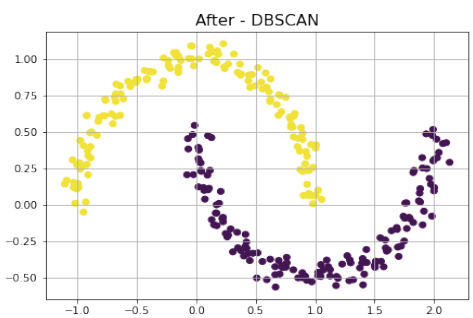
밀도 기반 군집화의 명확한 이해를 위해, 예제 데이터로 Moon 데이터 세트를 활용하였다. Moon 데이터는 make_moons 함수로 생성할 수 있으며, 샘플 수(n_samples)와 분산 정도(noise)를 조절해서 생성할 수 있다.

참고 SOURCECODE

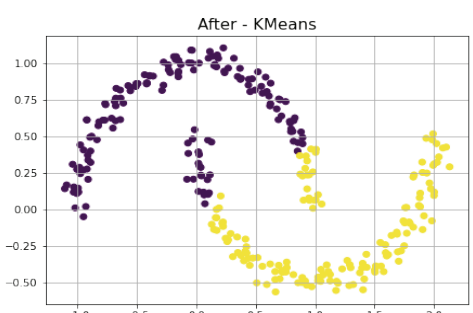
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
x,y = make_moons(n_samples=300,
noise=0.05, random_state=42)
df=pd.DataFrame(x)
#df.head()
#산점도
plt.figure(figsize=(7,5))
plt.title("Before", fontsize=15)
plt.plot(df[0], df[1], "o")
plt.grid()
plt.show()
```

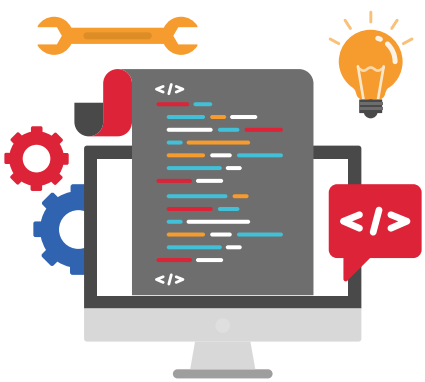



```
#DBSCAN함수로 클러스터링을 수행해보자
#eps : 반경 설정(속성값의 단위)
#min_samples : 최소 개체 수
from sklearn.cluster import DBSCAN
db_scan = DBSCAN(eps=0.3,
min_samples=5).fit(df.values)
df['cluster_db'] = db_scan.labels_
plt.figure(figsize=(7,5))
plt.title("After - DBSCAN", fontsize=15)
plt.scatter(df[0],df[1],c=df['cluster_db'])
plt.grid()
plt.show()
```




```
#K-means와 DBSCAN 비교
#위와 동일한 Moon 데이터 세트로 KMeans를
활용해 작업을 수행해본 결과이다.
from sklearn.cluster import KMeans
kmeans_ = KMeans(n_clusters=2,
random_state=42).fit(df.values)
df['cluster_km'] = kmeans_.labels_
plt.figure(figsize=(7,5))
plt.title("After - KMeans", fontsize=15)
plt.scatter(df[0],df[1],c=df['cluster_km'])
plt.grid()
plt.show()
```





Ⅳ. 딥러닝

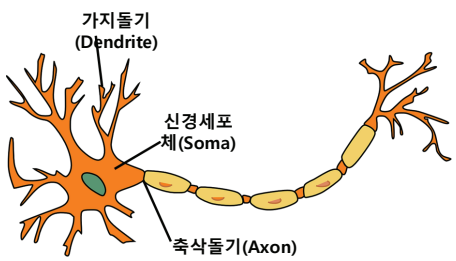
1. 인공신경망(Artificial Neural Network)
2. 심층신경망(Deep Neural Network)
3. 합성곱신경망(Convolutional Neural Network)
4. 순환신경망(Recurrent Neural Network)

Ⅳ. 딥러닝

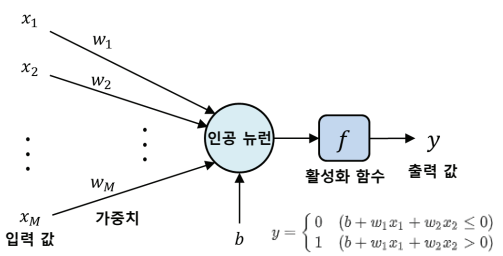
1 인공신경망(Artificial Neural Network)

핵심요약

인공신경망(ANN)은 기계학습과 인지과학 분야에서 고안한 학습 알고리즘이다. 신경세포의 신호 전달체계를 모방한 인공뉴런(노드)이 학습을 통해 결합 세기를 변화시켜 문제를 해결하는 모델 전반을 가리킨다.



〈신경세포〉

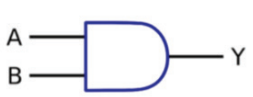


〈퍼셉트론〉

인공신경망의 시작, 퍼셉트론

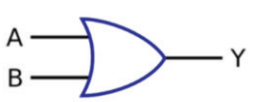
퍼셉트론은 신경세포 뉴런들이 신호, 자극 등을 받아 어떠한 임계값(threshold, $\boldsymbol{\theta}$)을 넘어서면 그 결과를 전달하는 신경세포의 신호 전달 과정을 착안하여 만들어졌다. 이를 수학적인 기호로 모델링 한 것이 퍼셉트론인데, 퍼셉트론은 각각의 가중치(ω_i)의 크기를 적절히 조절하여 입력 신호(χ_i)의 크기를 정한다. 입력 신호들의 합에 활성화 함수를 거쳐 나온 결괏값을 전달함으로써 원하는 결괏값을 얻는 방식으로 동작한다.

아래 그림은 단층 퍼셉트론으로 AND, OR, NAND와 같은 비교적 간단한 문제를 해결할 수 있는 예시이다. 입력값에 대해 정해진 결괏값을 출력할 수 있는 가중치의 경우의 수는 무수히 많으며(퍼셉트론 그림에서 출력 식 참고), 주어진 연산을 만족하는 가중치의 조합을 찾음으로써 문제를 푼다고 할 수 있다.



χ_1	χ_2	y
0	0	0
1	0	0
0	1	0
1	1	1

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$



χ_1	χ_2	y
0	0	0
1	0	1
0	1	1
1	1	1

$(w_1, w_2, \theta) = (0.5, 0.5, 0.2)$

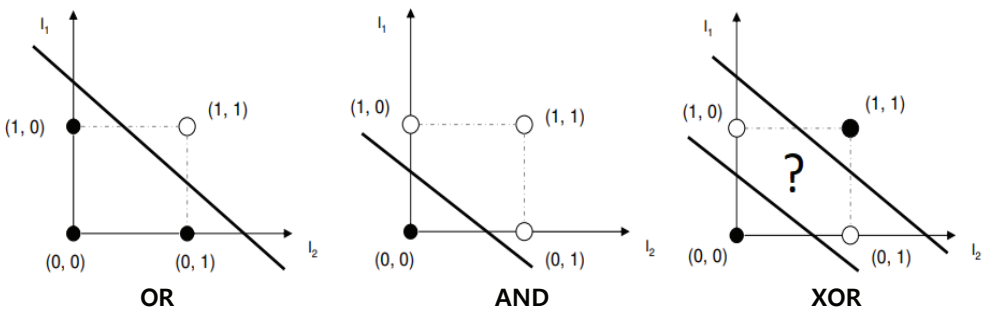
χ_1	χ_2	y
0	0	1
1	0	1
0	1	1
1	1	0

$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$

〈단층 퍼셉트론에서 AND, OR, NAND 문제 및 가중치 예시 〉

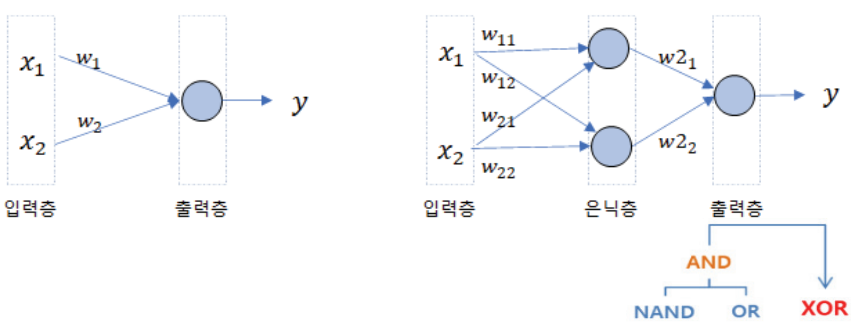
단층 퍼셉트론의 한계

앞서 살펴본 단층 퍼셉트론으로 AND, OR, NAND(=NOT AND)와 같은 비교적 단순한 논리 게이트는 구현할 수 있었지만 XOR같은 문제를 풀 수 없는 한계가 있었다. 이는 좌표평면상에 결정 경계(Decision Boundary)를 그어보면 좀 더 명확해지는데, 단층 퍼셉트론으로는 비선형적인 결정 경계를 표현하는 데 한계가 있다는 것을 알 수 있다.



다층 퍼셉트론의 등장

이와 같은 문제를 해결하기 위해 다층 퍼셉트론이 등장하였다. NAND와 OR연산의 결괏값을 다시 AND하여 XOR연산을 표현할 수 있다는 점에서, 입력층과 출력층 사이에 은닉층을 두어 다층 퍼셉트론을 만들면 좀 더 복잡한 문제를 해결할 수 있다는 것을 발견하였다. 현재 다양한 종류의 인공신경망은 모두 입력층, 은닉층, 출력층으로 나누어 구성된다.



x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

다층으로 해결

x_1	x_2	s_1 (NAND)	s_2 (OR)	y
0	0	1	0	0
1	0	1	1	1
0	1	1	1	1
1	1	0	1	0

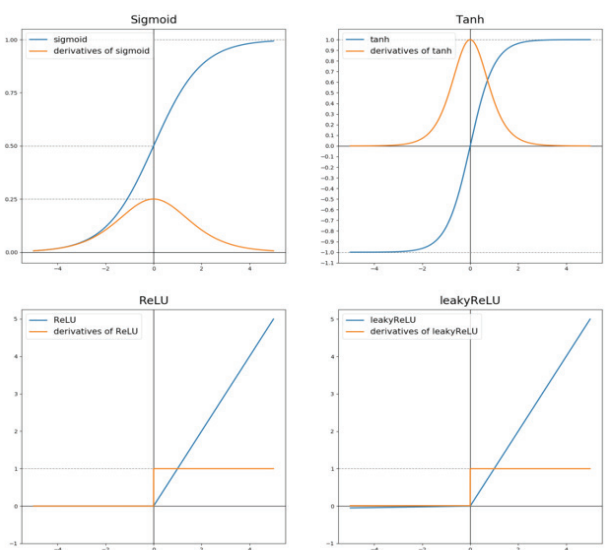
활성화 함수(Activation Function)

앞서 살펴본 것처럼 은닉층을 여러 층 두면 단순한 선형이 아니라 좀 더 복잡한 분류기를 만들 수 있는 가능성을 보았다. 하지만 단순히 은닉층(Hidden Layer)을 쌓는 것만으로는 부족하다. 왜냐하면 출력값이 다음 입력값으로 넘어갈 때 선형식(Linear)으로 전파된다면 은닉층을 여러 개 두는 방식이나 하나를 두는 것이나 차이가 없기 때문이다. 이러한 이유로 출력층의 결괏값을 비선형(Non Linear)으로 변환시켜 줄 장치인 '활성화 함수(Activation Function)'를 두게 되었다.

인공신경망에는 다양한 활성화 함수(Activation Function)가 존재한다. 대표적으로 Sigmoid, Tanh, ReLU 등이 있다. Sigmoid의 경우 양 끝에서 기울기가 거의 0에 수렴하고, 이로 인한 기울기 소실(Vanishing Gradient)문제가 생기기 때문에 대부분 인공신경망은 활성화 함수로 ReLU 함수를 사용한다.

ReLU함수는 0보다 작은 구간에서 기울기가 0이고 나머지 0보다 큰 구간에서는 항상 일정한 크기의 기울기를 가지는 특성을 갖고 있기 때문에 기울기 소실 문제를 막아 좋은 학습 결과를 기대할 수 있다.

활성화 함수는 인공신경망이 학습을 잘할 수 있도록 하는 하나의 수학적 장치이며 모델 설계 시 다양하게 실험하여 좋은 것을 선택하면 된다.



〈활성화 함수와 도함수의 그래프〉

활성화 함수가 비선형인 이유?

$f(x) = cx$ 일때, 은닉층이 3개 라면, 3개 층을 거친 결괏값은 $f(f(f(x)))$이 될 것이고
이는 $g(x) = cx^3$인 하나의 은닉층과 같은 결과이므로 은닉층을 여러 개 둔 효과를 기대하기 어렵다.

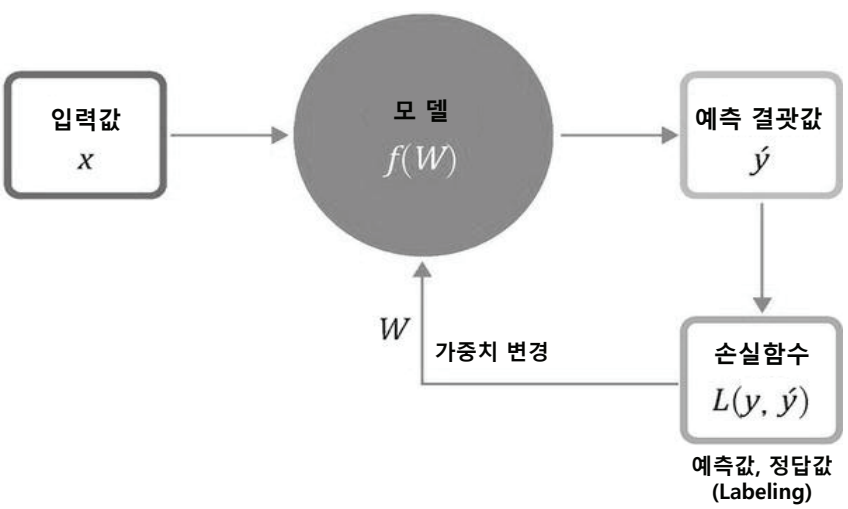
학습(Learning)

다층 퍼셉트론으로 좀 더 어려운 문제도 풀 수 있다는 것을 알게 되었지만 아직 중요한 부분이 남아 있다. 바로 학습이다. 아직까지는 입력값에 원하는 결괏값이 나올 수 있도록, 적절한 가중치(ω_i)를 사람이 직접 넣었기 때문에 학습을 한다고 할 수 없다. 학습을 위해서는 인공신경망이 데이터를 잘 표현(Representation)하도록 무수히 많은 가중치(ω_i)들의 조합을 스스로 찾을 수 있는 과정 및 장치들이 필요하다. 인공신경망이 학습하는 과정을 알아보자.

손실 함수(Loss Function)

인공신경망이 수많은 학습데이터를 이용하여 모델을 잘 표현하기 위해서는 입력데이터에 대한 결괏값(Output)이 원하는 정답값(Labeling)과 얼마나 다른지 모델에 끊임없이 피드백(Feedback)하는 과정이 필요하다. 손실 함수는 바로 이 과정에서 학습데이터가 모델(입력층→은닉층→출력층)을 통해 나온 결괏값이 실제로 우리가 원하는 값(Labeling)과 얼마나 다른지 정도를 측정하기 위해 사용하는 함수를 말하며 이를 토대로 가중치를 변경하게 된다. 같은 의미로 비용함수(Cost Function)라고 표현하기도 한다.

손실 함수(Loss Function)는 문제의 유형에 따라 대표적으로 평균제곱오차(Mean Squared Error)와 교차엔트로피 오차(Cross Entropy Error)가 있는데,두 가지 모두 입력에 대한 예측값(Prediction)과 정답값(Labeling)이 비슷할수록 작은 값이 나오게 설계되어 있다.

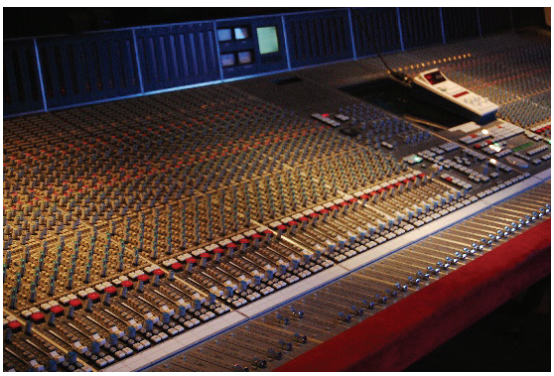


〈학습 : 손실함수를 통한 가중치 변경〉

가중치 변경

딥러닝 분야의 세계적인 권위자이자 페이스북 AI연구 이사 얀 르쿤(Yann LeCun) 교수는 신경망이 가중치의 조합을 찾는 것(학습)을 최적의 소리를 찾기 위해 음향장치의 수많은 조절기의 상태를 기계 스스로 찾는 것에 비유하기도 하였다. 그렇다면 이 조절기는 어떤 방법으로 조절하고 찾을 수 있을까?

〈얀 르쿤(Yann LeCun) 교수〉

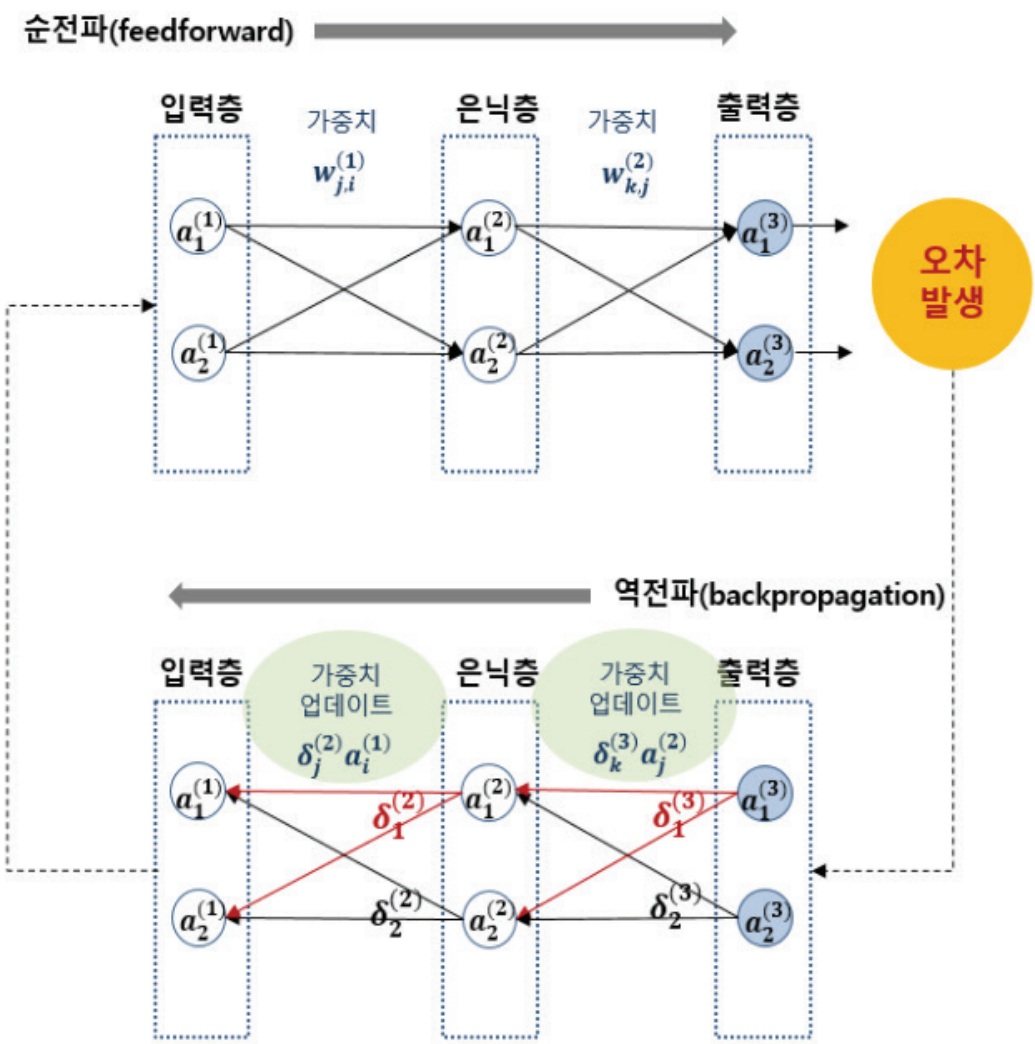


음향기기(Sound Control Mixer)

순전파(Feedforward), 역전파(Backpropagation), 최적화(Optimization)

앞에서 인공신경망이 수많은 가중치를 변경하여 데이터를 잘 표현하는 가중치를 찾아 문제를 해결하며, 가중치를 변경할 때 참고하는 도구로써 손실 함수(Loss Function)를 사용한다는 것을 살펴보았다.
그렇다면 우리의 고민은 이렇게 정리할 수 있다. '어떻게 하면 손실 함수(Loss Function)의 값을 최소로 만들 수 있을까?(예측값과 정답값을 비슷하게 하는 가중치의 조합을 찾기)' 인공신경망은 최적화(Optimization)와 역전파(Backpropagation)의 방식을 이용한다. 손실 함수가 각각의 가중치를 변수로 하는 함수라는 점을 이용하여, 손실 함수의 값을 각각 가중치들로 국소적 미분하고 지역적 최솟값(Local Minimum)으로 하는 가중치들을 찾는다. 이때 가중치는 연쇄 법칙(Chain Rule)을 사용하여 역전파(Backpropagation)법으로 최솟점을 찾는 기울기 하강(Gradient Descent)을 하며 변경(Update)된다.

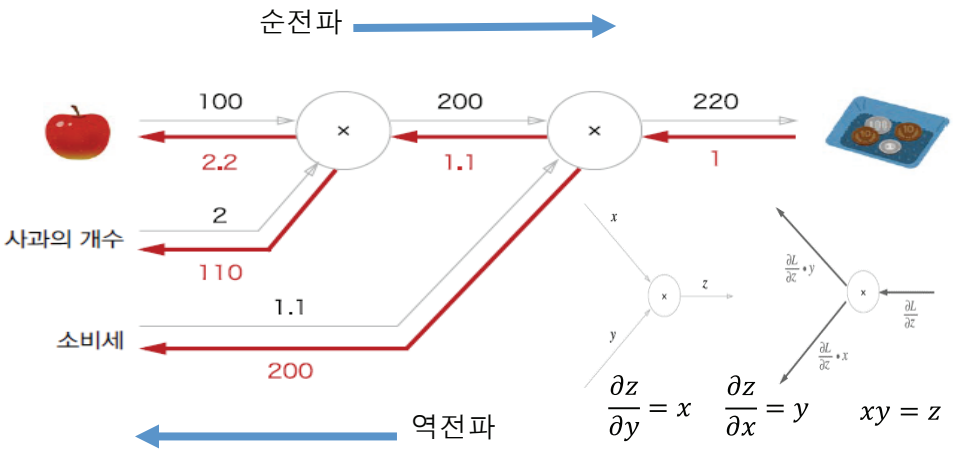
인공신경망은 모델이 예측 결괏값을 찾는 과정이 입력층-은닉층-출력층으로 순차적으로 흐른다고 하여 순전파(Feedforward)라 하며 가중치의 수정은 출력층-은닉층-입력층으로 흐르기 때문에 역전파(Backpropagation)라고 한다. 또한 최적의 가중치를 찾아가는 방식, 그러한 알고리즘을 최적화(Optimization)라고 한다.



〈순전파(Feedforward)와 역전파(Backpropagation)〉

역전파(Backpropagation) 이해하기

사과의 최종 가격이 어떤 요소에 의해서 영향을 받는다고 하자. 만약 다른 요소를 통제한 상태에서 하나의 요인이 최종 사괏값에는 어떠한 영향을 끼치는지 알고자 할 때, 다음과 같이 계산 그래프를 그려 결과를 도출할 수 있다.



곱셈의 역전파

1) 주어진 식(사과 단가, 개수, 소비세)에 따라 계산 그래프를 구성한다.

2) 그래프에서 계산을 왼쪽에서 오른쪽으로 진행한다. : 100원 * 2개 * 1.1 = 220원

3) 국소적 미분값을 표현하여 단계마다 표시한다.

4) 각 요소(단가, 개수, 소비세)에 대한 영향도의 크기를 계산한다.

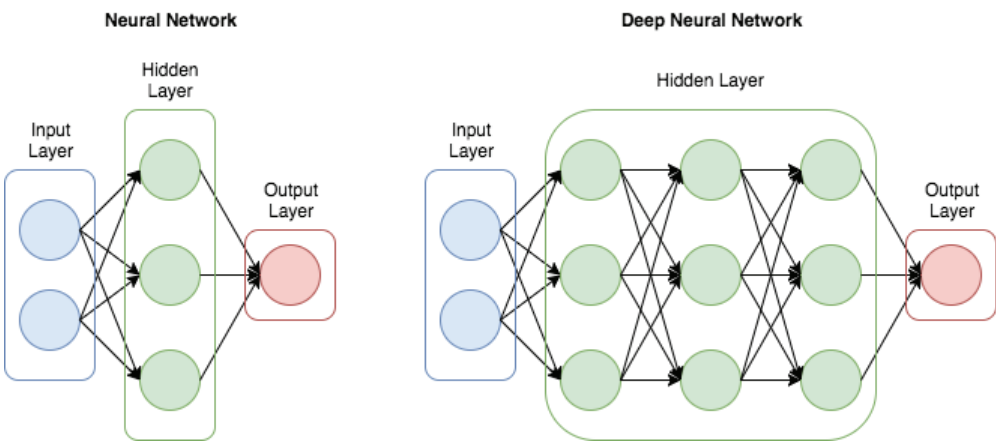
100원짜리 사과 2개가 소비세 10%를 더해 220원이라고 할 때, 우리는 계산 그래프를 통해 다음과 같이 해석할 수 있다. 사과가 1원 오르면 최종 금액은 2.2원 오르고, 사과의 개수가 하나 더 생길 때마다 총액은 110원이 오른다. 즉, 각 요소들이 총 금액에 대해 얼마만큼 영향을 끼치는지 알 수 있는 것이다.
이를 인공신경망에 적용하면 수많은 각각의 가중치의 값을 편미분(Partial Derivative)하여 비용함수(Cost Function)로부터 발생하는 값이 최소가 되는 가중치값을 구할 수 있고, 이는 결과적으로 데이터를 가장 잘 표현(Representation)하는 모델을 만드는 것이 된다.

〈출처 : 밑바닥부터 시작하는 딥러닝(저자 : 사이토 고키)〉

❷ 심층신경망(Deep Neural Network)

핵심요약

심층신경망(Deep Neural Network)은 인공신경망(Artificial Neural Network)과 동일한 구조와 동작 방식을 갖고 있다. 심층신경망은 단지 인공신경망에서 은닉층(Hidden Layer)의 깊이가 깊어진 형태를 말한다.



신경망의 발전계기

인공신경망의 기본 아이디어가 나온 것은 1950년대이고 이후에 발전을 거듭하지만 불과 2000년대 중반까지만 해도 사람들로 하여금 그다지 알려지고 활용되던 기술이 아니었다. 여기에는 몇 가지 이유가 있는데 하나는 당시에 GPU와 같은 수많은 가중치를 빠르게 계산할 수 있는 자원(Resource)이 충분치 않았고, 사람이 원하는 수준으로 성능을 올리기 위해서 필요했던 충분한 양의 데이터(Data)를 모으기에도 인프라가 부족했다. 또한 모델의 성능향상과 효율성의 측면에서 획기적인 도움을 주었던 역전파(Backpropagation) 및 최적화(Optimization)와 같은 효율적인 알고리즘(Algorithm)이 없었기 때문이었다.

딥러닝(Deep Learning)

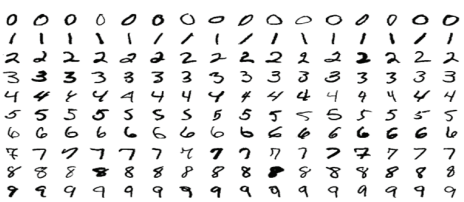
반도체 기술의 발전으로 GPU 자원이 상용화되고, 정보화 사회를 거치며 충분한 데이터를 모을 수 있는 인프라가 마련되었으며 학자들이 수학적 기법을 이용하여 다양한 학습 알고리즘을 개발하면서 이러한 인공신경망의 문제점을 극복할 수 있었다. 그리고 마침내 오랫동안 부정적이었던 이미지를 가진 인공신경망(Artificial Neural Network)이라는 용어를 대체할 '깊이 학습한다'는 의미를 지닌 딥러닝(Deep Learning)이라는 용어를 만들게 되었다.

문제정의

인공신경망 라이브러리 텐서플로우(Tnesorflow)와 케라스(Keras)를 이용하여 신경망 모델을 구성하여 실습을 진행한다. 숫자 데이터(MNIST)를 인공신경망에 학습시켜 임의의 숫자 데이터에 대해 얼마나 예측을 잘하는지 확인하는 방법으로 일반적인 기계학습의 학습(Learning)-평가(Evaluation) 과정과 동일하다.

DATASET

MNIST(Modified National Institute Standard and Technology)-미국표준기술연구소에서 제공하는 손으로 쓴 숫자 데이터베이스이다. 해상도는 28x28 픽셀(pixel)이고 흑백(grayscale)로 한 픽셀당 0~255 사이의 값을 가진다.



학습 데이터 : 60,000개
시험 데이터 : 10,000개

1) 신경망 구성 및 기계학습에 필요한 라이브러리 추가

```
import numpy as np # 수치연산 라이브러리
import pandas as pd # 데이터분석용
import matplotlib.pyplot as plt # 그래프

import mnist # 데이터셋

import tensorflow
from tensorflow.python.client import device_lib
import keras

from keras.models import Sequential
from keras.layers import Dense, Activation, BatchNormalization
from keras.losses import categorical_crossentropy# 교차엔트로피 # 손실함수
from keras.optimizers import SGD, Adagrad, Adam # (확률적)경사하 강 # 2015+ 최적화 알고리즘
from keras.utils.multi_gpu_utils import multi_gpu_model
from keras.backend import tensorflow_backend as K
```

2) 데이터 불러오기 및 데이터 구조 확인

flatten: 1차원 배열로 입력하기 위해 사용하는 인자
normalize: 정규화

```
(X_train, y_train), (X_test, y_test) = mnist.load_mnist(flatten=True, normalize=True)

X_train.shape, X_test.shape
y_train.shape, y_test.shape
```

```
> X_train.shape, X_test.shape
  ((60000, 784), (10000, 784))
> y_train.shape, y_test.shape
  ((60000,), (10000,))
```

학습데이터 60,000개 시험데이터 10,000개
28x28 pixel의 1차원 배열

3) 정답값(Labeling)을 원-핫 인코딩(One-Hot Encoding)방식으로 변환

```
Y_train = pd.get_dummies(y_train).values # one-hot encoding
Y_train
```

```
> y_train
  array([5, 0, 4, ..., 5, 6, 8]
```

```
> Y_train
  array([[0, 0, 0, ..., 0, 0, 0],
         [1, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         ...,
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 1, 0]],
```

4) 인공신경망 모델을 구성

```
model = Sequential()
model.add(Dense(50, activation = 'relu', input_shape=(784,)))
model.add(BatchNormalization())
model.add(Dense(100, activation = 'relu'))
model.add(BatchNormalization())
model.add(Dense(10, activation ='softmax', kernel_initializer = 'zeros'))
model.compile(loss=categorical_crossentropy, optimizer=Adam())
model.summary()
```

```
model.summary()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                39250
_________________________________________________________________
batch_normalization_1 (Batch (None, 50)                200
_________________________________________________________________
dense_2 (Dense)              (None, 100)               5100
_________________________________________________________________
batch_normalization_2 (Batch (None, 100)               400
_________________________________________________________________
dense_3 (Dense)              (None, 10)                1010
=================================================================
Total params: 45,960
Trainable params: 45,660
Non-trainable params: 300
```

은닉층을 2개 두고 각각 50개, 100개의 노드를 구성

출력층은 정답값이 10가지이므로 10개로 구성

원-핫 인코딩(One-Hot Encoding)

데이터를 표현하는 방식 중의 하나로 수많은 0과 하나의 1로 주어진 데이터를 표현하는 방식이다. 가령 0~9까지 숫자를 표현하는 정보라면 다음과 같이 나타낼 수 있다.

1 → [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

신경망에서는 분류 문제에서 교차 엔트로피 오차(Cross Entropy Error)를 이용하여 손실을 구하기 때문에 정답값(Labeling)을 원-핫 인코딩으로 변환하는 작업이 필요하다.

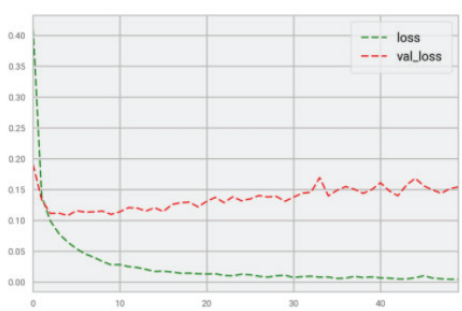
5) 학습데이터를 이용한 인공신경망 모델 학습

```
훈련결과 = model.fit(X_train, Y_train, batch_size=128, epochs=50, validation_split=0.2)
```

- Batch_size : 한 번 학습할 때 사용하는 데이터 세트의 크기를 말한다.
  (학습데이터의 크기가 60,000개인데 이를 한꺼번에 학습을 시킬 경우 비효율적인 자원활용으로 계산속도가 늦어진다.)
- Epochs : 전체 데이터 세트가 총 몇 번 반복 학습하는지를 나타낸다.
- Validation_split : 학습데이터 중 검증용으로 얼마나 할당할 것인지 설정한다.

```
Train on 48000 samples, validate on 12000 samples
Epoch 1/50

48000/48000 [==============================] - 3s 59us/step - loss:
0.4118 - val_loss: 0.1910
Epoch 2/50
 16 32 46 60 75 89102116130142154167180193206217230244257270281295
093253393533673813924004144284444584734800 0/48000 [==============
==============] - 2s 41us/step - loss: 0.1391 - val_loss: 0.1331
                              ...
Epoch 49/50
 16 30 44 60 75 89103119133148163179193207221234248262277291305321
3635236738039440942443945346748000/48000 [==========================
=====] - 2s 39us/step - loss: 0.0044 - val_loss: 0.1513
Epoch 50/50
 16 32 46 61 75 89104120135149165179193207221235250264279293307322
3935436938539941442844445847248000/48000 [==========================
=====] - 2s 39us/step - loss: 0.0041 - val_loss: 0.1541
```

```
손실변화_1.plot(style={'loss': 'g--', 'val_loss':'r--'})
<matplotlib.axes._subplots.AxesSubplot at 0x7f5df8703128>
```



약 3-4번까지 학습데이터의 손실과 검증데이터의 손실이 비슷하게 떨어지다가
그 후에는 검증과 학습데이터의 손실값이 어느 정도 차이를 보인다. 이는 약 3-4번 학습
시켰을 때 학습데이터와 시험데이터 사이에 차이가 가장 적음을 알 수 있다.

6) 모델 평가

```
분류정확도 = lambda y, y_pred: np.mean(y==y_pred)

y_train_pred = model.predict(X_train)
y_test_pred = model.predict(X_test)

y_train_pred = np.argmax(y_train_pred, axis=1)
y_test_pred = np.argmax(y_test_pred, axis=1)

훈련점수 = 분류정확도(y_train, y_train_pred)
시험점수 = 분류정확도(y_test, y_test_pred)

print(f'훈련점수:{훈련점수}, 시험점수 : {시험점수}')
```

```
> print(f'훈련점수:{훈련점수}, 시험점수 : {시험점수}')
  훈련점수:0.99185, 시험점수 : 0.9715
```

학습 데이터 : 60,000개
시험 데이터 : 10,000개

즉 인공신경망 모델이 60,000개의 데이터를 학습했더니 학습데이터 60,000개 중에 약 59,611개를 정답으로 맞추었고, 시험데이터는 10,000개 중에 9,715개를 맞추었다는 의미이다.

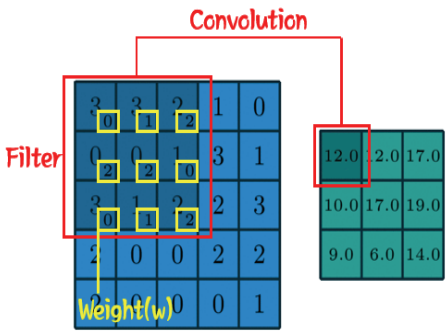
3 합성곱신경망(Convolutional Neural Network)

핵심요약

합성곱신경망(CNN)은 인공신경망 모델의 하나로 패턴을 찾아 이미지를 분석하는데 특화된 알고리즘이다. 주요 구성은 크게 합성곱(Convolution) 연산과 풀링(Pooling) 연산으로 나눌 수 있다.

합성곱(Convolution)

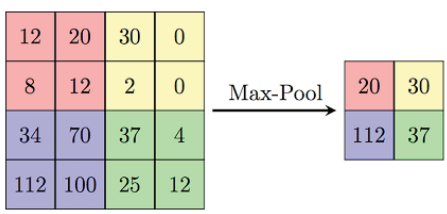
원본 이미지와 영상의 패턴을 추출할 수 있는 필터(Filter)를 이용하여 특징을 추출하는 과정이다. 필터는 원본 이미지를 움직이면서(Stride) 이미지의 특징을 뽑아내는 결과물(Feature Map)을 만든다. 결과물(Feature Map)은 원본 이미지의 인접한 픽셀 간 연관성 있는 패턴 정보를 잃지 않고 반영할 수 있다.



〈합성곱 연산〉

풀링(pooling)

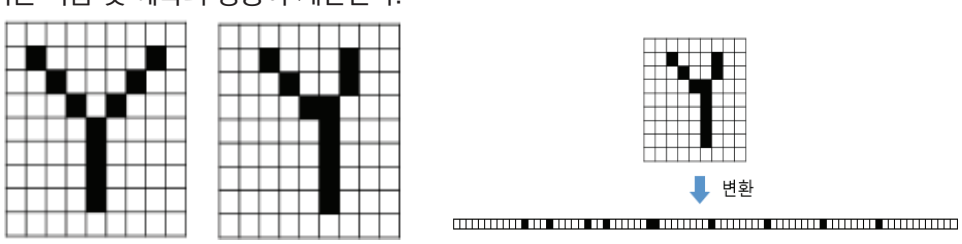
합성곱 연산을 통해 나온 결과물에서 대푯값들만 뽑아내는 과정이다. 이미지 패턴 정보를 단순화, 추상화하는 작업으로 생각할 수 있다. 풀링의 종류에는 최대(Max), 최소(Min), 평균(Average) 등 여러 가지가 있는데, 일반적으로 최대 풀링(Max-Pooling)을 사용한다.



〈Max-pooling 연산〉

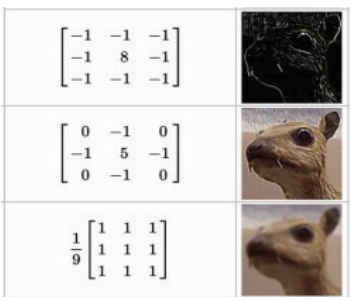
인공신경망과 이미지

일반적인 인공신경망은 데이터를 입력층(Input Layer)에 일차원 행렬 형태로 입력한다. 이 경우, 비교적 단순한 구조의 데이터는 문제가 되지 않으나, 분석할 데이터가 이미지와 같이 공간적인 정보를 포함할 때 발생한다. 즉, 이미지 정보는 같은 의미를 갖고 있는 정보일지라도 일부분이 변경되는 경우 일차원 행렬의 관점에서 보면 데이터의 변화가 크기 때문에 일반적인 인공신경망에서는 학습 및 예측의 성능이 제한된다.



필터(Filter)

원본 입력 데이터에 대해 특징값을 뽑기 위해 만들어진 장치이다. 원본이미지와 필터를 합성곱 연산을 시키면 오른쪽 그림과 같이 다양한 특징, 혹은 관점으로 이미지를 인식할 수 있는 결과물이 나온다. 합성곱 신경망에서는 이 필터에 값들이 가중치로서 학습과정에서 데이터에 맞게 변경된다.

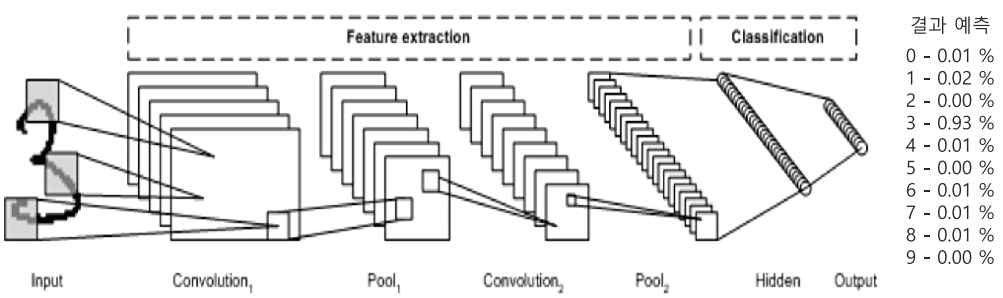


〈필터 행렬과 합성곱 연산된 이미지〉

앞에서 살펴본 합성곱(Convolution), 풀링(Pooling), 필터(Filter)와 같은 구성요소를 이용하여 합성곱신경망(Convolutional Neural Network)을 어떻게 활용할 수 있을지 살펴보자.

이미지 분류(Image Classification)

입력으로 이미지 정보를 받아서 이미지가 어디에 속할지 분류하는 문제이다. 가령 숫자를 인식하는 문제나 주어진 사진이 개인지 고양이인지 분류하는 문제의 유형이 여기에 해당된다.

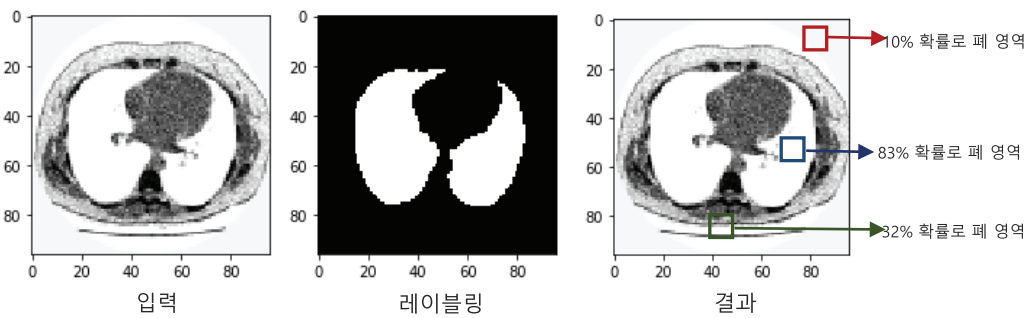


〈이미지 분류 문제 예시〉

위의 그림은 숫자 정보를 받아 0~9까지 숫자를 결과로 예측하는 합성곱신경망의 구조이다. 앞 부분에서 필터(filter)로 합성곱 연산하여 특징맵(Feature Map)을 추출하고 풀링(Pooling)연산을 통해 대푯값들만 추출하게 된다. 이 과정을 여러 번 거쳐 생성된 수많은 특징맵(Feature Map)을 인공신경망의 분류 모델에 입력하여 특징맵(Feature Map)의 중요도를 정하는 가중치 조합을 찾아 숫자를 예측한다. 즉, 합성곱신경망(CNN) 분류 모델은 필터(Filter)행렬과 신경망의 가중치 조합을 학습데이터에 맞게 찾아 결과를 예측한다고 볼 수 있다.

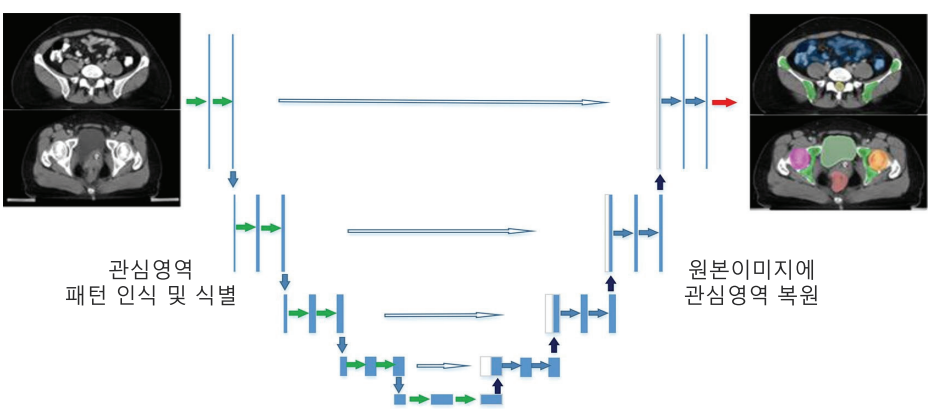
이미지 분할(Image Segmetation)

이미지의 관심 부분을 인식하는 문제이다. 자율주행자동차의 주변 환경정보(차도, 인도, 차선 등)를 인식하는 문제나, 의료분야에서 영상을 판독할 때 병변의 위치를 확인하는 예가 대표적이다. 이미지 분류(Image Classfication)가 이미지 전체에 대한 분류 문제였다면 이미지 분할(Image Segmentation)은 픽셀 정보 하나하나에 대한 분류 문제라 할 수 있다.



〈이미지 분할 예측 결과 예시〉

입력 이미지 정보로부터 관심영역(Region of Interest)을 찾기 위해서는 합성곱 연산을 이용해서 특정 패턴을 인식, 식별할 수 있어야 하고 동시에 식별된 부위의 위치정보를 복원(Reconstruction)할 수 있어야 한다. 일반적인 합성곱 신경망은 대푯값을 추출하는 풀링(Pooling) 연산을 통해 원본보다 크기가 작아지기 때문에 아래 그림의 구조처럼 각 부위를 식별하는 동시에 원본 크기의 이미지로 복원하여 부위별로 분할된 예측 결과를 보여준다.



〈이미지 분할 알고리즘(U-Net) 예시〉

문제정의

부위가 다른 복부(Abdomen) 및 폐(Chest) X-ray 영상을 분류(Classification)

DATASET

- 학습데이터 : 복부(Abdomen) 및 폐(Chest) X-ray 영상 100장
- 시험데이터 : 복부(Abdomen) 및 폐(Chest) X-ray 영상 10장

1) 학습데이터를 이용하여 인공신경망 모델 학습

```
from keras import applications, optimizers
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.models import Model, Sequential
from keras.layers import *
from skimage.transform import rotate
from skimage import exposure
import matplotlib.pyplot as plt
import numpy as np
import os
```

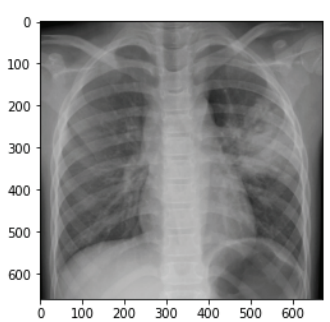
딥러닝 관련 라이브러리

이미지 처리 관련 라이브러리

2) 학습데이터를 이용하여 인공신경망 모델 학습

```
BASE_PATH = './dataset/xray_abd_chest'
TRAIN_DATA_PATH = os.path.join(BASE_PATH, 'TRAIN')
VAL_DATA_PATH = os.path.join(BASE_PATH, 'VAL')
TEST_DATA_PATH = os.path.join(BASE_PATH, 'TEST', 'chest1.png')
```

```
img = plt.imread(TEST_DATA_PATH)
plt.imshow(img, cmap='Greys')
plt.show()
```




```
np.shape(img)
```

```
(661, 673, 3)
```

```
img
```

```
array([[[0.01568628, 0.01568628, 0.01568628],
        [0.01960784, 0.01960784, 0.01960784],
        [0.01568628, 0.01568628, 0.01568628],
        ...,
        [0.16862746, 0.16862746, 0.16862746],
        [0.16862746, 0.16862746, 0.16862746],
        [0.16470589, 0.16470589, 0.16470589]],

       [[0.01960784, 0.01960784, 0.01960784],
        [0.01960784, 0.01960784, 0.01960784],
        [0.01960784, 0.01960784, 0.01960784],
        ...,
        [0.18039216, 0.18039216, 0.18039216],
        [0.18039216, 0.18039216, 0.18039216],
        [0.18039216, 0.18039216, 0.18039216]],
```

이미지는 행렬(가로, 세로, 채널)의 수치정보로 표현되어 있다.

3) 합성곱 신경망을 구성

```
basemodel = applications.inception_v3.InceptionV3(weights='imagenet',
                                                  include_top=False,
                                                  input_shape=(IMG_HEIGHT, IMG_WIDTH, 3))
model_top = Sequential()
model_top.add(GlobalAveragePooling2D())
model_top.add(Dense(256, activation='relu'))
model_top.add(Dropout(0.5))
model_top.add(Dense(1, activation='sigmoid'))


model = Model(inputs=basemodel.input, outputs=model_top(basemodel.output))
model.compile(optimizer=Adam(lr=LEARNING_RATE,
                             epsilon=1e-8,
                             decay=DECAY_RATE),
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.summary()
```

이미 사전 학습된 'inception'이라는 모델을 사용하였다. Inception 모델은 이미지 분류 문제에 사용될 수 있으며 이렇게 사전 학습된 모델을 활용하는 것을 전이학습이라고 한다.

활성화함수는 ReLU를 사용

손실함수로 두 가지 분류를 위한 이진 교차 엔트로피를 사용

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 256, 256, 3)      0
_________________________________________________________________
conv2d (Conv2D)              (None, 127, 127, 32)      864
_________________________________________________________________
batch_normalization (BatchNorma (None, 127, 127, 32)      96
_________________________________________________________________
activation (Activation)      (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 32)      9216
_________________________________________________________________
batch_normalization_1 (BatchNor (None, 125, 125, 32)      96
_________________________________________________________________
activation_1 (Activation)    (None, 125, 125, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 125, 125, 64)      18432
                                                       ...
_________________________________________________________________
batch_normalization_93 (BatchNo (None, 6, 6, 192)     576
_________________________________________________________________
activation_85 (Activation)   (None, 6, 6, 320)      0
_________________________________________________________________
mixed9_1 (Concatenate)       (None, 6, 6, 768)      0
                                                       activation_87[0][0]
                                                       activation_88[0][0]
_________________________________________________________________
concatenate_1 (Concatenate)  (None, 6, 6, 768)      0
                                                       activation_91[0][0]
                                                       activation_92[0][0]
_________________________________________________________________
activation_93 (Activation)   (None, 6, 6, 192)      0
_________________________________________________________________
mixed10 (Concatenate)        (None, 6, 6, 2048)     0
                                                       activation_85[0][0]
                                                       mixed9_1[0][0]
                                                       concatenate_1[0][0]
                                                       activation_93[0][0]
_________________________________________________________________
sequential (Sequential)      (None, 1)              524801
=================================================================
Total params: 22,327,585
Trainable params: 22,293,153
Non-trainable params: 34,432
_________________________________________________________________
```

합성곱 신경망 구성 요약

4) 학습데이터를 이용하여 인공신경망 모델 학습

```
Epoch 1/10
13/13 [==============================] - 52s 3s/step - loss: 0.6424 - accuracy: 0.6357 - val_loss: 0.0956 - val_accuracy: 1.0000
Epoch 2/10
13/13 [==============================] - 42s 3s/step - loss: 0.2862 - accuracy: 0.8597 - val_loss: 0.0141 - val_accuracy: 1.0000
Epoch 3/10
13/13 [==============================] - 42s 3s/step - loss: 0.1961 - accuracy: 0.9262 - val_loss: 0.0063 - val_accuracy: 1.0000
Epoch 4/10
13/13 [==============================] - 42s 3s/step - loss: 0.0678 - accuracy: 0.9769 - val_loss: 0.0076 - val_accuracy: 1.0000
Epoch 5/10
13/13 [==============================] - 42s 3s/step - loss: 0.0396 - accuracy: 1.0000 - val_loss: 0.0050 - val_accuracy: 1.0000
Epoch 6/10
13/13 [==============================] - 42s 3s/step - loss: 0.0904 - accuracy: 0.9785 - val_loss: 0.0021 - val_accuracy: 1.0000
Epoch 7/10
13/13 [==============================] - 42s 3s/step - loss: 0.1304 - accuracy: 0.9382 - val_loss: 4.7813e-04 - val_accuracy: 1.0000
Epoch 8/10
13/13 [==============================] - 42s 3s/step - loss: 0.1719 - accuracy: 0.9377 - val_loss: 2.3517e-05 - val_accuracy: 1.0000
Epoch 9/10
13/13 [==============================] - 42s 3s/step - loss: 0.0468 - accuracy: 1.0000 - val_loss: 1.2927e-06 - val_accuracy: 1.0000
Epoch 10/10
13/13 [==============================] - 42s 3s/step - loss: 0.0131 - accuracy: 1.0000 - val_loss: 6.6884e-07 - val_accuracy: 1.0000
```

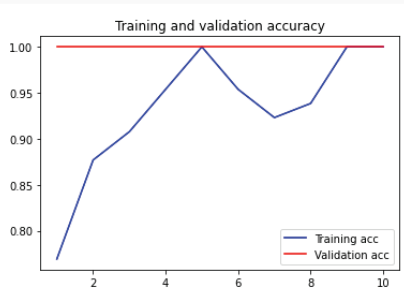
→ 학습데이터 100개에 대해서 총 10번 반복 학습하였다.

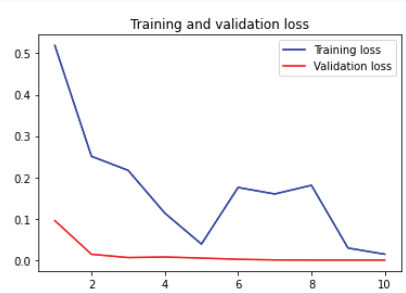
5) 모델 평가

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
plt.plot(epochs, acc, 'b', color='blue', label='Training acc')
plt.plot(epochs, val_acc, 'b', color='red', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.show()
```

```
plt.plot(epochs, loss, 'b', color='blue', label='Training loss')
plt.plot(epochs, val_loss, 'b', color='red', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```



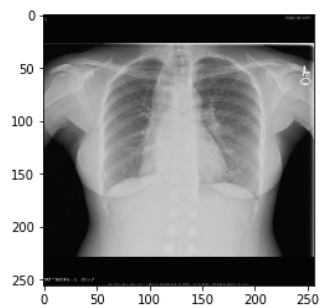


모델이 학습이 잘 되었는지 검증을 하기 위해 훈련 점수 및 손실 함수의 값을 측정하고 그래프로 확인하였다. 시험점수와 학습점수가 학습 횟수가 반복될수록 1에 수렴하고 있고 손실 함수의 값은 반대로 0에 수렴하는 것을 확인할 수 있다.

6) 임의의 영상이 복부(abdomen)인지 폐(chest)인지 예측 확인

```
img1_path = os.path.join(BASE_PATH, 'TEST', 'chest2.png')
img2_path = os.path.join(BASE_PATH, 'TEST', 'abd2.png')
```

```
img1 = image.load_img(img1_path, target_size=(IMG_HEIGHT, IMG_WIDTH))
img2 = image.load_img(img2_path, target_size=(IMG_HEIGHT, IMG_WIDTH))
```




```
img1 = image.img_to_array(img1)
img1 /= 255.
img1 = img1[np.newaxis, :, :, :]
```

```
score1 = model.predict(img1)
print('Predicted:', 'Chest X-ray' if score1 < 0.5 else 'Abd X-ray', ', Score:', score1[0,0])
```

```
Predicted: Chest X-ray , Score: 5.930332e-05
```

→ 예측 값이 약 0.00053 이므로 폐 영상으로 인식

```
plt.imshow(img2)
plt.show()
```


```
img2 = image.img_to_array(img2)
img2 /= 255.
img2 = img2[np.newaxis, :, :, :]
```

```
score2 = model.predict(img2)
print('Predicted: ', 'Chest X-ray' if score2 < 0.5 else 'Abd X-ray', ', Score:', score2[0,0])
```

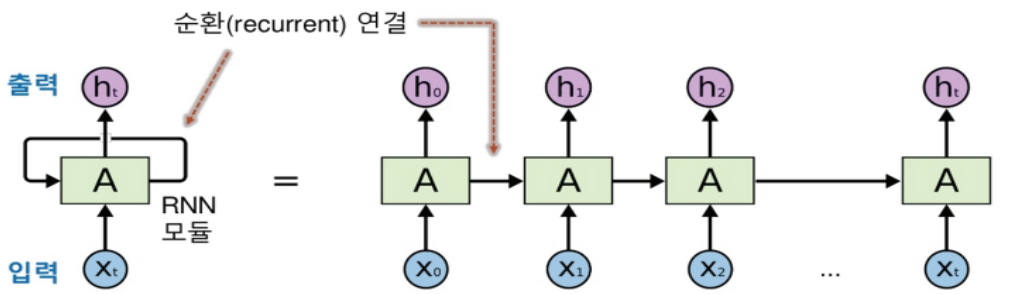
```
Predicted:  Abd X-ray , Score: 0.98762
```

→ 예측 값이 약 0.988 이므로 복부 영상으로 인식

4 순환신경망(Recurrent Neural Network)

핵심요약

순환신경망(Recurrent Neural Network)은 시간 순서가 있는 데이터를 잘 예측하도록 설계된 인공신경망 모델들 중 하나이다. 과거의 신호를 기억할 수 있는 장치(Hidden State)를 두어 입력신호를 순환, 반복하는 순환적 구조를 갖고 있어서 시간적 순서 특성을 추출하는데 용이하다.



<출저: https://heung-bae-lee.github.io/2020/01/12/deep_learning_08/>

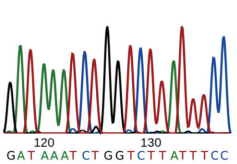
〈순환신경망(RNN)의 구조〉

특징

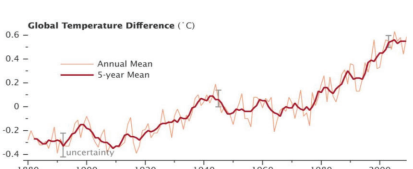
일반적인 인공신경망이 은닉층을 통해 한꺼번에 가중치 연산이 이루어 지는 것과 비교해서 순환신경망은 이전 데이터의 가중치 연산이 다음 데이터에 일정 부분 영향을 줄 수 있도록 구성되어 있다. 이러한 구조는 시간 순서의 정보가 중요한 데이터에서 이전 데이터를 보고 다음 데이터를 예측하는데 도움을 준다.

순차 데이터(Sequential Data)

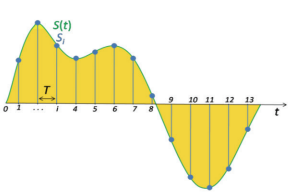
순서나 시간이 전체 데이터에서 의미가 있으며 그 순서가 달라질 경우 의미가 크게 변하는 데이터를 말한다. 여기서 단순히 시간적 순서가 있는 데이터를 순차 데이터(Sequential Data)라 부르고 그 중에서도 시간적 간격이 일정한 것을 시계열 데이터(Time Series data)라고 부른다.



DNA 염기 서열
(Sequential Data)



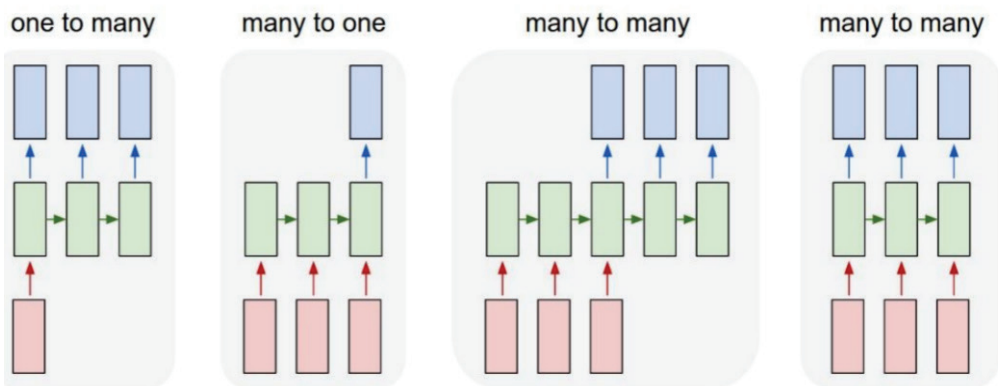
세계 기온 변화
(Temporal Sequence)



샘플링된 소리 신호
(Time Series)

활용사례

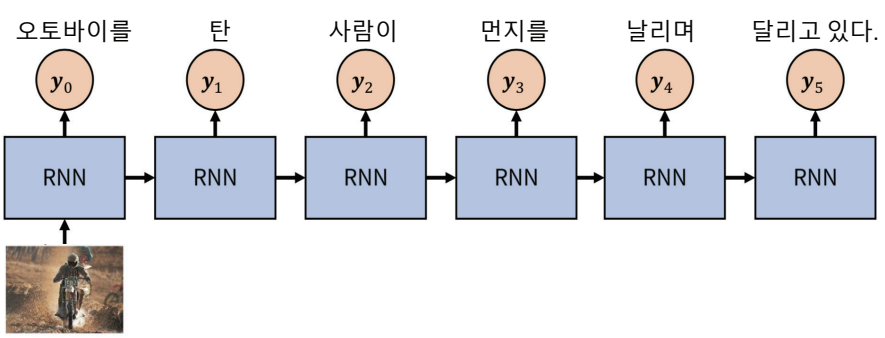
순환신경망은 순차 데이터(Sequential Data)를 이용하여 다양한 분야에 사용할 수 있다. 순차 데이터의 입력값 혹은 출력값의 크기에 따라 크게 세 가지 유형(One to Many, Many to One, Many to Many)으로 나누어 구분할 수 있다. 다음 세 가지 유형에 대해 사례별로 살펴보자.



〈순환신경망(RNN)의 유형〉

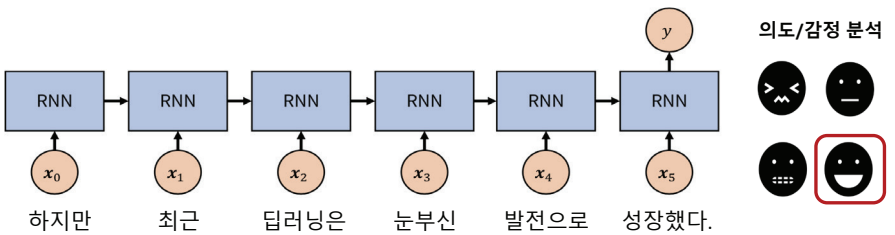
One to Many

고정된 이미지 정보를 해석하여 설명하는 문장을 만드는 이미지 캡션 – Image Caption(Show and Tell, Google)이 대표적인 사례이다. 엄밀하게 이 사례는 두 가지 신경망이 결합된 형태이다. 입력데이터로 들어온 이미지 정보가 합성곱신경망(CNN)을 거쳐 특징(Feature)정보를 가지고 그림의 의미를 단어로 나타내면 순환신경망(RNN)이 이것을 입력으로 받아 이미지 정보 의미에 맞게 형태소를 순서대로 배열하여 순서 있는 문장으로 출력하는 것이다.



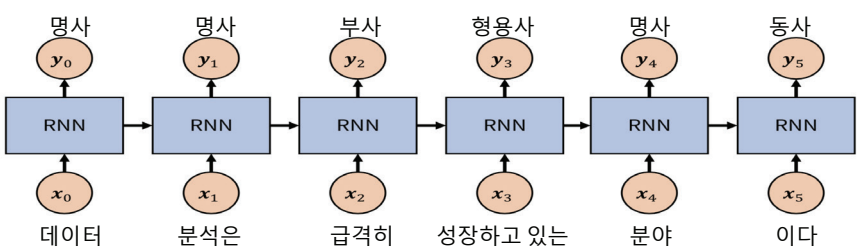
Many to One

입력값으로 순차 데이터(Sequential Data)가 들어가고 출력값으로 고정된 값을 출력하는 경우를 말한다. 문장을 입력받아 해당 문장이 어떤 감정 혹은 의도를 갖고 있는지 학습하여, 순서 있는 문자 정보에 대해 의도나 감정 분류(classification)하는 문제이다.

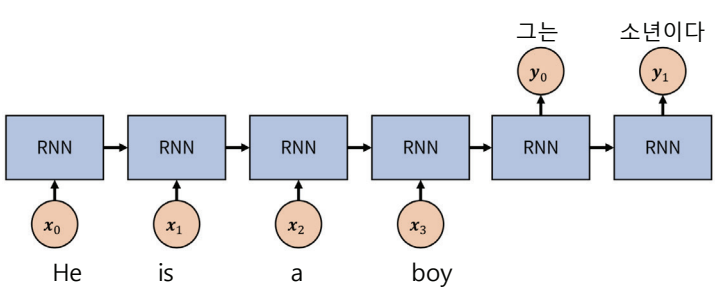


Many to Many

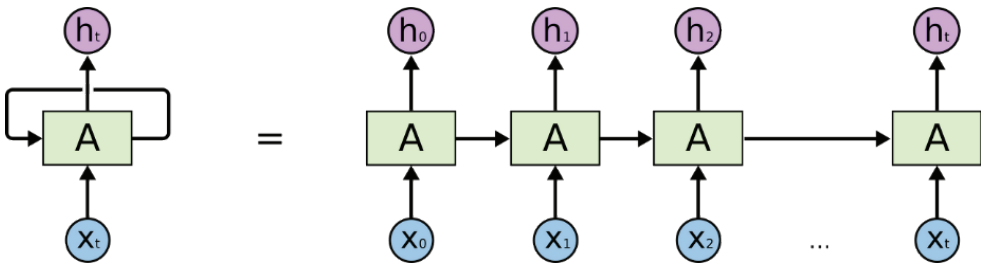
입력 및 출력값 모두 순차 데이터로 사용한다. 우리는 Many to Many를 다시 두 가지로 나눌 수 있는데, 하나는 데이터가 입력될 때마다 매 순간(Time-Step)마다 결괏값이 생성되는 경우이며 문장을 입력할 때마다 결괏값이 바뀔 수 있는 형태소 분석 등이 여기에 해당이 된다.



그리고 나머지 하나는 입력값으로 모든 순차 데이터가 정해졌을 때 결괏값이 생성되는 번역기와 같은 기능을 사례로 들 수 있다.

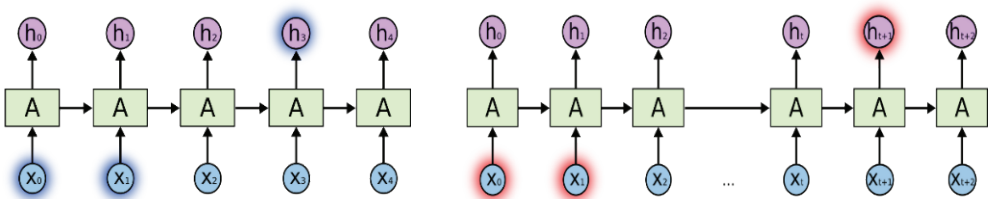


순환신경망(RNN)의 한계와 LSTM(Long Short Term Memory)의 등장



〈순환신경망의 기본적인 구조〉

일반적인 순환신경망(RNN)은 위 그림에서 보는 것처럼 입력 X_0에 대한 결과 H_0 를 구한 후, 이 정보를 다음 연산에도 흘려 그 이전 정보를 반복 사용하는 것을 알 수 있다. 이처럼 과거신호를 기억할 수 있는 장치(hidden state)는 순차적인 데이터를 예측할 때 순서 및 시간 정보를 모델에 반영할 수 있는 획기적인 방법이었고 다양한 사례에 활용되면서 합성곱신경망(CNN)과 함께 큰 축을 차지하게 되었다.

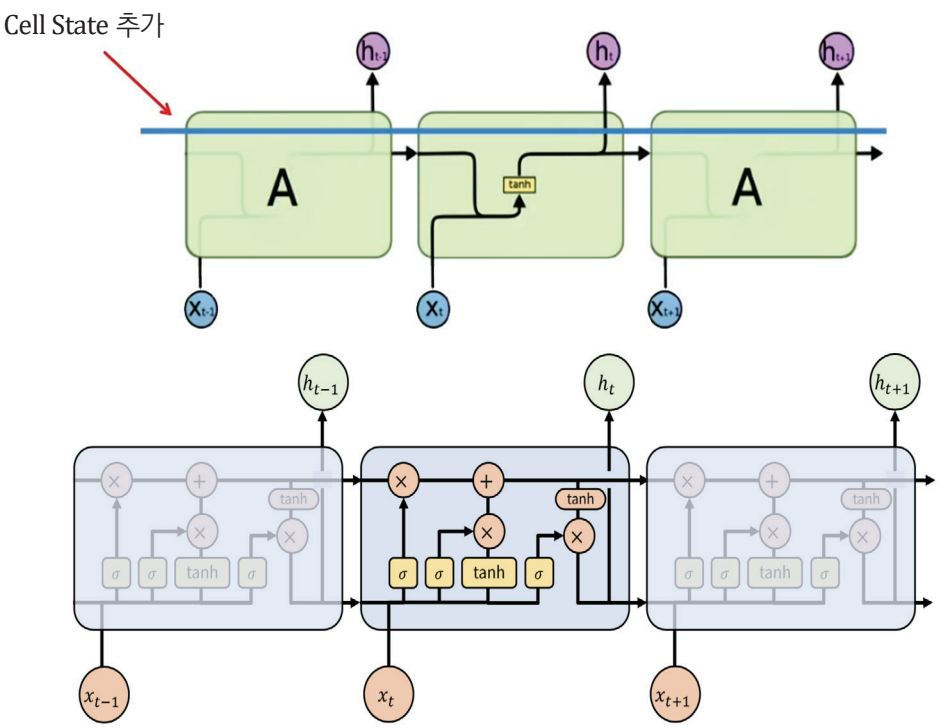


〈순환신경망의 긴 거리 의존성 문제〉

하지만 일반적인 구조의 순환신경망(RNN)은 입력 신호와 출력 신호 사이의 거리가 가까운 경우에 정보를 전달하는 것은 별문제가 없지만, 거리가 멀어지면 멀어질수록 정보를 제대로 전달하지 못하는 문제가 발생한다. 순차 데이터 간의 가중치가 서로 동일하기 때문에 거리가 멀수록 기울기 소실문제(Gradient Vanishing Problem)가 더 일어나는 것이다. 이를 순차 데이터 학습에서 긴 기간 의존성 문제(Long term Dependency)라고 하며 이를 해결하기 위해 LSTM(Long Short Term Memory)이 나오게 되었다.

LSTM(Long Short Term Memory)

기본적인 순환신경망(RNN)구조에 Cell State라는 장치가 더해진 구조를 하고 있다. Cell State를 통해 기억이라는 기능을 좀 더 정교하게 구현할 수 있게 되었는데, 항상 일정한 크기로 이전 정보를 흘려주었던 일반적인 순환신경망(RNN)과 달리 이전 순번의 데이터들 중에서 기억해야 할 것은 더 오래 기억하고 별로 의미 없는 것들은 빨리 잊어버리는 역할을 하는 것이다.



Cell State를 새로 추가 하면서 기능별로 LSTM을 정리하면 다음과 같다.

1. 이전 Cell State에서 불필요한 정보를 제거
2. 이전 Cell State와 현재 입력값의 중요도를 고려해 Cell State를 생성
3. 이전 및 현재 Cell State를 이용하여 Hidden State를 만든다.
4. 현재 Cell State와 Hidden State를 다음 순서의 신경망에 전달한다.

문제정의

접수일자별 청구 데이터를 통해 앞으로 들어올 청구금액을 예측

DATASET DW

데이터베이스에서 본원 심사 청구서를 접수일자(1996.02.~2021.05.)별로 요양급여총액, 청구금액 총액, 본인부담금 총액, 진료비 총액 등 7개의 특성값으로 총 7,518건 추출하였다. 이중에서 보험자 구분은 '건강보험'이고, 청구구분은 '일반청구'를 대상으로 하였다.

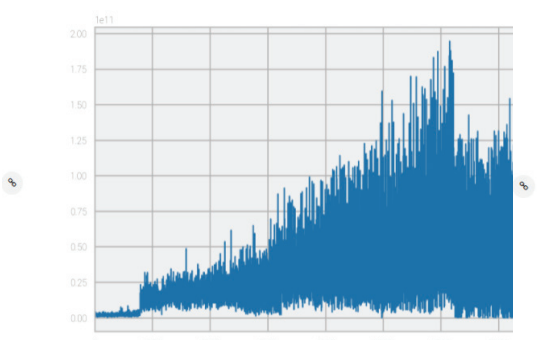
1. 데이터 확인하기

```
> DMD = pd.read_csv("./RNN3.csv", encoding = 'CP949')
> DMD.tail()
```

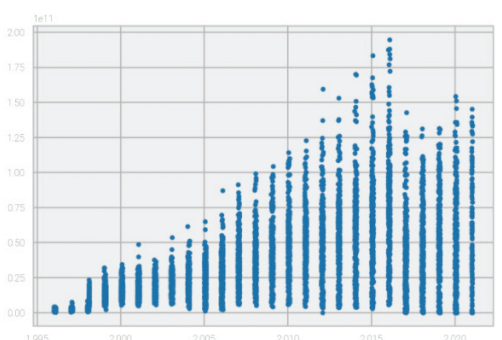
	RCV_DD	RPE_TAMT_AMT	DMD_TOT_AMT	SLF_BRDN_TOT_AMT	DMD_SPAMT_TOT_AMT	HNDP_FND_TOT_AMT	DAMT_TOT_AMT	DMD_TOT_CNT
7529	20210501	2.877265e+07	2.645385e+07	2166130	0	152670	2.877265e+07	326
7530	20210503	2.370060e+10	1.861886e+10	5073499150	69933490	8212750	2.371479e+10	105657
7531	20210504	1.330090e+11	1.178240e+11	15121669239	134518240	29836530	1.330230e+11	38547
7532	20210506	1.338060e+11	1.178680e+11	15882509240	104933984	35351240	1.339210e+11	49179
7533	20210507	8.942386e+10	7.669402e+10	12612386430	159706353	28322610	8.935688e+10	144995

2. 일자 별 총 청구금액 추이 그래프

```
> google.RPE_TAMT_AMT.plot(grid=True)
  <matplotlib.axes._subplots.AxesSubplot at 0x7fea0c33fb38>
```




```
> plt.scatter(DMD['RCV_DD'], DMD['RPE_TAMT_AMT'], marker='.')
  <matplotlib.collections.PathCollection at 0x7fe97428ccf8>
```



전체적으로 늘었다 줄었다를 반복하면서 1996년부터 서서히 증가하다가 2015, 2016년에 최고점을 찍고 최근 3년간은 조금 줄어든 것을 확인할 수 있다.

3. 학습데이터 정규화

```
from sklearn.preprocessing import MinMaxScaler

DMD.sort_index(ascending=False).reset_index(drop=True)

scaler = MinMaxScaler()
scale_cols = ['DMD_TOT_CNT', 'SLF_BRDN_TOT_AMT', 'RPE_TAMT_AMT', 'DMD_SPAMT_TOT_AMT', 'HNDP_FND_TOT_AMT', 'DMD_TOT_AMT', 'DAMT_TOT_AMT' ]
df_scaled = scaler.fit_transform(DMD[scale_cols])
df_scaled = pd.DataFrame(df_scaled)
df_scaled.columns = scale_cols

df_scaled
```

각 변수인 청구 총액, 진료비 총액, 본인 부담금 총액 등은 변수별로 다른 범위를 갖고있는 변수이다. 따라서 학습 시 모든 값을 0에서 1사이의 값으로 변경(rescale)해주었다.

	DMD_TOT_CNT	RPE_TAMT_AMT	DMD_SPAMT_TOT_AMT	HNDP_FND_TOT_AMT	DMD_TOT_AMT	DAMT_TOT_AMT
0	0.010372	0.005216	0.000000	0.000000	0.003884	0.000000
1	0.011688	0.006839	0.000000	0.000000	0.004782	0.000000
2	0.003933	0.005897	0.000000	0.000000	0.005460	0.000000
3	0.008233	0.007813	0.000000	0.000000	0.005160	0.000000
7430	0.170597	0.354979	0.240824	0.124419	0.339706	0.354694
7431	0.231344	0.488013	0.193961	0.182653	0.455753	0.487536
7432	0.003240	0.085680	0.022195	0.056325	0.091224	0.085594
7433	0.210503	0.319303	0.261049	0.101466	0.289922	0.319047

4. 데이터 전처리

```
TEST_SIZE = 75
WINDOW_SIZE = 30

train = df_scaled[:-TEST_SIZE]
test = df_scaled[-TEST_SIZE:]


def make_dataset(data, label, window_size):
    feature_list = []
    label_list = []
    for i in range(len(data) - window_size):
        feature_list.append(np.array(data.iloc[i:i+window_size]))
        label_list.append(np.array(label.iloc[i+window_size]))
    return np.array(feature_list), np.array(label_list)

from sklearn.model_selection import train_test_split

feature_cols =  ['DMD_TOT_CNT', 'RPE_TAMT_AMT', 'DMD_SPAMT_TOT_AMT', 'HNDP_FND_TOT_AMT', 'DMD_TOT_AMT', 'DAMT_TOT_AMT' ]
label_cols = ['DMD_TOT_AMT']
train_feature = train[feature_cols]
train_label = train[label_cols]

train_feature, train_label = make_dataset(train_feature, train_label, WINDOW_SIZE)
x_train, x_valid, y_train, y_valid = train_test_split(train_feature, train_label, test_size=0.2)
```

TEST_SIZE와 WINDOW_SIZE 값을 각각 70일과 30일로 설정하였다. 여기서 TEST_SIZE는 70일치를 예측하겠다는 의미이며, WINDOW_SIZE는 과거 30일치를 기준으로 다음 번에 들어올 청구 총 금액을 예측하겠다는 의미이다. 학습 및 검증 데이터 분리는 8:2로 구성하였다.

5. 모델 생성 및 학습

```
> from keras.models import Sequential
> from keras.layers import Dense
> from keras.callbacks import EarlyStopping, ModelCheckpoint
> from keras.layers import LSTM
> model = Sequential()
> model.add(LSTM(45,
                 input_shape=(train_feature.shape[1], train_feature.shape[2]),
                 activation='relu',
                 return_sequences=False)
          )
> model.add(Dense(1))
> model.summary()
```

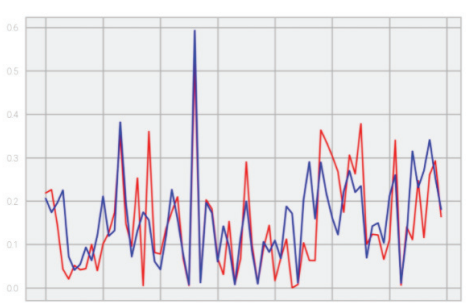
```
> model.compile(loss='mean_squared_error', optimizer='adam')
> early_stop = EarlyStopping(monitor='val_loss', patience=5)
> model_path = './'
> filename = os.path.join(model_path, 'tmp_checkpoint.h5')
> checkpoint = ModelCheckpoint(filename, monitor='val_loss', verbose=1, save_best_only=True, mode='auto')
> history = model.fit(x_train, y_train,
                      epochs=200,
                      batch_size=16,
                      validation_data=(x_valid, y_valid),
                      callbacks=[early_stop, checkpoint])
```

메모리 셀의 개수를 16개, 입력값에 30일치, 하루에 6개의 특성값을 입력값으로 받아 하나의 결과(청구 총 금액)를 예측하는 Many-to-One LSTM을 구성하였다. 오차함수로 회귀 문제에 사용되는 오차평균제곱 함수를 사용하였고, 최적화 함수로는 'adam'을 사용하여 200번 반복을 하면서 검증 오차가 향상될 때까지 반복 학습을 진행하였다.

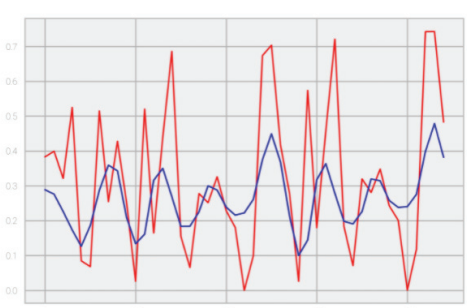
6. 결과

```
Epoch 00028: val_loss did not improve from 0.00868
Epoch 29/200
  64 22 27 33 38 44 49 54 60 67 73 78 84 89 94 99104110115120124131136140147152156161166171177182187192196201206211216220225230235240244251256260265270
275280284289294299304308313318323328333337342347352356363368372377382387392396403409416422427432436441446451456460465470475480484489494499504508513518
235285325375425475525565615665715765805855904/5943 [==============================>.] - ETA: 0s - loss: 0.0077

5943/5943 [==============================] - 7s 1ms/step - loss: 0.0077 - val_loss: 0.0088

Epoch 00029: val_loss did not improve from 0.00868
```

28번째 반복 학습 만에 오차함수의 값이 0.00868에서 학습이 더 진행되지 않은 채 종료되었다. 학습 데이터를 모델을 넣어 확인한 결과와 테스트 데이터를 통해 확인한 결과는 다음과 같다.



〈학습데이터〉



〈시험데이터〉

학습에 사용한 일자별 청구 데이터가 시간 순서를 기반으로 하지만 데이터가 앞뒤 시간 순서 간 관련성이 크지 않기 때문에 구체적인 금액보다 주기나 패턴 위주로 예측하는 것을 확인할 수 있었다.

V. 심평원 사례

1. 의료영상심사판독시스템

2. 분석심사를 위한 특성기관 예측

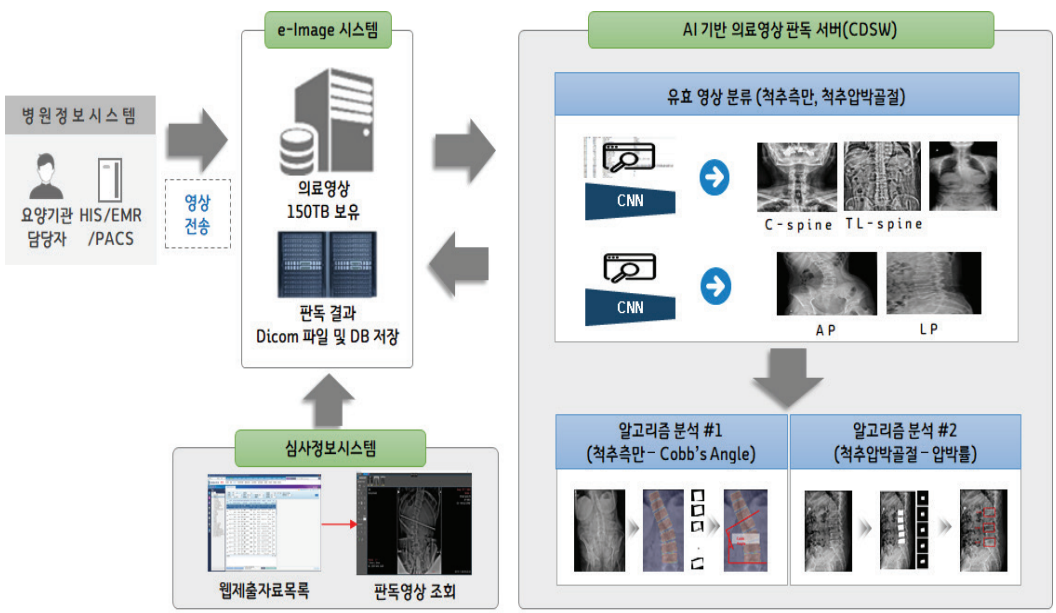
3. 급여정보분석시스템

V. 심평원 사례

1 의료영상심사판독시스템

주요 내용

척추질환을 심사하기 위해 의료기관에서 보낸 심사자료(의료영상 파일)와 합성곱 신경망 모델을 이용하여 급여기준(척추측만의 만곡도와 압박골절의 압박률)을 자동 측정하여 사용자에게 제공한다.



〈의료영상심사판독시스템 구성 흐름도〉

요양기관이 송신한 의료영상은 e-Image 시스템을 통해 심사연계된다. 심사판독 시스템은 심사연계 과정 중에 청구명세서 정보(상병, 수가 코드), 영상정보(장비종류, 검사상세, 부위 등), 촬영각도 분류 모형을 거쳐 인공지능 판독 대상을 선별하고 척추이미지를 인식하는 판독과정을 거친다.

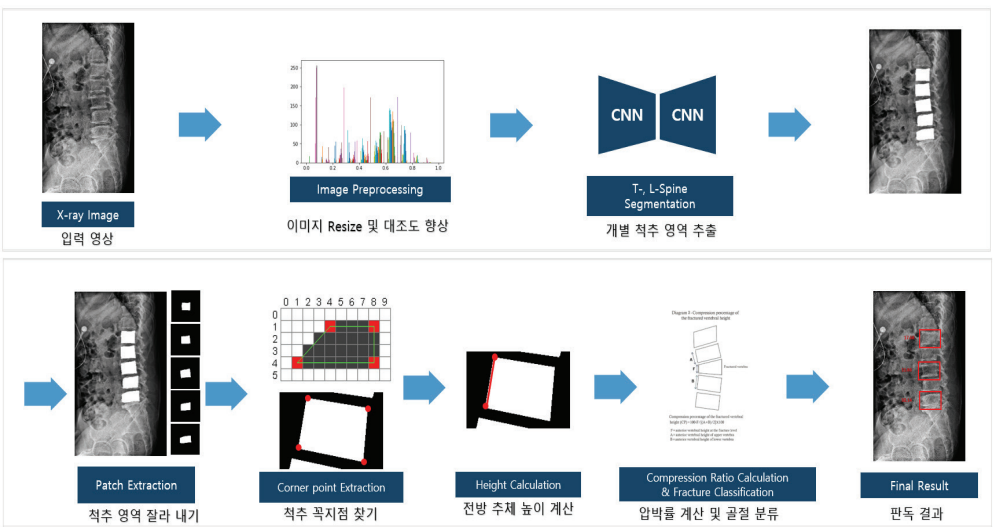
심사자 및 전문위원은 심사시스템에서 제공되는 의료영상 뷰어를 통해 원본 영상과 판독영상을 참조하여 심사에 활용할 수 있다. 의료영상심사판독시스템은 심사판독의 보조 수단으로써 업무 효율성과 판독 객관성을 기대할 수 있다.

분석예시

압박골절 방사선 영상(X-ray)으로 척추 뼈를 인식하여 나온 결과 이미지를 가지고 압박골절의 압박률을 측정하는 예시이다.

DATASET

- 입력 : 측면 척추 의료영상 파일(dcm)
- 결과 : 판독 결과 의료영상 파일(dcm)



〈척추 압박골절의 학습 및 판독 과정〉

1. 데이터 확인하기

```
def comp_1st_prep(image):
    resized_image, crop_coords = resizing_image(image, 512)
    normalized_image = cv2.normalize(resized_image, 0, 255, norm_type=cv2.NORM_MINMAX)
    normalized_image = clahe(normalized_image)
    normalized_image = normalized_image * (1./255.)
    return normalized_image, crop_coords
```

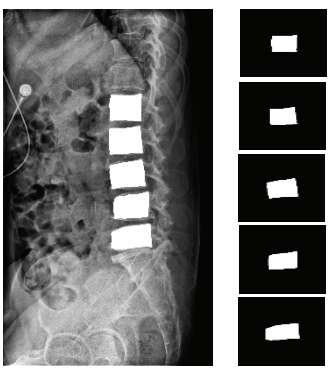
해상도 조정

정규화된 데이터를 CLAHE 처리함

다양한 요양기관 및 장비 업체에서 생성된 영상을 일정하게 정리하기 위해 영상의 해상도를 (512, 512) resize하고, 각 pixel 당 8bit형태의 영상으로 정규화(0-255)를 하였다. 또한 대조도를 높여 보다 선명한 이미지 인식을 위해 CLAHE(Contrast Limited Adaptive Histogram Equalization) 처리를 하였다.

2. 모델 생성(개별 척추 영역 추출)

```
def ai_analyze_compression(image, pixel_spacing, file_name, model_1):
    prep_img_1, crop_coords = comp_1st_prep(image)
    cf_1_pred = model_1.predict(prep_img_1.reshape((1, 512, 512, 1))).reshape((512, 512, 1))
    highest_comp_rate, result_image, exception = comp_1st_post(image, cf_1_pred, crop_coords, image.shape, pixel_spacing,
                                                               '', file_name)
    return highest_comp_rate, result_image, exception
```



〈척추가 식별된 결과 예시〉

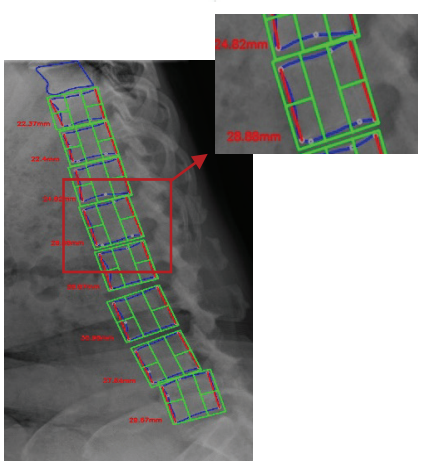
이미 학습이 된 모델에 전처리된 이미지를 입력하여 입력 영상과 동일한 크기(512, 512)의 분할 처리된 이미지(Segmentation Prediction Mask) 결과를 얻었다.

Segmentation Prediction Mask는 우리가 X-ray영상에서 판독에 필요한 척추 이미지만을 식별하기 위해 0(척추영역 아님), 1(척추 영역)의 값으로 관심영역만을 추출한 결과 이미지를 말한다.

3. 관심영역(Resign Of Interset) 표시

```
# pred_mask_binary = mask_opening(pred_mask_binary, KERNEL_SIZE, pixel_spacing)     # For coarse-ing the dilated spine area
cropped_spine_properties, exception_bool = get_spine_properties(original_image, pred_mask_binary, pixel_spacing, debug_image_dir, file_name)

result_img = cv2.cvtColor(original_image.copy(), cv2.COLOR_GRAY2BGR)
contours, _ = cv2.findContours(pred_mask_binary.astype(np.uint8), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

분할(Segmentation)모델을 통해 생성된 결과 이미지를 후처리하여 척추를 꼭짓점과 다면체를 기준으로 점과 선을 연결하여 인식한다. 이미지를 인식한 결과를 기준으로 전방 척추체의 높이를 각각 계산하였다. 여기서 파란색 도형은 분할모델 윤곽선이고 회색 점은 그 결과물에서 얻은 다각형 꼭짓점이다.



〈전방 척추체 높이 계산〉

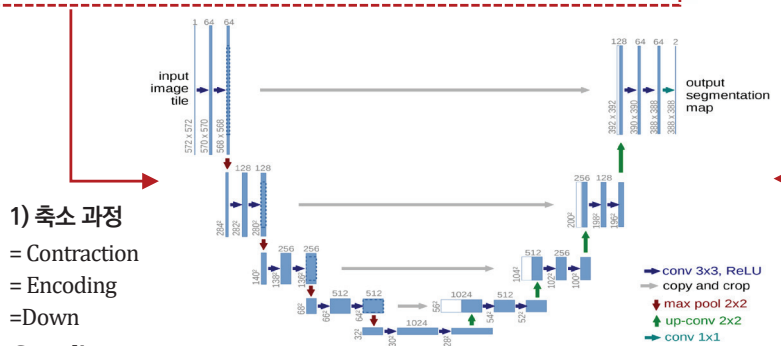
◆ U-Net 모델 생성 및 구조

의료영상에서 영역분할(Segmentation) 문제에 가장 많이 사용하는 U-Net을 활용하여 모델을 생성하였다. U-Net은 각 이미지 안에 픽셀정보를 분류(Classification)하기 위해 고안된 모델이다.

```
import torch.nn as nn
import torch
import torch.nn.functional as F

class Modified2DUNet(nn.Module):
    def __init__(self, in_channels, n_classes, base_n_filter = 8):
        super(Modified2DUNet, self).__init__()
        self.in_channels = in_channels
        self.n_classes = n_classes
        self.base_n_filter = base_n_filter

        self.lrelu = nn.LeakyReLU()
        self.dropout3d = nn.Dropout3d(p=0.6)
```

U-Net의 전반부인 Context를 찾는 과정을 구현한 것으로 해상도를 떨어뜨려가며 다양한 채널로 특징맵을 추출해가는 과정이다.

```
# Level 1 context pathway
self.conv3d_c1_1 = nn.Conv2d(self.in_channels, self.base_n_filter, kernel_size=3, stride=1, padding=1, bias=False)
self.conv3d_c1_2 = nn.Conv2d(self.base_n_filter, self.base_n_filter, kernel_size=3, stride=1, padding=1, bias=False)
self.lrelu_conv_c1 = self.lrelu_conv(self.base_n_filter, self.base_n_filter)
self.gnorm3d_c1 = nn.GroupNorm(self.base_n_filter//2, self.base_n_filter)

# Level 2 context pathway
self.conv3d_c2 = nn.Conv2d(self.base_n_filter, self.base_n_filter*2, kernel_size=3, stride=2, padding=1, bias=False)
self.norm_lrelu_conv_c2 = self.norm_lrelu_conv(self.base_n_filter*2, self.base_n_filter*2)
self.norm_lrelu_conv_c2 = self.norm_lrelu_conv(self.base_n_filter*2, self.base_n_filter*2)
self.norm_lrelu_conv_c2 = self.norm_lrelu_conv(self.base_n_filter*2, self.base_n_filter*2)
self.gnorm3d_c2 = nn.GroupNorm(self.base_n_filter, self.base_n_filter*2)
```



1) 축소 과정
= Contraction
= Encoding
=Down
Sampling

2) 확장 과정
= Expansion
= Decoding
= Up
Sampling

```
# Level 3 localization pathway
self.conv_norm_lrelu_l3 = self.conv_norm_lrelu(self.base_n_filter*4, self.base_n_filter*4)
self.conv3d_l3 = nn.Conv2d(self.base_n_filter*4, self.base_n_filter*2, kernel_size=1, stride=1, padding=0, bias=False)
self.norm_lrelu_upscale_conv_norm_lrelu_l3_1 = self.norm_lrelu_upscale_conv_norm_lrelu_1(self.base_n_filter*2)
self.norm_lrelu_upscale_conv_norm_lrelu_l3_2 = self.norm_lrelu_upscale_conv_norm_lrelu_2(self.base_n_filter*2, self.base_n_filter)

# Level 4 localization pathway
self.conv_norm_lrelu_l4 = self.conv_norm_lrelu(self.base_n_filter*2, self.base_n_filter*2)
self.conv3d_l4 = nn.Conv2d(self.base_n_filter*2, self.n_classes, kernel_size=1, stride=1, padding=0, bias=False)

self.ds2_1x1_conv3d = nn.Conv2d(self.base_n_filter*8, self.n_classes, kernel_size=1, stride=1, padding=0, bias=False)
self.ds3_1x1_conv3d = nn.Conv2d(self.base_n_filter*4, self.n_classes, kernel_size=1, stride=1, padding=0, bias=False)
```

U-Net의 후반부인 Localization 과정으로 앝은 층 이전의 특징맵과 Upsampling한 특징맵을 서로 결합하여 지역(Localization)과 의미(Context) 사이의 트레이드 오프(Trade off)를 줄일 수 있다.

```
Layer (type)                 Output Shape              Param #   Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 256, 256, 1) 0
__________________________________________________________________________________________________
conv2d_1 (Conv2D)            (None, 256, 256, 64) 640       input_1[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 256, 256, 64) 256       conv2d_1[0][0]
__________________________________________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 256, 256, 64) 0         batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)            (None, 256, 256, 64) 36928     leaky_re_lu[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 256, 256, 64) 256       conv2d_2[0][0]
__________________________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 256, 256, 64) 0         batch_normalization_1[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 128, 128, 64) 0         leaky_re_lu_1[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)            (None, 128, 128, 128 73856     max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 128, 128, 128 512       conv2d_3[0][0]
__________________________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 128, 128, 128 0         batch_normalization_2[0][0]
__________________________________________________________________________________________________
conv2d_4 (Conv2D)            (None, 128, 128, 128 147584    leaky_re_lu_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 128, 128, 128 512       conv2d_4[0][0]
__________________________________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 128, 128, 128 0         batch_normalization_3[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 64, 64, 128)  0         leaky_re_lu_3[0][0]
__________________________________________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 256)  295168    max_pooling2d_1[0][0]
__________________________________________________________________________________________________
                                                                  ⋮
__________________________________________________________________________________________________
conv2d_18 (Conv2D)           (None, 128, 128, 128 295040    concatenate_2[0][0]
__________________________________________________________________________________________________
batch_normalization_14 (BatchNo (None, 128, 128, 128 512       conv2d_18[0][0]
__________________________________________________________________________________________________
leaky_re_lu_14 (LeakyReLU)   (None, 128, 128, 128 0         batch_normalization_14[0][0]
__________________________________________________________________________________________________
conv2d_19 (Conv2D)           (None, 128, 128, 128 147584    leaky_re_lu_14[0][0]
__________________________________________________________________________________________________
batch_normalization_15 (BatchNo (None, 128, 128, 128 512       conv2d_19[0][0]
__________________________________________________________________________________________________
leaky_re_lu_15 (LeakyReLU)   (None, 128, 128, 128 0         batch_normalization_15[0][0]
__________________________________________________________________________________________________
up_sampling2d_3 (UpSampling2D)  (None, 256, 256, 128 0         leaky_re_lu_15[0][0]
__________________________________________________________________________________________________
conv2d_20 (Conv2D)           (None, 256, 256, 64) 32832     up_sampling2d_3[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 256, 256, 128 0         leaky_re_lu_1[0][0]
                                                               conv2d_20[0][0]
__________________________________________________________________________________________________
conv2d_21 (Conv2D)           (None, 256, 256, 64) 73792     concatenate_3[0][0]
__________________________________________________________________________________________________
batch_normalization_16 (BatchNo (None, 256, 256, 64) 256       conv2d_21[0][0]
__________________________________________________________________________________________________
leaky_re_lu_16 (LeakyReLU)   (None, 256, 256, 64) 0         batch_normalization_16[0][0]
__________________________________________________________________________________________________
conv2d_22 (Conv2D)           (None, 256, 256, 64) 36928     leaky_re_lu_16[0][0]
__________________________________________________________________________________________________
batch_normalization_17 (BatchNo (None, 256, 256, 64) 256       conv2d_22[0][0]
__________________________________________________________________________________________________
leaky_re_lu_17 (LeakyReLU)   (None, 256, 256, 64) 0         batch_normalization_17[0][0]
__________________________________________________________________________________________________
conv2d_23 (Conv2D)           (None, 256, 256, 2)  1154      leaky_re_lu_17[0][0]
__________________________________________________________________________________________________
batch_normalization_18 (BatchNo (None, 256, 256, 2)  8         conv2d_23[0][0]
__________________________________________________________________________________________________
leaky_re_lu_18 (LeakyReLU)   (None, 256, 256, 2)  0         batch_normalization_18[0][0]
__________________________________________________________________________________________________
conv2d_50 (Conv2D)           (None, 256, 256, 1)  3         leaky_re_lu_18[0][0]
==================================================================================================
Total params: 31,055,245
Trainable params: 31,043,465
Non-trainable params: 11,780
__________________________________________________________________________________________________
```

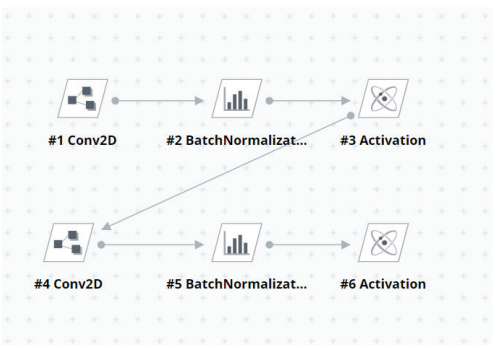
〈Model Summary〉

모델의 요약에서 총 31,055,245개의 파라미터들이 사용되었음을 확인할 수 있다. 그리고 활성화 함수로 Leaky ReLU를 사용하고 Max Pooling을 사용하였다.

ConvLayer → Nomalization → Activation을 반복하는 ConvBlock을 중복하여 사용하는 기존 U-Net의 구조에서 간단한 요소(component)들을 추가하였다.

예시 모델에서는 ConvBlock 사이에 가우시안 드롭아웃(Gaussian Noise Dropout)을 사용하였다.

일반적으로 드롭아웃(Dropout)은 인공신경망에서 몇몇 노드(Node)에 무작위로 노이즈(Noise)를 주어 새로운 데이터에 좀 더 유연하게 대응하기 위해 사용하는 장치로서 과적합(Overffiting) 현상을 막을 수 있다.



〈ConvBlock의 구조〉

4. 결과 확인

심사시스템에서 심사자료(웹제출자료목록 조회) 화면을 통해 영상 확인

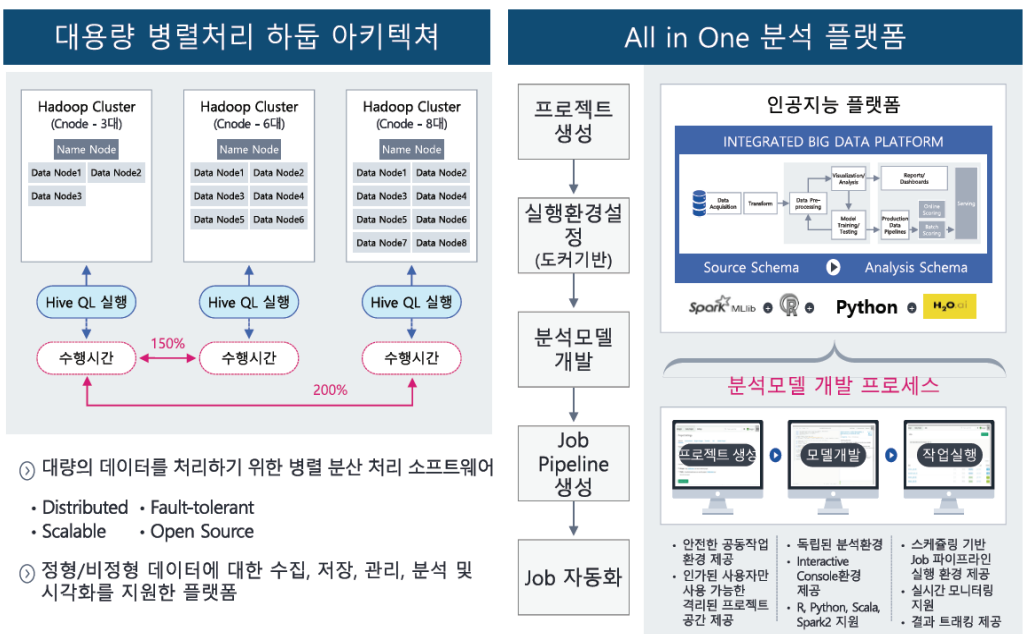
〈원본 영상〉 〈판독 영상〉

심사자는 영상 뷰어에서 제공하는 원본영상과 판독영상을 함께 열람할 수 있고 판독된 영상을 참고하여 심사에 활용할 수 있다.

❷ 분석심사를 위한 특성기관 예측

주요 내용

분석심사를 위해 임상결과와 치료비용에 대한 월별/분기별 심사결정자료를 대상으로 지표를 산출하고 요양기관의 진료추세를 표기하여 임상 및 비용에 대해 진료행태의 경향성을 파악하여 집중 심사(분석)가 필요한 AZ기관(임상↓, 비용↑)을 회귀분석을 이용해 예측하는 분석모델이다.



〈분석심사 인공지능 플랫폼 구성 및 데이터 분석 흐름도〉

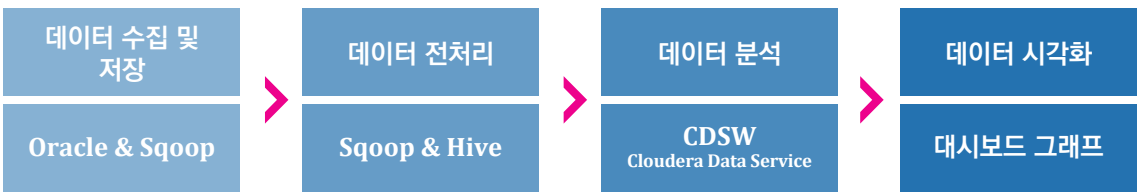
생성된 모델로 지표별 가중치를 산출하여 집중대상 기관 AI지수와 순위를 생성한다. 심사자는 분석심사포털 집중분석기관 선정화면에서 분석심사 지표가 동일점수인 경우 AI분석 지수와 순위를 이용하여 심사에 활용할 수 있다.

분석심사와 AZ기관 선별

기존의 청구명세서의 건별 심사에서 진료정보에 대해 주제별로 분석지표, 청구현황 등을 다차원 분석하여 전문심사위원회가 분석결과와 의학적 근거, 진료 특성 등 종합적으로 검토 후 중재방법을 정하는 심사방식이다. 이 과정에서 진료정보 데이터를 분석하여 집중 심사 대상 요양기관을 선별한다.

Hadoop 기반의 R&Python을 이용한 모델 개발

집중분석 대상 기관을 선별하기 위한 로지스틱 회귀분석을 위한 절차는 다음과 같다. 분석심사 인공지능 플랫폼은 Hadoop 기반에서 동작하기 때문에 수집, 전처리 분석 등에 Sqoop, Hive, CDSW와 같은 분석 도구들이 사용된다.



〈데이터 분석 환경〉

주제별 지표 → 집중 분석기관 선별 → 심사 활용

AZ기관(임상↓, 비용↑)을 선별하기 위해 판단에 유의미한 영향을 주는 변수를 선정하는 회귀분석을 통해 지표를 추출한다. 추출된 지표를 이용하여 로지스틱 회귀모델로 집중분석 대상 기관(4분위 대상)을 선별하여 심사에 활용한다.

〈특성기관 선별을 위한 데이터 분석 절차〉

분석예시

슬관절 질환의 청구명세서를 이용하여 집중분석 대상 기관인 AZ기관(임상↓, 비용↑)을 선별하는 모델을 만들어 심사에 활용하는 사례이다.

DATASET

2019년 3, 4분기 슬관절 데이터

- 입력 : 이미 개발된 지표, 신규입력변수(청구현황, 환자특성, 요양기관 특성 등)
- 결과 : 지표산출 결과 4사분위 데이터 – 집중분석 대상 기관 선별용

기 개발 지표
65세 미만 환자 수술 비율
복잡기준 수술 비율
수술 전후 비경구 항생제 평균 투여일수
환자보정 총진료비
환자보정 총진료비 열외군 비율
환자보정 입원일수
환자보정 입원일수 열외군 비율
진료비 변동 추이
장기입원 환자 비율
⋮

청구현황
대상 명세서 건수_3개월
내원환자수_3개월
처방전 건수_3개월
명세서당 평균 상병수
대상상병 금액 비율
대상상병 환자 비율
전분기 대비 대상 명세서 비율
전분기 대비 환자수 비율
전분기 대비 처방전 건수 비율
⋮

환자 특성
성별 환자수 비율_여성
성별 환자수 비율_남성
성별 평균 내원일수_여성
연령대별 환자수 비율_30대미만
연령대별 환자수 비율_30대
연령대별 환자수 비율_40대
연령대별 청구금액 비율_30대미만
연령대별 청구금액 비율_30대
연령대별 청구금액 비율_40대
⋮

요양기관 특성
개설 후 경과개월수
종별 평균 대비 총청구금액 비율
종별 평균 대비 원내진료비 비율
종별 평균 대비 원외처방약제비 비율
지역 평균 대비 총청구금액 비율
지역 평균 대비 원내진료비 비율
지역 평균 대비 원외처방약제비 비율
진료과목 평균 대비 총청구금액 비율
진료과목 평균 대비 원내진료비 비율
⋮

1. 데이터 전처리

- 결측치 제거
- 데이터 정렬

```
# ==========================================================================
# 결측치를 0으로 변경(02. 전처리/12 결측치 처리.R)
# --------------------------------------------------------------------------
df_org[is.na(df_org)] <- 0

# ==========================================================================
# 산출년도, 산출분기, 요양기호를 오름차순으로 정렬(02. 전처리/09 데이터 정렬.R)
# --------------------------------------------------------------------------
df_org <- df_org[order(df_org$cmpt_yr, df_org$cmpt_qt, df_org$ykiho, decreasing = FALSE),]
```

2. 데이터 탐색

- 기초 통계량 확인

  합계, 평균, 표준편차,
  분산, 변동계수,
  최빈값, 최댓값 등

```
## 기초통계 수치 구하기
for (i in 1:ncol(df_org)) {
  # 컬럼명을 하나씩 추출하여 col에 저장
  col <- colnames(df_org)[i]

  # 추출된 컬럼명(col)과 일치하는 열로 데이터프레임 만들기
  df_col <- select(df_org, col)

  # 만들어진 데이터프레임을 벡터로 변환
  vt <- as.vector(as.matrix(df_col))

  # 해당 데이터가 numeric일 경우에만 기초통계 확인
  if(is.numeric(vt)){
    group_result <- data.frame(컬럼명 = col,
                               합계 = sum(vt),
                               평균 = round(mean(vt, na.rm = TRUE),2),
                               표준편차 = round(sd(vt, na.rm = TRUE),2),
                               표준분산 = round(var(vt, na.rm = TRUE),2),
                               변동계수 = round(sd(vt, na.rm = TRUE)/mean(vt, na.rm = TRUE),2),
                               최빈값 = getmode(vt),
                               최소값 = min(vt, na.rm = TRUE),
                               최대값 = max(vt, na.rm = TRUE),
                               Q1 = quantile(vt, 1/4, na.rm = TRUE),
                               중위수 = median(vt, na.rm = TRUE),
                               Q3 = quantile(vt, 3/4, na.rm = TRUE),
                               IQR = IQR(vt, na.rm = TRUE) )

    # 결과 데이터프레임에 해당 컬럼의 기초통계 결과 붙이기
    group_result_apnd <- rbind(group_result_apnd,group_result)

  }
}
rownames(group_result_apnd) <- seq(1:nrow(group_result_apnd))

df_group_result_apnd <- select(group_result_apnd, c(컬럼명, 합계, 평균
```

```
> head(df_group_result_apnd, n=10)
        컬럼명      합계  평균 표준편차       Q1      중위수       Q3
1   admin_0001   80670.78  1.26     2.90 0.1611047 0.4945598 1.226994
2     cli_0001 2201140.66 34.47    26.85 13.8461538 33.3333333 50.000000
3     cli_0002 1599841.40 25.06    25.80  4.3478261 18.1818182 35.616438
4     cli_0003 1683428.69 26.36    30.32  0.0000000 14.2857143 47.619048
5     cli_0004 3121335.74 48.88    36.82 10.0000000 50.0000000 83.928571
6     cli_0005  644477.07 10.09    22.28  0.0000000  0.0000000  4.878049
7     cli_0006  933971.95 14.63    26.41  0.0000000  0.0000000 17.241379
8     cli_0007 1320354.92 20.68    28.75  0.0000000  7.4074074 30.769231
9     cli_0008 1419213.34 22.23    29.04  0.0000000  8.3333333 36.363636
10  cost_0001   62680.72  0.98     0.35  0.7944082  0.9639308  1.141372
```

3. 유의 변수 도출

```r
# 변수 선택을 위한 p-value 기준값
num_crtr_pv <- 0.05

# x변수만 추출
df_tmp_num_var <- df_train[!names(df_train) %in% li_factor]

for(i in 1:ncol(df_tmp_num_var)){
  col_i <- colnames(df_tmp_num_var)[i]

  if(col_i == chr_y_var){
    next()
  }

  # 회귀식 만들기
  form_i <- formula(paste0(chr_y_var, " ~ ", col_i))

  # 회귀 모델 만들기
  mdl_i <- glm(form_i, data = df_tmp_num_var, family = "binomial")

  # estimate, p-value 가져오기
  df_coef <- as.data.frame(summary(mdl_i)$coefficients)
  df_coef <- df_coef[,c("Estimate", "Pr(>|z|)")]
  df_coef <- cbind("var_nm" =  rownames(df_coef), df_coef)
  colnames(df_coef) <- c("var_nm", "estimate", "pvalue")
  df_coef <- subset(df_coef, df_coef$var_nm == col_i)

  if(i == 1){
    df_all_coef <- df_coef
  }else{
    df_all_coef <- rbind(df_all_coef, df_coef)
  }

}

rownames(df_all_coef) <- seq(1, nrow(df_all_coef), by = 1)


# p-value가 0.05이하인 변수만 추출
df_selc_var1 <- subset(df_all_coef, df_all_coef$pvalue < num_crtr_pv)
```

```r
# 변수선택을 위한 vif 기준값
num_max_degree <- 10

chr_x_var <- chr_selc_var
num_max_vif <- NA

while(is.na(num_max_vif) | num_max_vif > num_max_degree){

  # 모델식 생성
  chr_vif_form <- paste0(chr_y_var, " ~ ", paste(chr_x_var, collapse = " + "))

  # 회귀분석
  mdl_tmp_vif <- glm(formula = chr_vif_form, data = df_man_train, family = "binomial")

  # 다중공선성 확인
  df_tmp_vif <- as.data.frame(vif(mdl_tmp_vif))
  df_tmp_vif <- cbind(var_nm = row.names(df_tmp_vif), df_tmp_vif)
  colnames(df_tmp_vif) <- c("var_nm", "vif_value")

  # vif 내림차순으로 정렬하여 가장 높은 vif값 확인
  df_tmp_vif <- df_tmp_vif[order(df_tmp_vif$vif_value,  decreasing = TRUE),]
  num_max_vif <- df_tmp_vif$vif_value[1]

  # 가장 높은 vif값이 위에서 지정한 vif 기준값보다 작으면 중지
  if(num_max_vif <= num_max_degree){
    break;
  }else{
    print(paste0("vif 제거 변수: ", df_tmp_vif$var_nm[1], ",  vif: ", df_tmp_vif$vif_value[1]))
  }

  # 가장 높은 vif를 제외한 변수들을 x변수로 다시 저장
  chr_x_var <- as.character(df_tmp_vif[2:nrow(df_tmp_vif),1])
}
rownames(df_tmp_vif) <- seq(1:nrow(df_tmp_vif))
```

최종 선택 변수

```
 [1] "cli_0001"                    "tgt_sick_spec_cnt_rt"
 [3] "bf_qt_cpe_cr_tot_diag_amt_rt" "bf_qt_cpe_vst_ddcnt_rt"
 [5] "tgt_sick_amt_rt"             "hinsu_avg_vst_ddcnt"
 [7] "age65_lt_fml_ptnt_cnt_rt"    "ptnt_pp_vst_ddcnt"
 [9] "cr_tot_diag_amt"             "a65g_fml_ptnt_cnt_rt"
[11] "a65g_fml_avg_vst_ddcnt"      "maid_ptnt_cnt_rt"
[13] "cost_0001"                   "dmd_sel_incl_rpe_tamt_amt"
[15] "maid_avg_vst_ddcnt"          "cost_0005"
[17] "cost_0002"                   "cost_0004"
[19] "cli_0003"                    "cost_0003"
[21] "a65g_male_avg_vst_ddcnt"     "age65_lt_fml_avg_vst_ddcnt"
[23] "cli_0004"                    "cli_0002"
[25] "estb_af_elaps_mms_cnt"       "czu_tot_amt"
[27] "bf_qt_avg_prsc_ddcnt_rt"
```

▪ 다변량 회귀분석

–변수 선택을 위한 P-value 설정

–회귀분석 및 유의수준 확인

–P-value 〈 0.05인 변수 추출

▪ 다중공선성 처리

–변수 선택을 위한 VIF 기준 설정

–회귀분석 및 분산 VIF 확인

–VIF 값이 가장 큰 변수 제거

–최대 VIF 값이 〈 10이 될 때까지 반복

4. 로지스틱 회귀분석

```
# 로지스틱 회귀모형 생성
form_reg <- formula(paste0(chr_y_var, " ~ ", paste(chr_selc_fin_var, collapse = " + ")))
mdl_reg <- glm(formula = form_reg, data = df_fin_train, family = "binomial")

# 모형 확인
summary(mdl_reg)

# 테스트 데이터 예측
pred_test <- predict(mdl_reg, newdata = df_test, type = "response")
df_test_pred <- cbind(df_test[chr_y_var], predict = round(pred_test, 5))
```

테스트 데이터 예측 결과

mutat_adm_yn	predict
0	0.01973
0	0.01942
0	0.01973
0	0.01942
0	0.00720
0	0.00720
0	0.00888
0	0.00888
0	0.01164
0	0.01164
0	0.01678
0	0.01678
0	0.05722
0	0.04655
0	0.04655
0	0.05722
0	0.00629

모델 결과

```
summary(mdl_reg)

Call:
glm(formula = form_reg, family = "binomial", data = df_fin_train)

Deviance Residuals:
     Min       1Q   Median       3Q      Max
-3.2999  -0.4053  -0.1914  -0.0571   2.7260

Coefficients:
                                 Estimate       Std. Error z value      Pr(>|z|)
(Intercept)                   -17.7935391061351  1.0572672713116 -16.830  0.76786
cli_0001                        0.0378619235571  0.0083837238123   4.515  0.67304
tgt_sick_spec_cnt_rt         -229.6414807911034 117.0928654038123  -1.961  0.00778 **
bf_qt_cpe_cr_tot_diag_amt_rt   -0.8851935425924  0.4695875329936  -1.885  0.09514 .
bf_qt_cpe_vst_ddcnt_rt          0.5712013431155  0.3945135971250   1.448  0.06754 .
tgt_sick_amt_rt                 6.6392825870671  2.1052224725725   3.154  0.72803
hinsu_avg_vst_ddcnt             0.0503732112735  0.0218332712417   2.307  0.21764
age65_lt_fml_ptnt_cnt_rt       -2.2361489539472  0.8162656360462  -2.739  0.04013 *
ptnt_pp_vst_ddcnt              -0.0324755189574  0.0207432857330  -1.566  0.00364 **
cr_tot_diag_amt                -0.0000000472937  0.0000000634454  -0.745  0.00850 **
a65g_fml_ptnt_cnt_rt            0.7700785549055  0.5794197164758   1.329  0.00587 **
a65g_fml_avg_vst_ddcnt          0.0062174063012  0.0136891729034   0.454 0.00000000149 ***
maid_ptnt_cnt_rt               -0.2660885845220  0.6777476218547  -0.393  0.94192
cost_0001                       8.2434578381010  0.7030788542670  11.725  0.97780
dmd_sel_incl_rpe_tamt_amt      -0.0000000003419  0.0000000003929  -0.870  0.53429
maid_avg_vst_ddcnt              0.0128949630039  0.0143531220015   0.898  0.31783
cost_0005                       0.9366962313260  0.3688032976640   2.540  0.73340
```

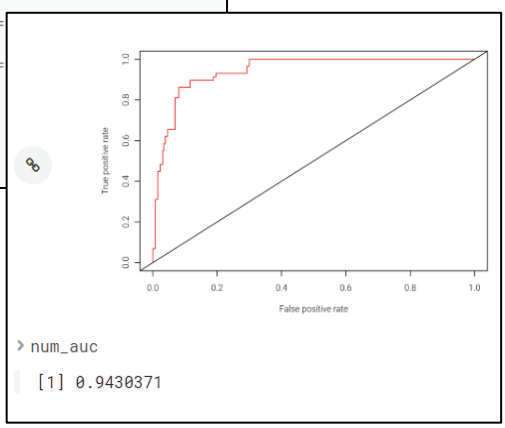
회귀계수

P Value

로지스틱 회귀모델을 생성하여 모형을 확인하고 시험 데이터를 통해 예측 결괏값을 바탕으로 변이기관을 확인할 수 있다.

5. 모형검증

```
# ======================================================================
# ROC 커브(05. 검증/03 ROC_AUC.R)
# ======================================================================

# 예측값과 실제값 비교
pr_test <- prediction(pred_test, df_test[,chr_y_var])

# TPR(True Accept Rate;민감도), FPR(False Positive Rate;특이도) 계산
prf_test <- performance(pr_test, measure = "tpr", x.measure = "fpr")

# ROC커브 표시
plot(prf_test, col = "red")
abline(0,1)
```




```
# ================================================
# AUC 통계량(05. 검증/03 ROC_AUC.R)
# ================================================

# AUC(Area Under the Curve) 계산
auc <- performance(pr_test, measure = "auc")
num_auc <- auc@y.values[[1]]
num_auc
```

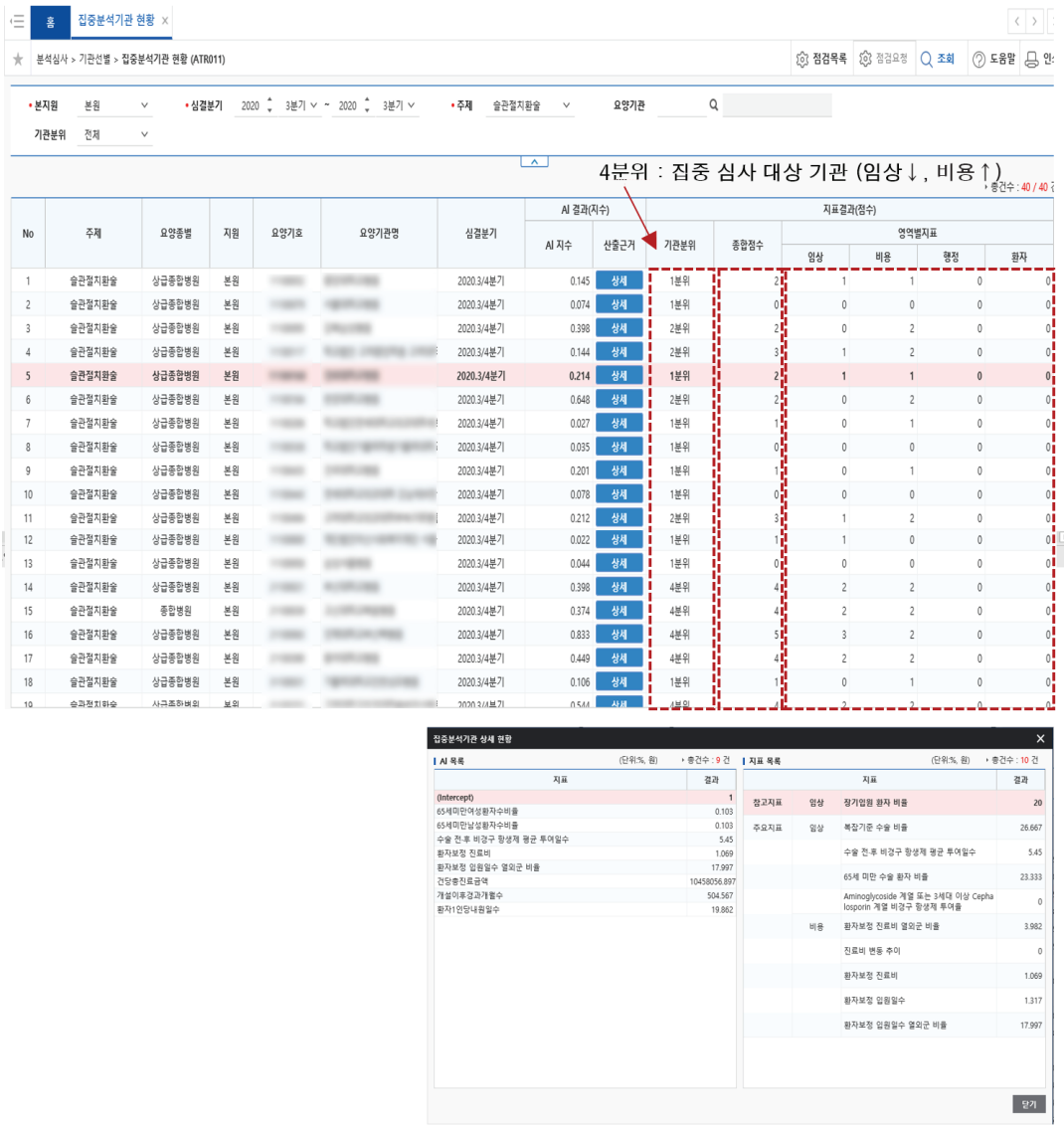
```
> num_auc
[1] 0.9430371
```

▪ 생성된 모형의 성능 검증

ROC는 분류 모델을 임계값에 따른 분류모델의 성능을 보여주는 그래프로 (어떤 연속형 변수가 이분형 결과변수를 예측하기에 적절한가를 살펴보는 검증방법) X축은 1-특이도, Y축은 민감도를 나타낸다. 성능이 높을수록 좌상향으로 치우쳐 나타나게 된다.

AUC(Area Under Curve)는 ROC 곡선 아래 영역의 넓이로 높을수록 모델의 성능이 우수하다고 볼 수 있다. 결괏값은 0.9430371로 높은 결괏값을 확인할 수 있다.

6. 집중 분석 기관 현황



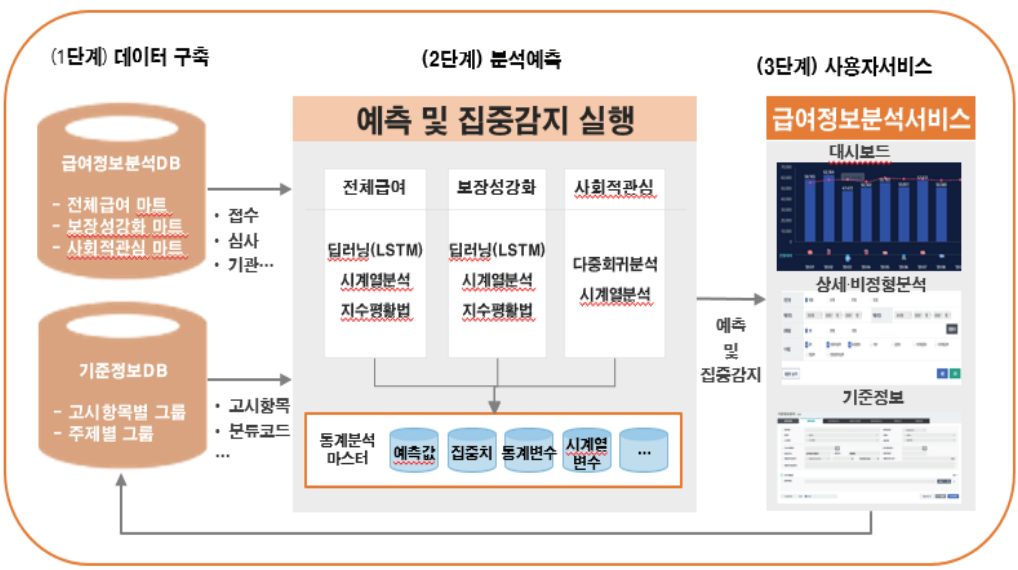
심사자는 화면을 통해 결과뿐만 아니라 다양한 지표들을 참고할 수 있다.

실제 변이기관을 최종 판단하는 'AI 지표'와 4분위값을 확인할 수 있고 '산출근거 상세' 기능을 통해 4가지 영역(임상, 비용, 행정, 환자)에 상세 결과와 사용된 변수들을 확인할 수 있다.

3 급여정보분석시스템

주요 내용

월별 접수·심사결정 자료에서 총 진료비와 보험자부담금을 대상으로 시간 기준(접수·진료·심사 년월), 관점 기준(종별, 분류유형별 등)으로 데이터 세트를 구축하고 시계열 분석 기법을 이용하여 사용자에게 진료비 예측 정보를 제공하는 시스템이다.



〈급여정보분석 시스템 흐름도〉

- 1단계 : DW 청구자료, 사용자등록자료 등을 기반으로 분석 관점별 기준정보 및 급여정보분석용 마트를 구축하여 예측 모델에 입력값으로 활용
- 2단계 : 보장성 강화·전체급여·사회적 관심 업무 관점별로 예측 및 집중감지
- 3단계 : 급여정보포털에서 대시보드·상세분석 등 분석 및 예측 결과 조회

분석심사와 AZ기관 선별

기존의 청구명세서의 건별 심사에서 진료정보에 대해 주제별로 분석지표, 청구현황 등을 다차원 분석하여 전문심사위원회가 분석결과와 의학적 근거, 진료 특성 등 종합적으로 검토 후 중재 방법을 정하는 심사방식이다. 이 과정에서 진료정보 데이터를 분석하여 집중 심사 대상 요양기관을 선별한다.

분석예시

기준 연월의 입력 데이터를 시계열 분석 모델을 이용하여 전체급여 항목별 급여비용을 예측하는 실습 예제이다. 예측의 분류유형 중 행위에 대해서만 요양종별, 지역별로 세분화하여 예측하며 유형별로 정해진 기간 동안 총 진료비와 최소 횟수 기준을 적용해 일정 수준 이하인 경우 제외 처리하였다.

분류유형	총 진료비 기준	횟수(3개월)	비고
행위	5,000,000원(1개월)	30회	최근 1개월 총 사용량 실시횟수가 20회 이하이며, 조회기간 내 200회를 초과한 월이 없는 경우
약제	38,000,000원(4개월)	100회	
치료재료	1,700,000원(2개월)	50회	

〈분류 유형별 예측 제외 대상〉

DATASET

2021년 1월 ~ 2021년 7월 전체 급여명세서 정보

- 입력 : 전체급여 명세서 마스터정보, 항목별 입력데이터 정보
- 결과 : 전체급여 총액

1. 입력데이터 불러오기

```
   -- 전체급여_항목별
WITH DS_ITEM AS (
   SELECT --+ PARALLEL(16) FULL(X)
          DIV_TY_CD,MNTR_MDIV_CD,MNTR_DIV_TY_CD, MNTR_RECU_CL_CD, SIDO_CD,
          CASE WHEN SN <= 100 THEN '1'  -- 분류유형별 상위 100개 항목
               WHEN MNTR_DIV_TY_CD = '1' AND SN <= 7000 THEN '2'  -- 행위 7000개
               WHEN MNTR_DIV_TY_CD = '2' AND SN <= 2000 THEN '2'  -- 약제 2000개
               WHEN MNTR_DIV_TY_CD = '3' AND SN <= 700  THEN '2'  -- 치료재료대 700개
               ELSE '0' END AS ANAL_TP
     FROM (
        SELECT --+ PARALLEL(16) FULL(A) FULL(B)
               A.DIV_TY_CD, A.MNTR_MDIV_CD,
               B.MNTR_DIV_TY_CD, MNTR_RECU_CL_CD, SIDO_CD,
               ADDC_ADT_AMT,
               ROW_NUMBER() OVER (PARTITION BY B.MNTR_DIV_TY_CD ORDER BY ADDC_ADT_AMT DESC) AS SN
          FROM (
             SELECT --+ PARALLEL(16) FULL(A)
                    DIV_TY_CD, MNTR_MDIV_CD,
                    CASE WHEN DIV_TY_CD IN ('1', '2', 'A', 'B', 'X') THEN MNTR_RECU_CL_CD ELSE '$' END MNTR_RECU_CL_C
                    CASE WHEN DIV_TY_CD IN ('1', '2', 'A', 'B', 'X') THEN SIDO_CD ELSE '$' END SIDO_CD,
                    SUM(ADDC_ADT_AMT) AS ADDC_ADT_AMT
               FROM TMJWR107 A
               JOIN TMJWA527 B ON A.RECU_CL_CD = B.RECU_CL_CD
10             WHERE RCV_YM  = TO_CHAR(ADD_MONTHS(TO_DATE('%s','YYYYMM'),-1),'YYYYMM')
20             WHERE DIAG_YM = TO_CHAR(ADD_MONTHS(TO_DATE('%s','YYYYMM'),-4),'YYYYMM')
30             WHERE RV_YM   = TO_CHAR(ADD_MONTHS(TO_DATE('%s','YYYYMM'),-2),'YYYYMM')
               GROUP BY DIV_TY_CD, MNTR_MDIV_CD
                  , CASE WHEN DIV_TY_CD IN ('1', '2', 'A', 'B', 'X') THEN MNTR_RECU_CL_CD ELSE '$' END
                  , CASE WHEN DIV_TY_CD IN ('1', '2', 'A', 'B', 'X') THEN SIDO_CD ELSE '$' END
          ) A
          JOIN TMJWA418 B ON B.DIV_TY_CD = A.DIV_TY_CD
          -- 행위, 약제, 치료재료대에 따라 일정금액 이상만 분석 대상 포함
           AND ADDC_ADT_AMT > CASE WHEN MNTR_DIV_TY_CD = '1' THEN  5000000
                                   WHEN MNTR_DIV_TY_CD = '2' THEN 38000000
                                   WHEN MNTR_DIV_TY_CD = '3' THEN  1700000
                                                             ELSE 999999999 END
     ) X
```

총 진료비 기준 제외 대상 조건

2. 데이터 전처리

```
164     # 전처리: 항목 기준
165     def preproc_itm(self):
166         # 전처리 기준은 총진료비로 함
167         data = self.df['ADDC_ADT_AMT']
168 
169         s_idx = outlier_pos(data, amt=0.5)
170         _len = len(self.df) - s_idx  # 이상치 분리데이터의 길이
171 
172         if s_idx:  # 잘려진 경우 플래그 지정
173             self.slice_flag = True
174 
175         # 데이터가 올라가는 경우에는 보장성강화 스타일의 항목이기 때문에
176         # 이전 데이터 1건 또한 살려준다 (데이터 길이 <= 3)
177         if not _len > 3 and (s_idx and data[s_idx-1] < data[s_idx]):
178             s_idx = s_idx - 1
179 
180         self.df = self.df.iloc[s_idx:].reset_index(drop=True)
181 
182         for name in self.output_names.values():
183             s = self.df[name]
184 
185             # 데이터가 0인 항목에 대한 전처리
186             for i in s[s == 0].index:
187                 # 마지막 데이터가 0인 경우, 마지막 3개월 이전까지의 평균으로 보정
188                 if i == s.size - 1:
189                     start = 0 if not i > 2 else i - 3
190                     m = s[start:i].mean()
191                     s.iat[i] = m if not np.isnan(m) else 0  # ValueError:
192 
193                 # 첫 데이터가 0인 경우, 다음 데이터의 0.7배로 보정
194                 elif i == 0:
195                     s.iat[i] = s[i+1] * 0.7
196 
197                 # 중간 데이터는 양쪽의 평균으로 보정
198                 else:
199                     s.iat[i] = (s[i-1] + s[i+1]) / 2
200 
201             # 비정상 데이터 발생시 0으로 채움
202             self.df[name] = s.fillna(0)
203 
204         return self.df
```

1) 총 진료비가 0원인 경우 다음과 같이 처리

- -마지막 데이터가 0원인 경우, 이전 3개월의 평균을 보정
- -첫 데이터가 0인 경우 다음 데이터의 0.7배로 보정
- -중간 데이터가 0인 경우 양쪽의 평균으로 보정

2) 비정상적인 상승 또는 하락이 있는 경우 제외

- -진료비가 이전 진료비의 1.5배 상승 또는 감소
- -이전과 이후의 평균 진료비가 가중치 이상인 경우

3. 예측 기간에 따른 모델 선택

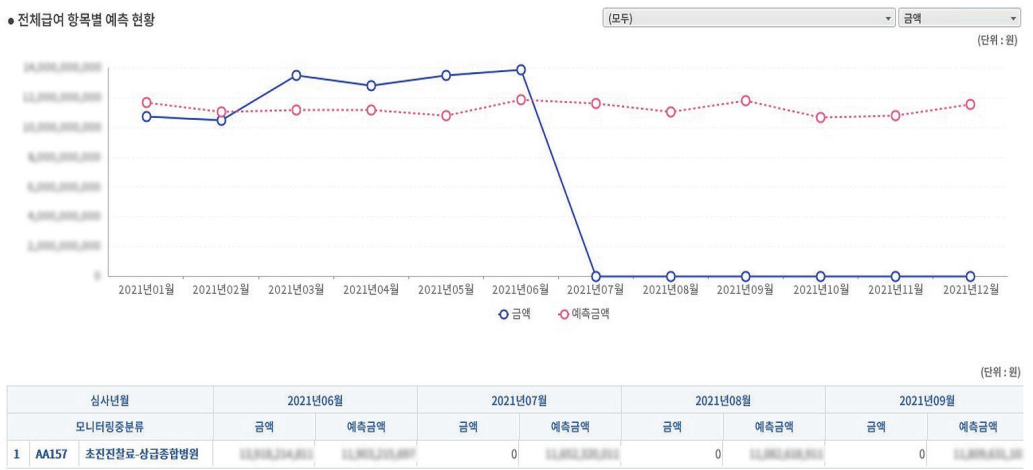
```
87      if not total_preds:
88          # 전처리된 데이터를 사용함
89          df = proc.preproc()
90
91          # 총진료비, 보험자부담금액별 분석
92          for nuv_czitm_cd, name in proc.output_names.items():
93              data = df[name].values
94
95              preds = []
96              # ARIMA 대상인 경우 바로 분석
97              if not preds and int(anal_type):  # S-ARIMA
98                  seasonal = True if data.size >= 24 else False
99                  preds, order, seasonal_order = arima_func(data, proc.predict_size, seasonal=seasonal)
100
101             # 계절성 적용한 ARIMA가 분석되지 않은 경우 제외 후 분석
102             if not preds and int(anal_type):  # ARIMA
103                 preds, order, seasonal_order = arima_func(data, proc.predict_size, seasonal=False)
104
105             # 처리되지 않은 경우
106             if not preds:  #
107                 if len(data) >= 5:
108                     preds = simple_exp_smoothing(data[2:], proc.predict_size)  # 지수평활
109                 else:  # 월별 데이터 기준 진료데이터 비중 적용
110                     pred_temp = grant_weight(data, proc.predict_size, crtr_dd_tp_cd=crtr_dd_tp_cd)
111                     preds = np.ones(proc.predict_size) * data[-1] if proc.slice_flag else pred_temp
112
113             total_preds.append(preds)
```

입력 변수 데이터 보유 기간에 따라 다른 시계열 모델을 적용하여 동일한 기간(24개월) 결과를 예측한다. 예를 들어 새로 생성된 질환이나 수가 코드의 경우 생성된 시점에서 충분한 데이터를 보유하지 않은 경우 ARIMA에서 분석하기 어렵기 때문에 기간에 따라 다른 모델을 선정하여 분석하였다.

입력 변수 보유 기간	분석 모델	예측 기간
24개월 이상	S-ARIMA	24개월
12개월 이상	ARIMA	
5개월 이상	단순지수평활법	
4개월 이하	월별 데이터 기준 진료비 증감률 적용	

〈분류 유형별 예측 제외 대상〉

4. 결과 확인 : 총 진료비 금액 및 본인 부담금

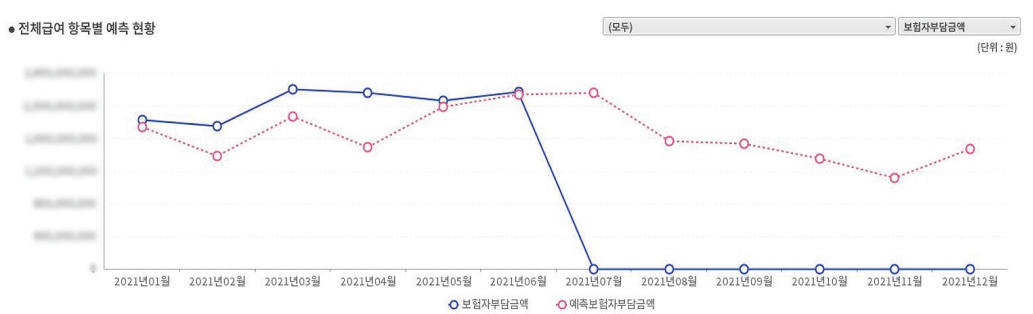


(단위 : 원)

심사년월			2021년06월		2021년07월		2021년08월		2021년09월	
모니터링중분류			금액	예측금액	금액	예측금액	금액	예측금액	금액	예측금액
1	AA157	초전진찰료-상급종합병원	[illegible]	[illegible]	0	[illegible]	0	[illegible]	0	[illegible]

〈총 진료비 예측 화면〉

분석 일자 기준(2021년 6월)으로 입력 변수(가산적용청구금액, 가산적용인정금액, 내원일수, 원외처방일수·건수, 청구·심결본인부담금액, 청구·심결진료비총금액, 인정총사용량실시횟수 등)를 통해 총진료비를 24개월 간 예측한 화면이다.



(단위 : 원)

2021년06월		2021년07월		2021년08월		2021년09월		2021년10월	
보험자부담금액	예측보험자부담금액	보험자부담금액	예측보험자부담금액	보험자부담금액	예측보험자부담금액	보험자부담금액	예측보험자부담금액	보험자부담금액	예측보험자부담금액
[illegible]	[illegible]	0	[illegible]	0	[illegible]	0	[illegible]	0	[illegible]

〈보험자 부담금 예측 화면〉

분석 일자 기준(2021년 6월)으로 입력 변수(가산적용보험자부담금액, 가산적용인정 보험자부담금액, 내원일수, 원외처방일수·건수, 청구·심결본인부담금액, 청구·심결진료비총금액, 인정총사용량실시횟수 등)를 통해 보험자부담금을 24개월 간 예측한 화면이다.

〈부록〉 학습에 도움이 되는 사이트 모음

1. https://www.kaggle.com (캐글)
   - https://www.kaggle.com/competitions
   - https://www.kaggle.com/rankings
   - https://www.kaggle.com/progression
2. https://dacon.io (한국형 캐글)
3. https://www.kmooc.kr (한국형 무크)
4. https://www.kocw.net (열린 대학 강좌)
5. https://kdata.or.kr (한국데이터산업진흥원)
6. https://dataonair.or.kr (한국데이터산업진흥원 데이터 교육)
7. https://sti.kostat.go.kr (통계청 교육센터)
8. https://tong.kostat.go.kr (통계청 동그라미)
9. https://youtu.be/d_YnllRC1Gw (이상철 교수, 시리즈 청강)
10. https://youtu.be/0-e4bHkwKek (구자환 교수 시리즈 청강)
11. https://tacademy.skplanet.com (T아카데미)
12. https://www.python.org (파이썬)
13. https://keras.io/ko (케라스)
14. https://pytorch.kr (파이토치)
15. https://www.tensorflow.org (텐서플로우)

파이썬을 활용한 데이터·AI 분석 사례

발 행 일 2021년 9월 14일

발 행 인 김선민

발 행 처 건강보험심사평가원

주 소 강원도 원주시 혁신로 60(반곡동)

홈 페 이 지 www.hira.or.kr

편 찬 위 원 최동진 정보운영실장

편 찬 실 무 정보운영실 심사정보표준화부
신윤기 부장
한미순 팀장
양승수 과장
유동한 대리
이종욱 대리