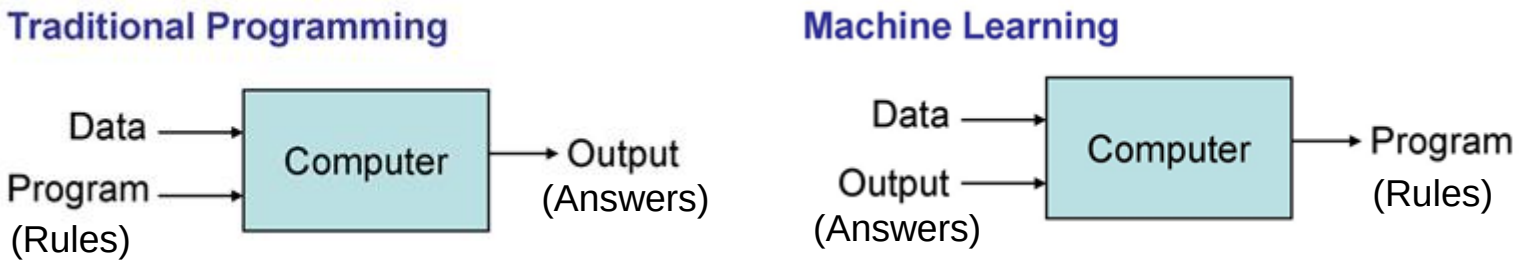
Machine Learning : Concepts, Methods and Tools

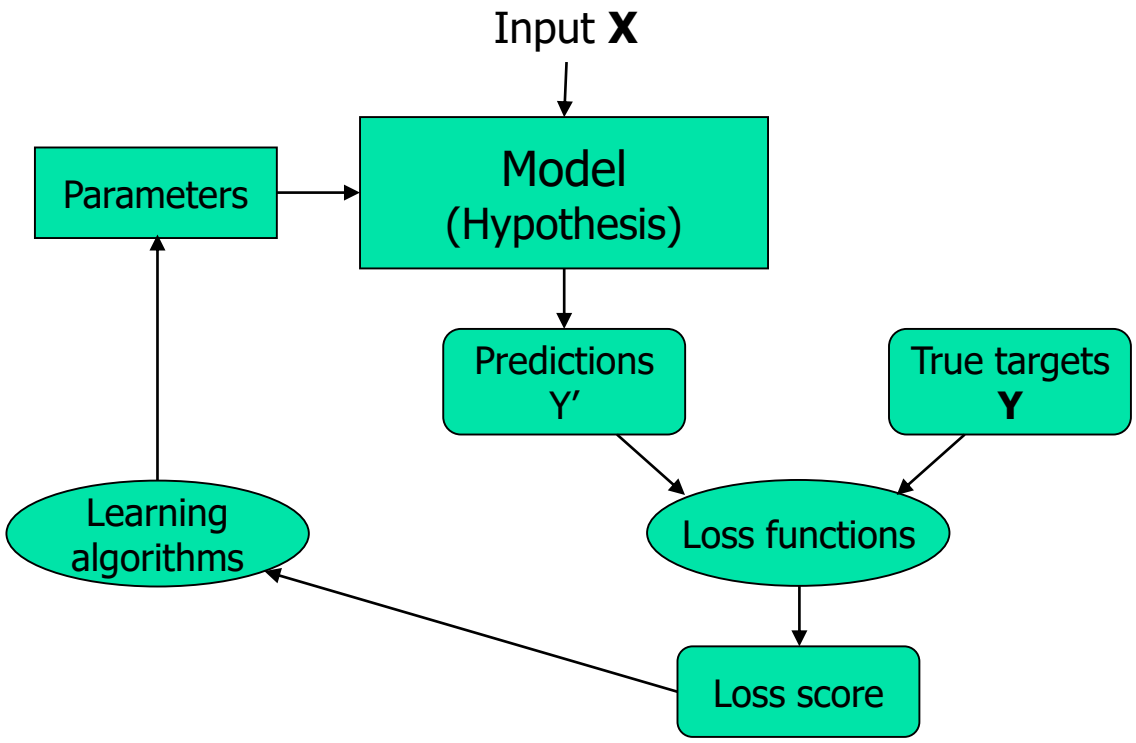
머신러닝(Machine Learning)이란 ?

- Limitations of explicit programming
  - Spam filter: many rules
  - Automatic driving: too many rules

- Machine learning
  - "Field of study that gives computers the ability to learn without being explicitly programmed", Arthur Samuel (1959)



Machine Learning Procedure



왜 머신러닝을 사용하는가?

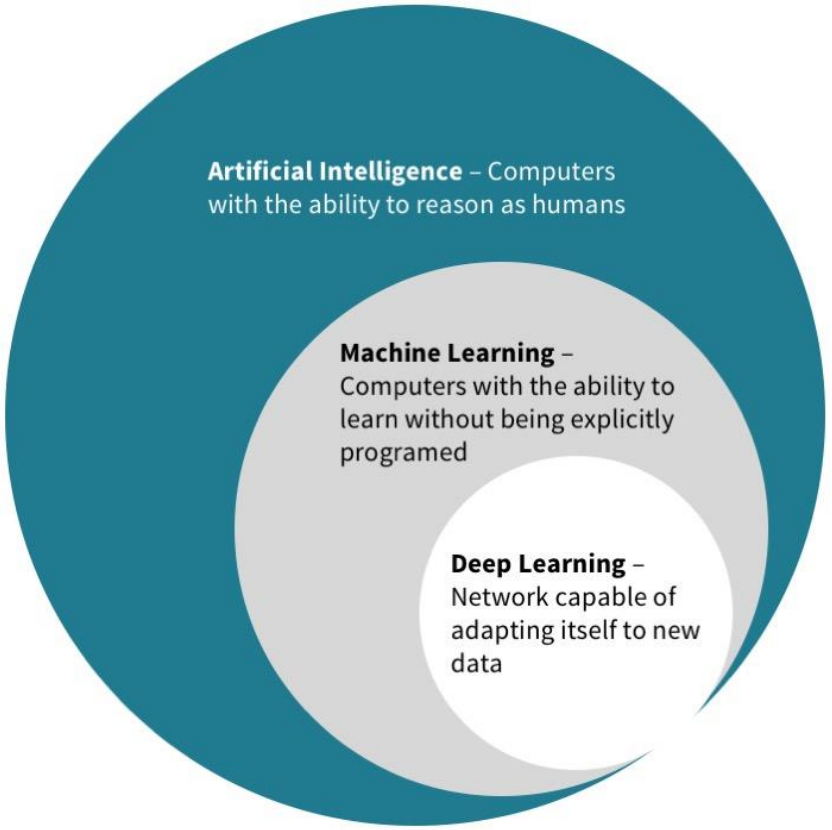
- 기존 솔루션으로는 많은 hand-tuning과 규칙이 필요한 문제의 경우 머신러닝을 이용하면 간단한 코드로 더 나은 성능을 얻을 수 있다.
- 전통적인 방법으로는 해결이 불가능한 복잡한 문제에 있어서 머신러닝이 새로운 해결책이 될 수 있다.
- 머신러닝은 변화가 심한 환경에서 새롭게 생성되는 데이터에 적응력이 뛰어나다.
- 머신러닝은 복잡한 문제와 대용량 데이터로부터 통찰을 얻을 수 있게 해준다.

Source: "Hands-On Machine Learning with Scikit-Learn and TensorFlow" , urélien Géron

혼동되어 사용되는 용어

- Data Mining
  - Data analysis processes that apply ML techniques to solving real world problems

- AI & Deep Learning

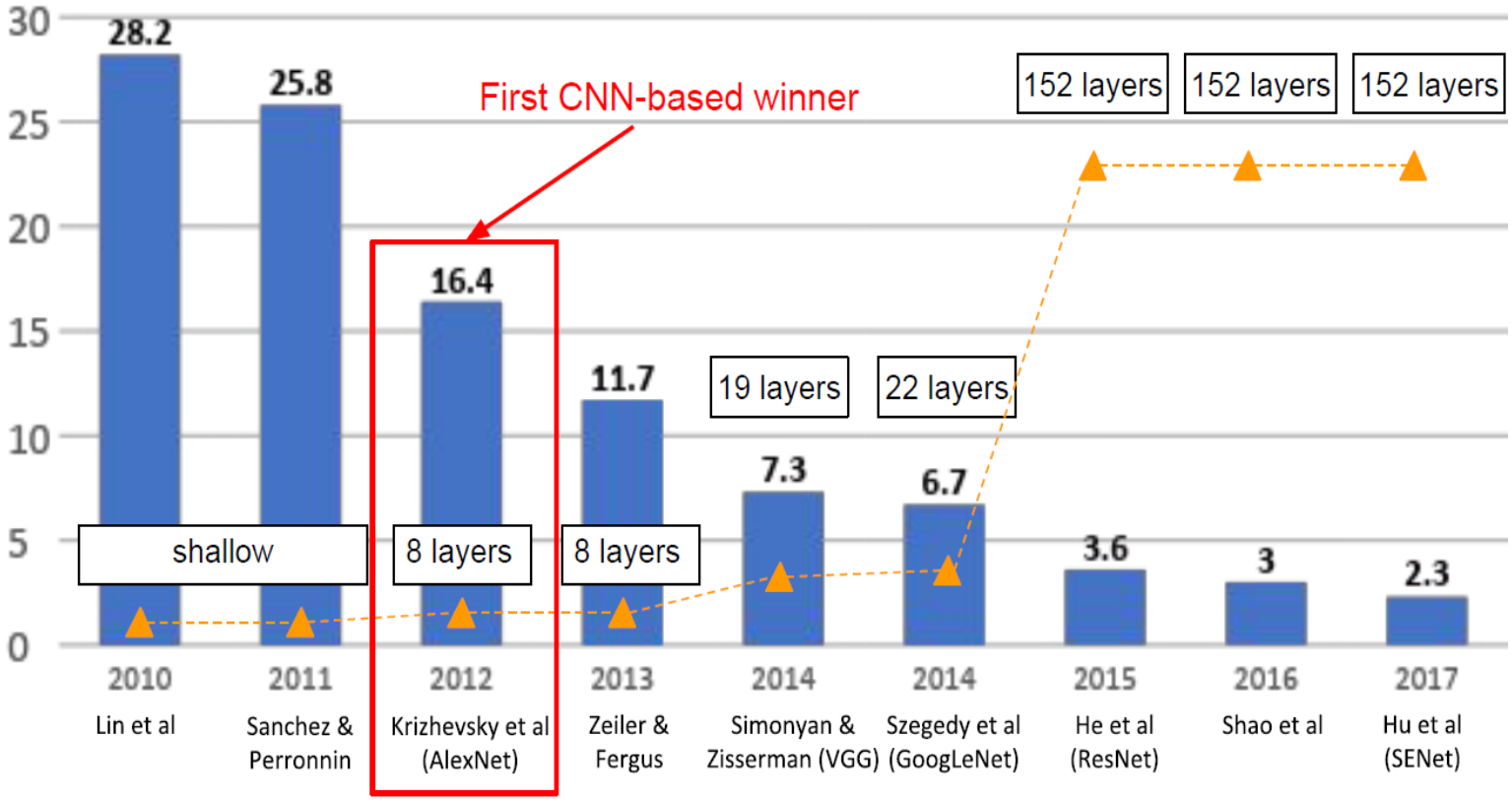


Source: https://leverton.de/blog/2016/12/21/3-deep-learning-trends-2017/

인공지능 三大將

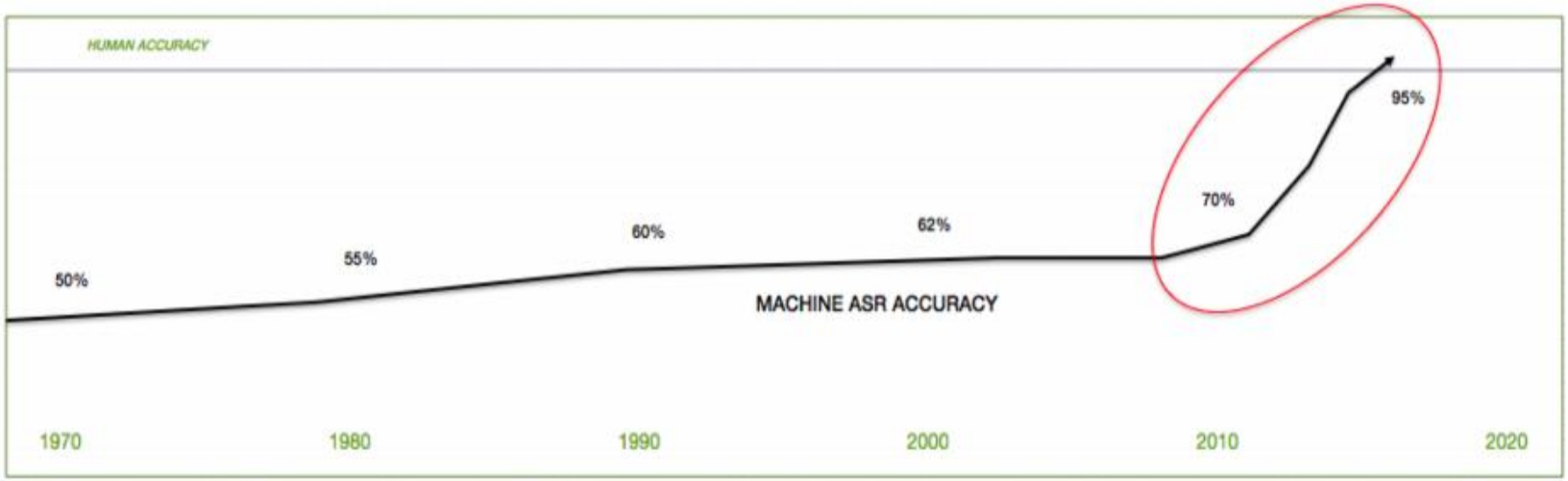
- Computer Vision – Image Recognition
- Automatic Speech Recognition (ASR)
- Natural Language Processing – Machine Translation

- ILSVRC ImageNet challenge
  - The images are large (256 pixels high) and there are 1,000 classes, some of which are really subtle (try distinguishing 120 dog breeds)
  - The top-5 error rate for image classification fell from over 26% to barely over 3% in just five years
    (human error rate: 5.1%)



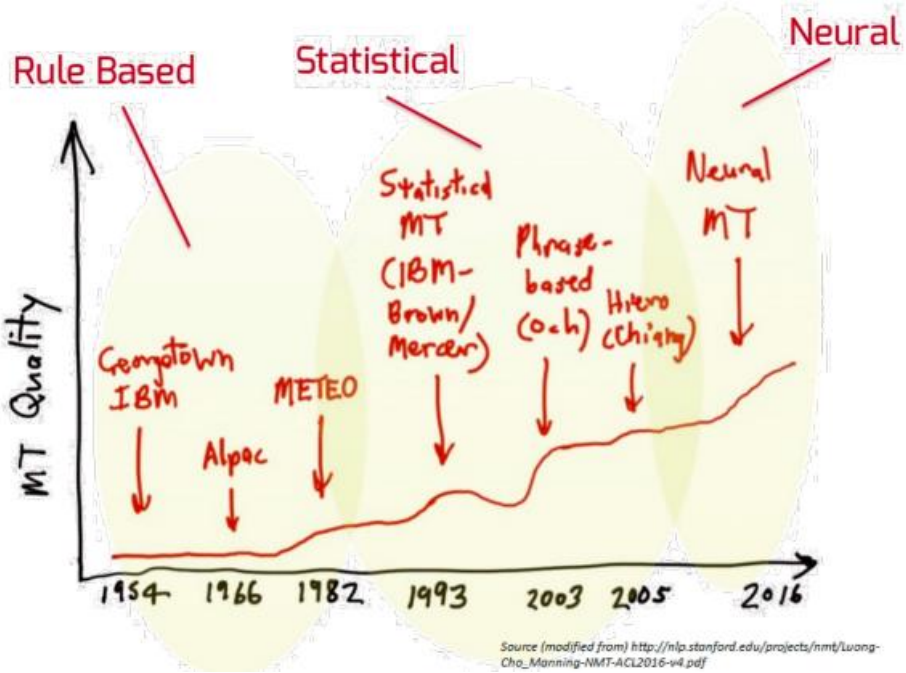
source: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture09.pdf

Speech Recognition



- 2012년부터 딥러닝을 활용하여 큰 발전
- 오히려 이 분야에서는 vision분야에 비해서 딥러닝 기술을 활용하여 상용화에까지 성공한 더욱 인상적인 사례

Neural Machine Translation



- 2014년 Sequence-to-sequence(seq2seq)가 소개됨
- 기계번역은 가장 늦게 혁명이 이루어졌지만, 가장 먼저 딥러닝만을 사용해 상용화가 된 분야
- 현재의 상용 기계번역 시스템은 모두 딥러닝으로 대체

통계분석 vs. 머신러닝

- 통계분석

  대상집단이 있으며, 모집단의 분포 혹은 모형 등 여러 가지 가정을 전제로 하게 되며 이 전제 조건하에서 분석을 실시

  - 표본의 관찰을 통해 모수 전체를 추론하는 과정
  - (연역적 접근법) 경험을 토대로 가설(모형)을 세움 ⇒ 표본에 적용하여 가설 검증

- 머신러닝

  표본조사/실험에서 필연적으로 수반되는 분포라든가 모형에 대한 전제조건이 필요하지 않음

  - 모집단의 전체자료를 이용하여 정보화하는 과정
  - (귀납적 접근법) 알고리즘이 데이터로부터 가설(모형)을 스스로 생성 ⇒ 데이터에 적용하여 검증(일반화)

단순회귀모형

$$y = \beta_0 + \beta_1 x + \epsilon$$

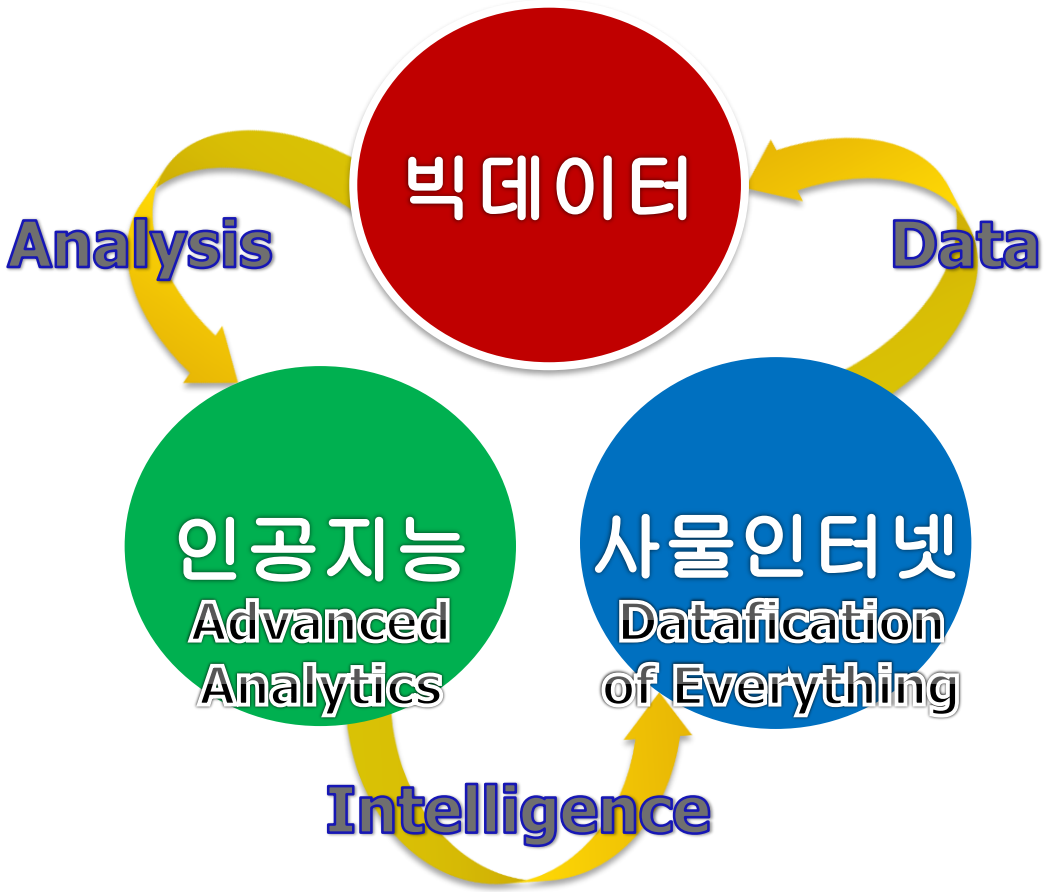
- β_0: y-절편 (모수)
- β_1: 기울기 (모수)
- ε: 오차항 (확률변수: 평균 0, 분산 σ^2)
- 추정된 회귀모형: $\hat{y} = b_0 + b_1 x$

회귀모형의 가정

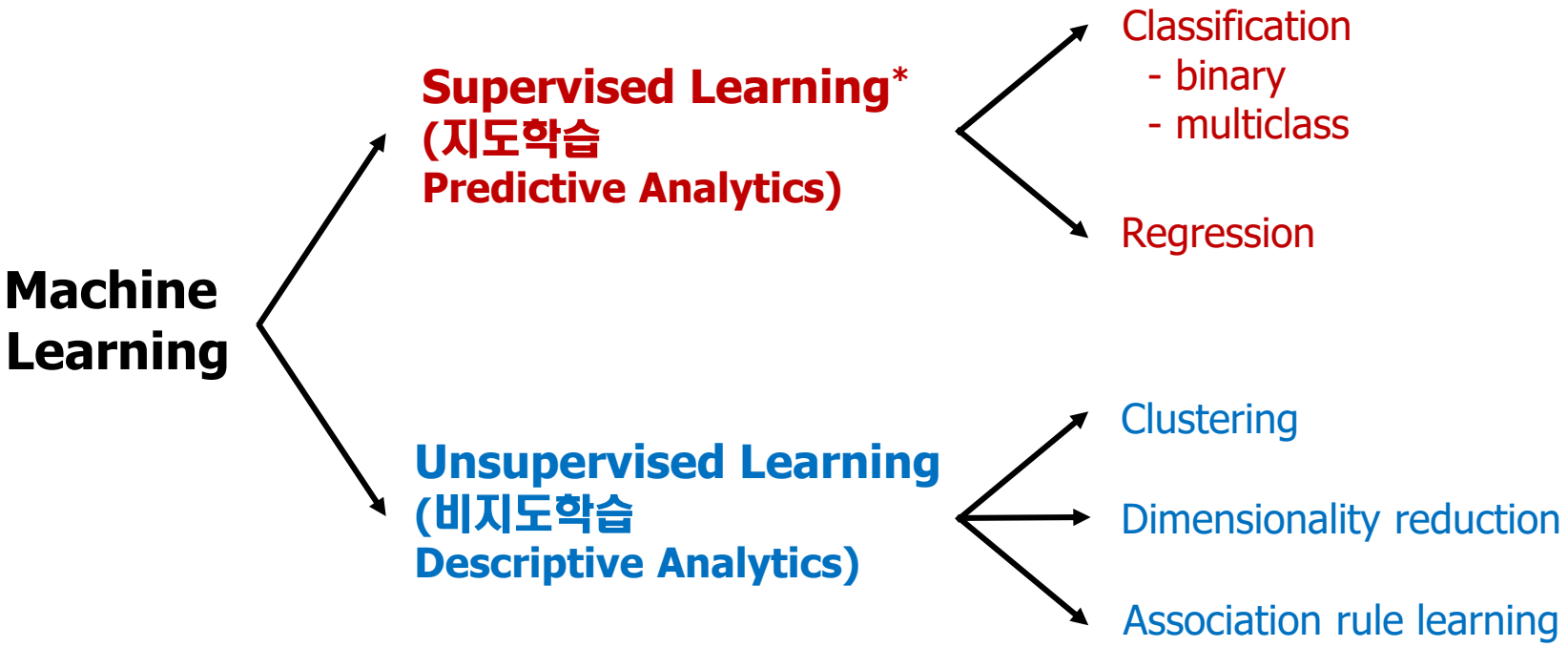
- y 값들은 서로 독립이다.
- y 와 x는 선형관계이다.
- $y|x$ (같은 x 값에 대한 y)의 분산은 동일하다.
- $y|x$는 정규분포를 따른다.

현재 가장 Hot한 3가지 정보기술

3가지 정보기술의 역할과 관계

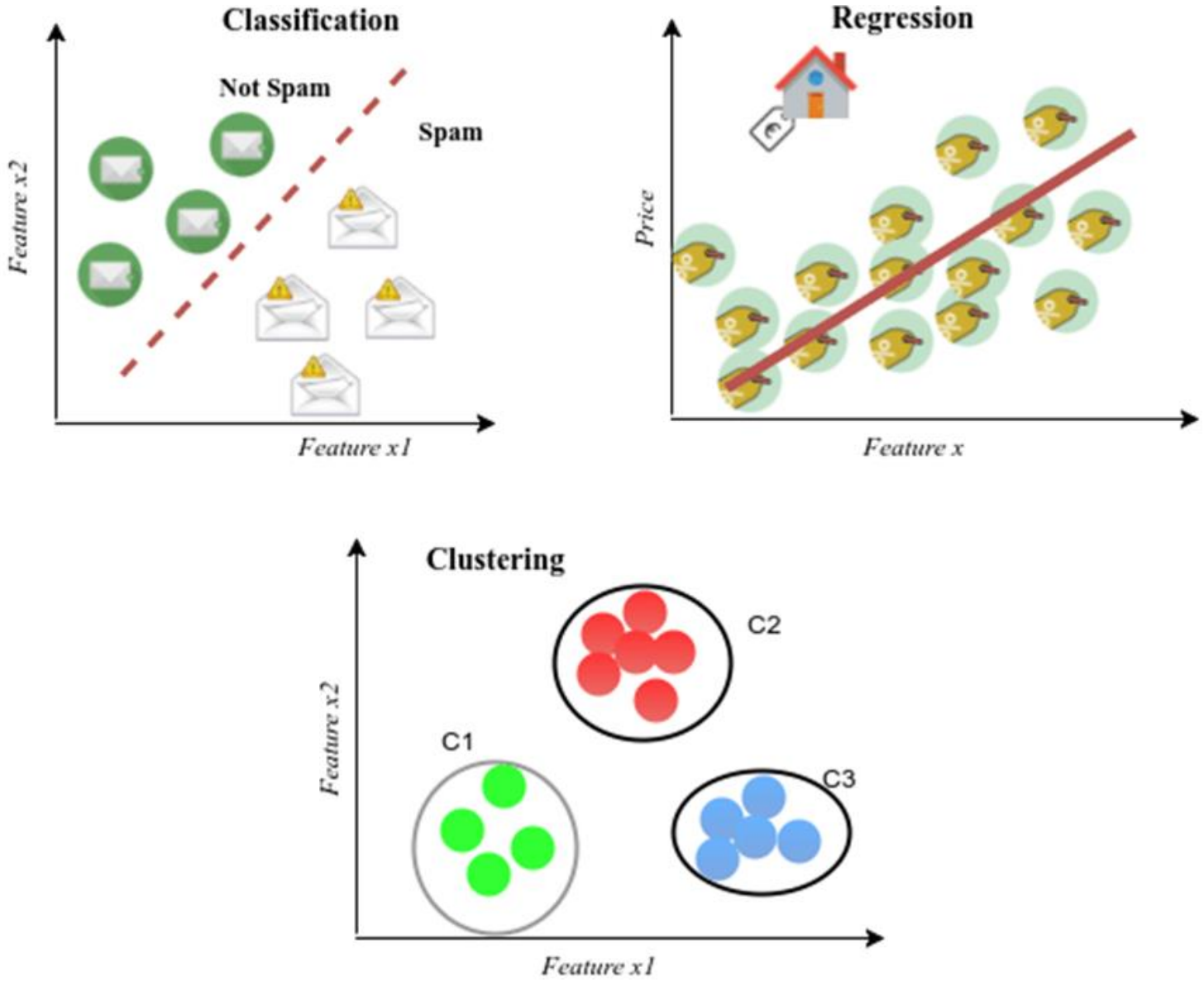


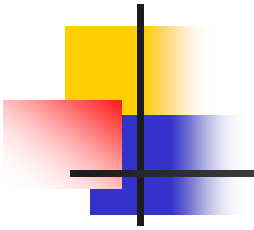
머신러닝의 유형



☞ *Some researchers classify Reinforcement Learning into a type of machine learning*

* *전체 머신러닝 사례의 **95%** 이상을 차지*


Classification
Not Spam
Spam
Feature x2
Feature x1
Regression
Price
Feature x
Clustering
C1
C2
C3
Feature x2
Feature x1



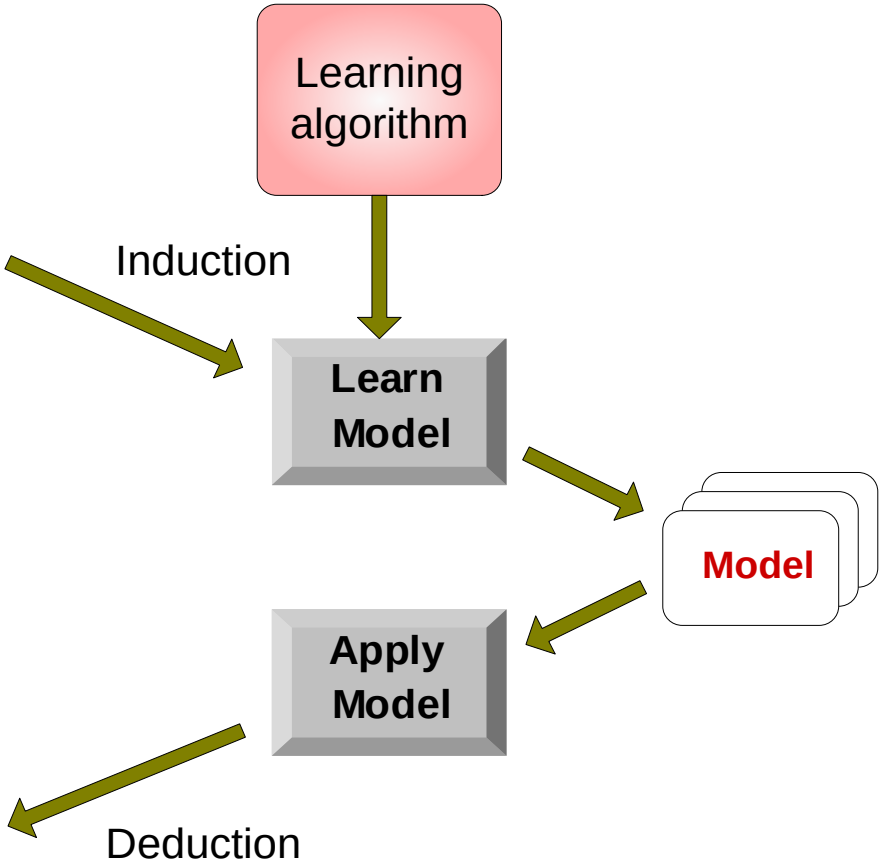
Naming Conventions

- **Features = predictor variables = independent variables**
- **Class = target variable = dependent variable**

분류(Classification)분석의 개념

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

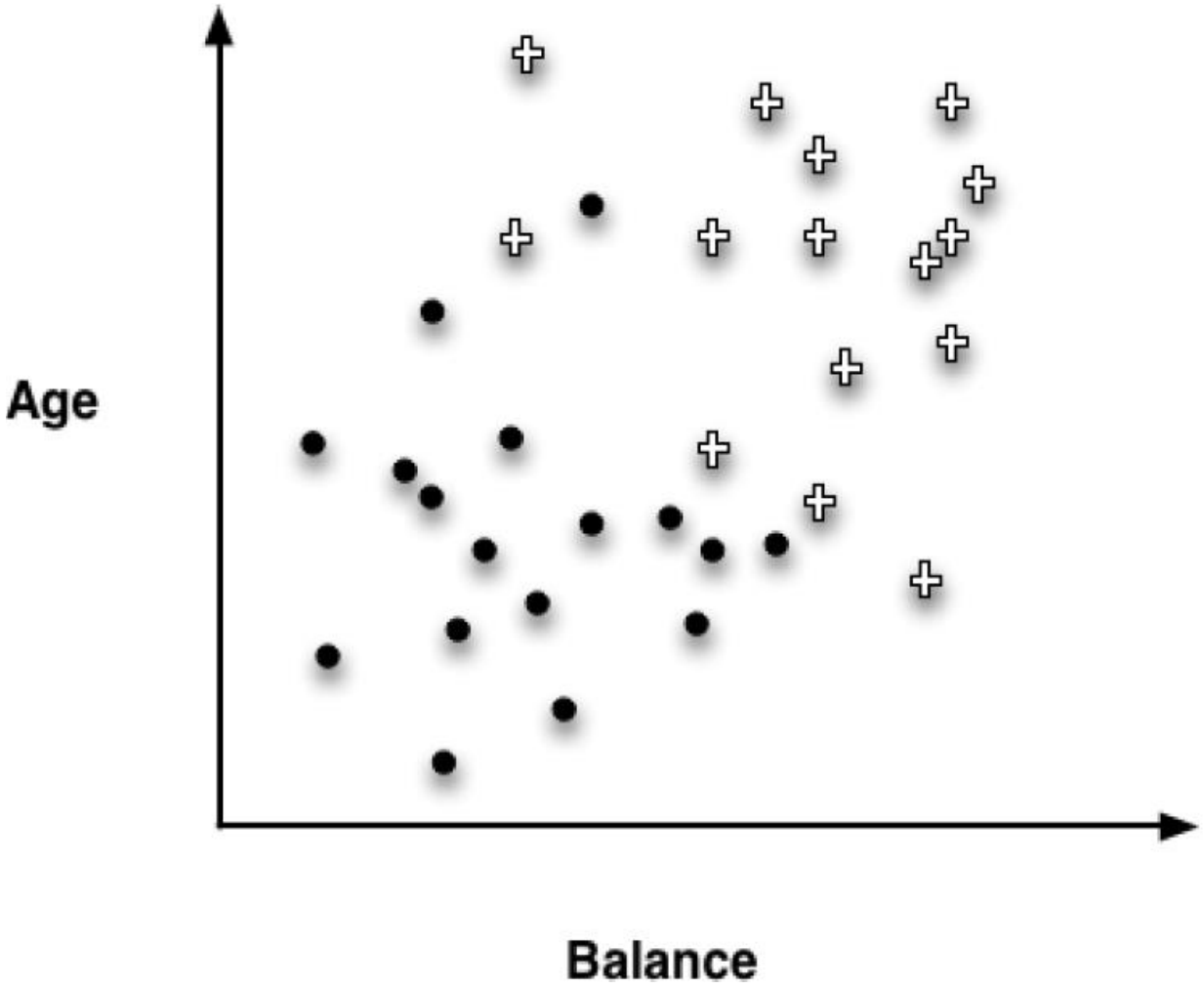


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

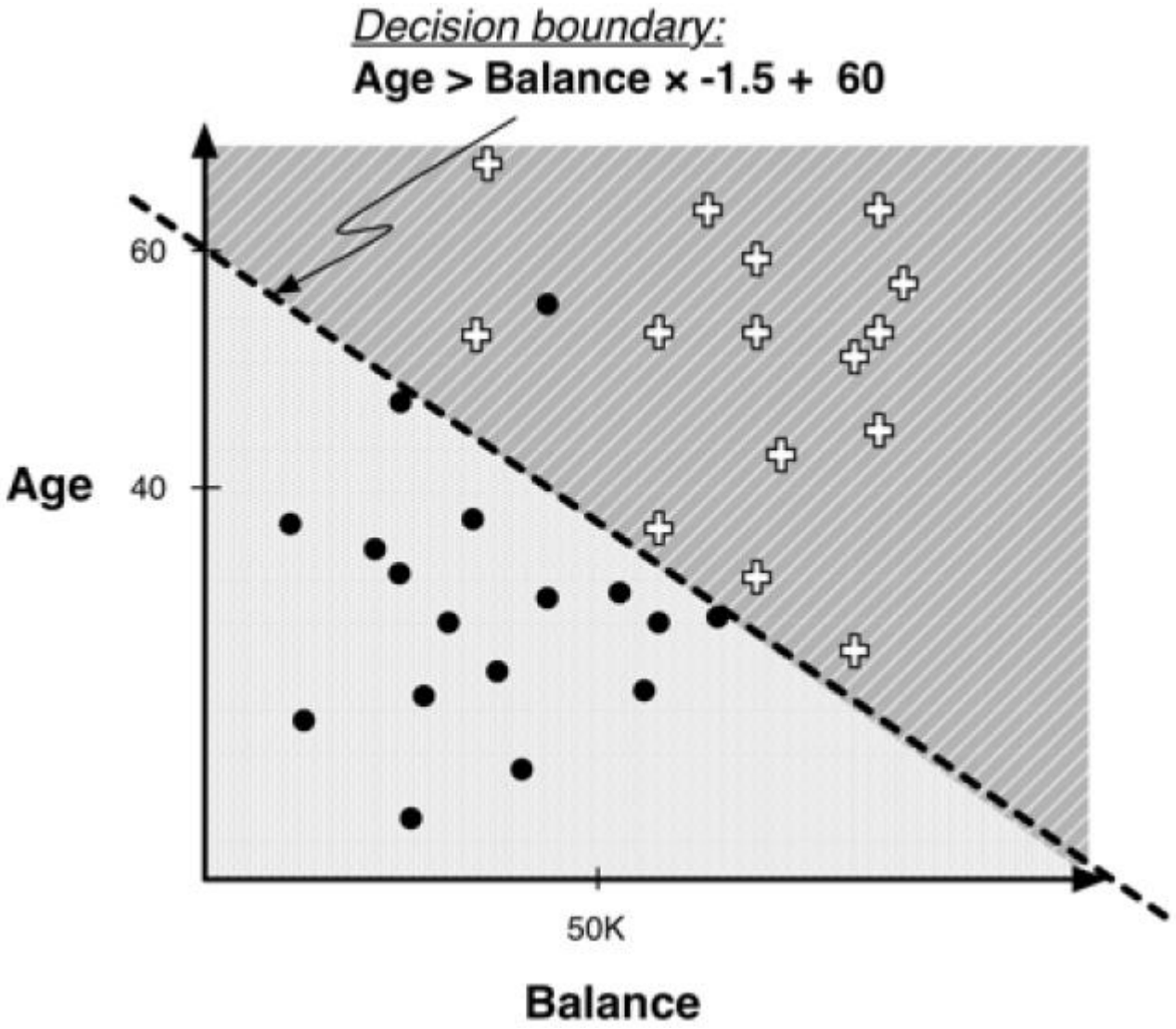
Source: Introduction to Data Mining by Tan, Steinbach & Kumar

분류분석 기법 (시각적 해석)



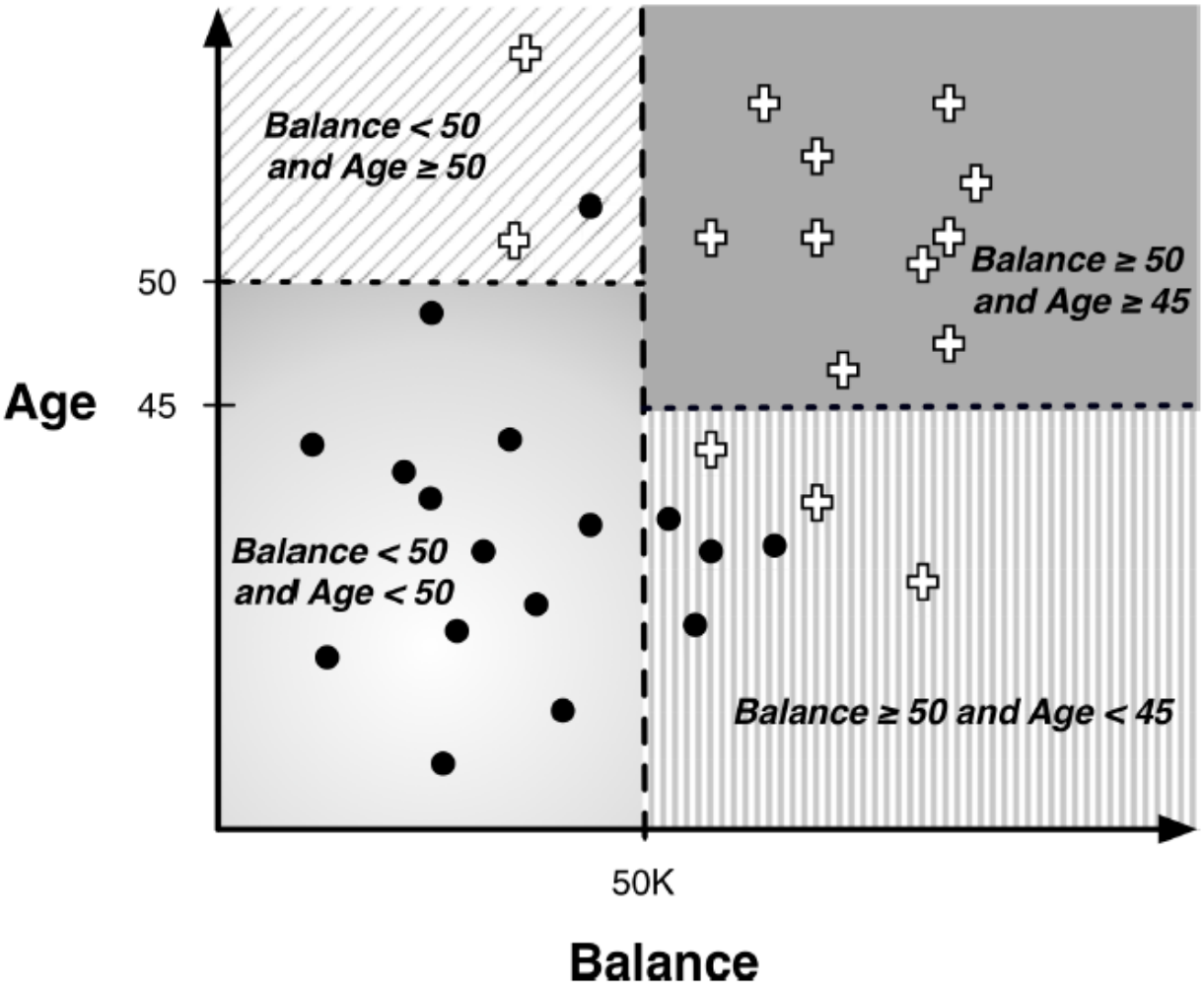
Source: Data Science for Business by Provost & Fawcett

분류분석 기법
– Logistic Regression (or SVM)



Source: Data Science for Business by Provost & Fawcett

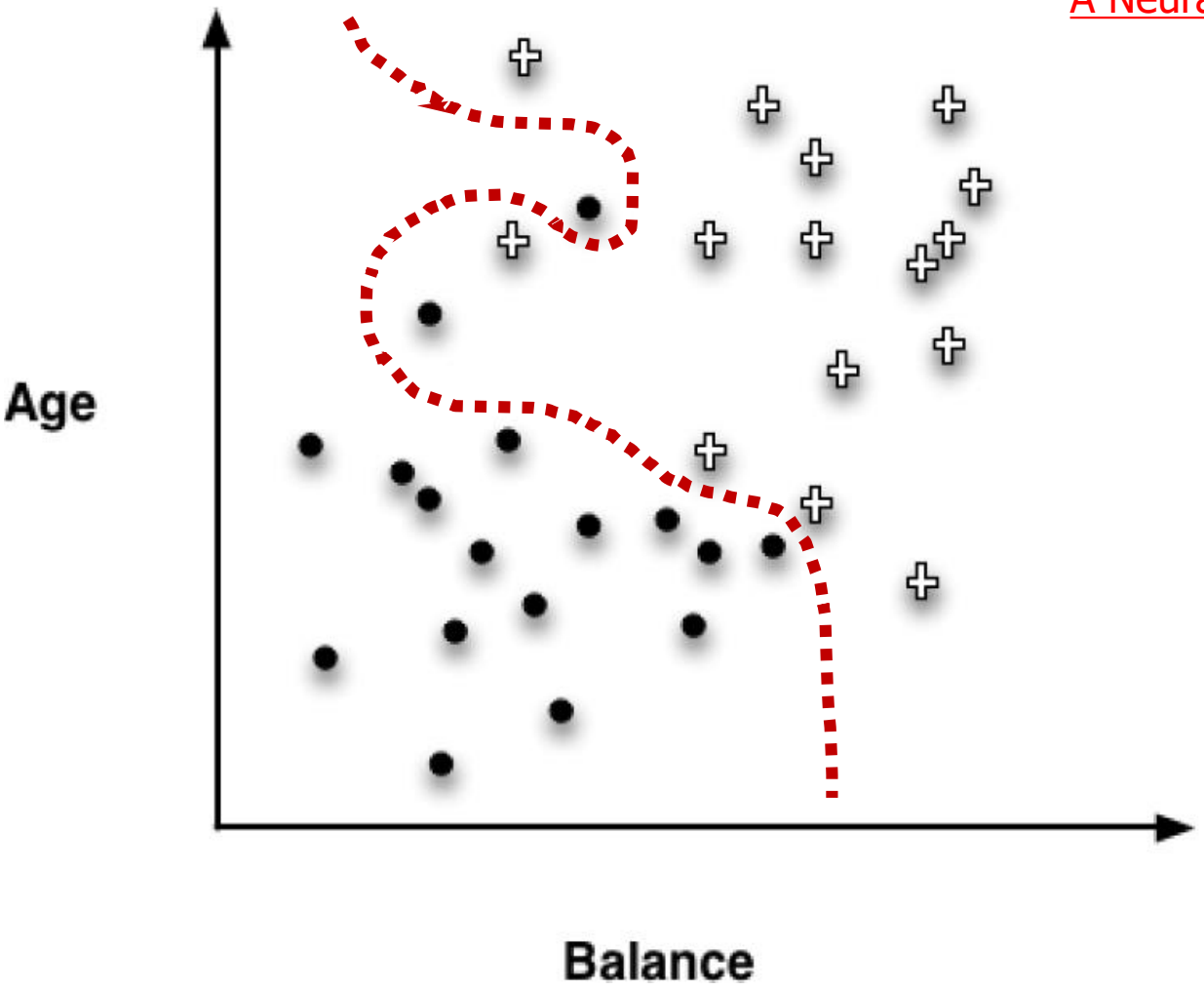
분류분석 기법
– Decision Tree based Methods



Source: Data Science for Business by Provost & Fawcett

분류분석 기법
– Neural Networks

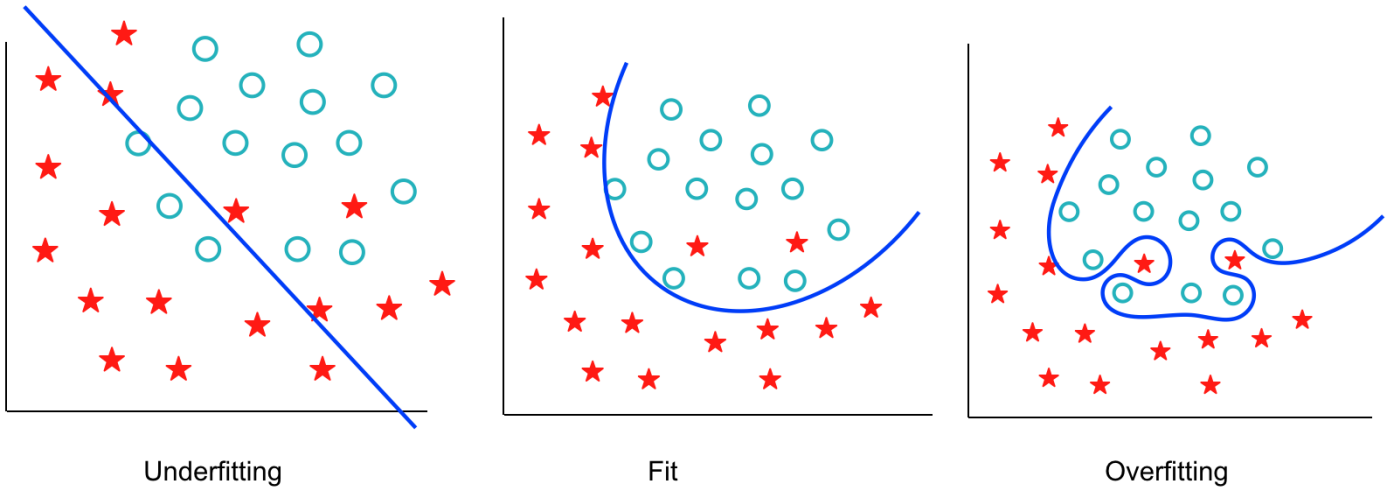
A Neural Network Playground



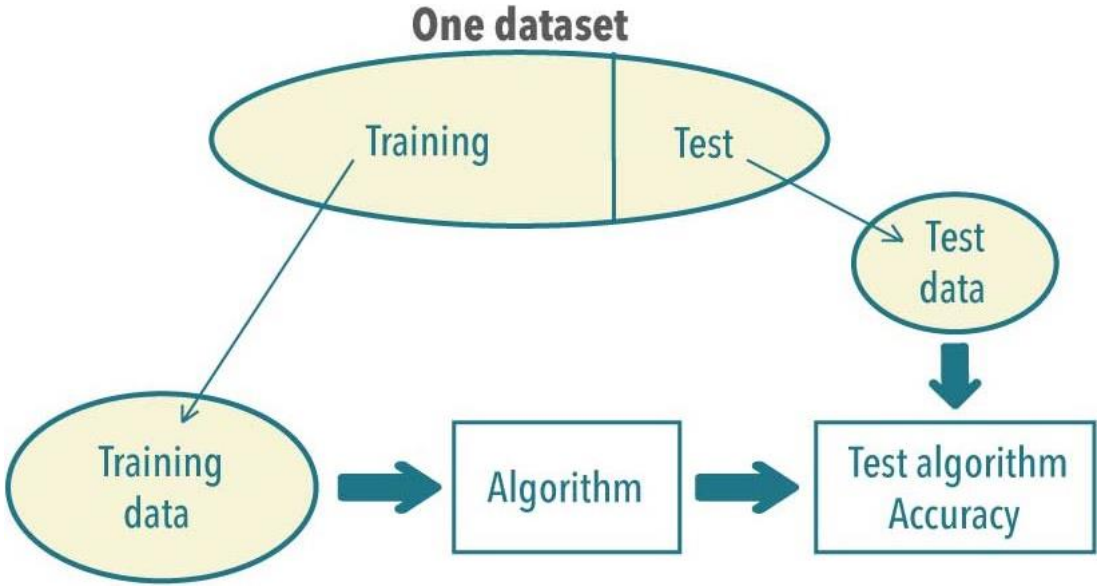
Source: Data Science for Business by Provost & Fawcett

Overfitting vs. Underfitting

- 과대적합(overfitting)
  - 모델이 데이터에 필요이상으로 적합한 모델
  - 데이터 내에 존재하는 규칙 뿐만 아니라 불완전한 레코드도 학습
- 과소적합(underfitting)
  - 모델이 데이터에 제대로 적합하지 못한 모델
  - 데이터 내에 존재하는 규칙도 제대로 학습하지 못함



데이터 분할(Data Partitioning)



- 모형의 Overfitting 여부(일반화 가능성)를 검증하기 위해 데이터를 학습용과 평가용으로 분리
- 학습데이터 (Training data) → 모형적합(Model Fitting)
- 평가데이터 (Test data) → 모형평가(Model Evaluation)
- 60%(학습) – 40%(평가) 분할, 75% – 25% 분할을 주로 사용

분류분석이 적용될 수 있는 비즈니스 문제

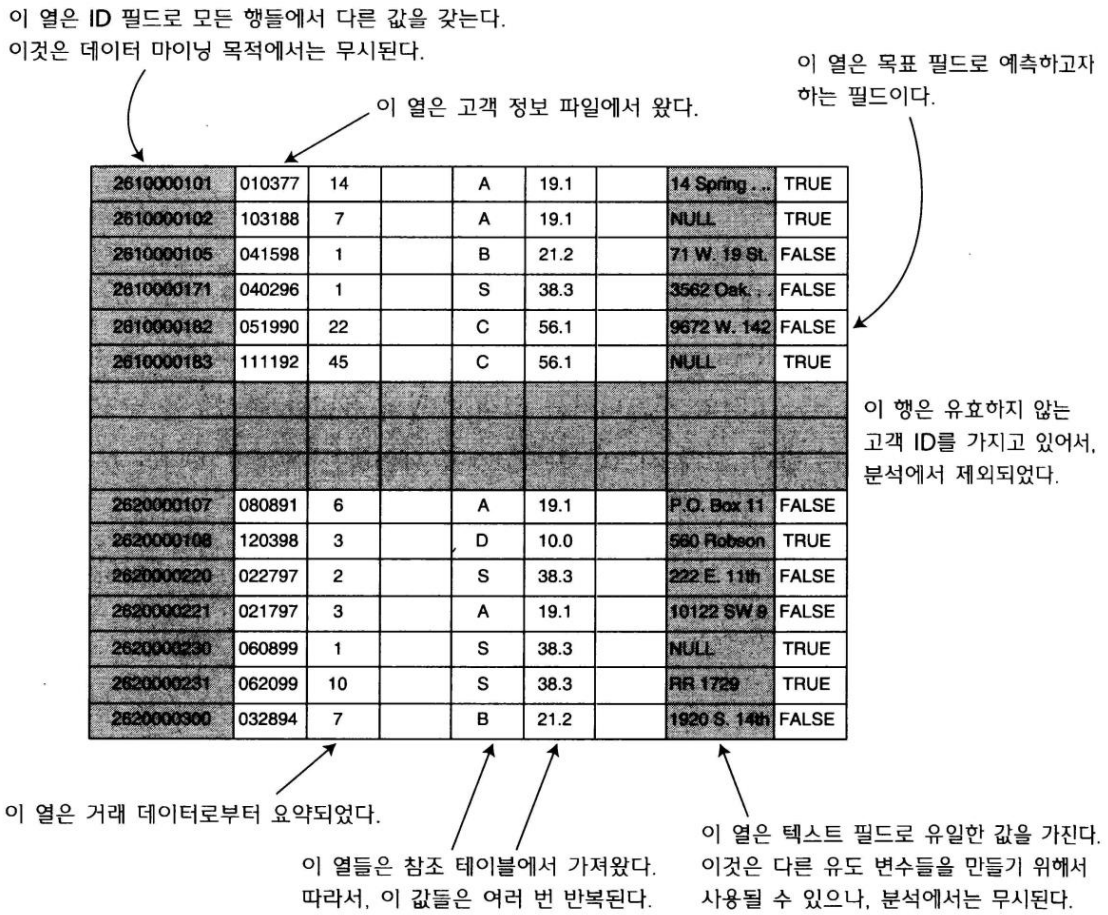
- 특정 상품(Ex: 퇴직연금)에 가입할 가능성이 높은 고객은 누구인가?
- 추가 가입을 유도하기 위해 대상고객에게 어떤 상품을 추천해야 하는가?
- 향후 6개월 안에 이탈(Ex: 계약 만기 전에 실효 또는 중도 해약)할 가능성이 높은 고객들은 누구이며 그들의 특징은?
- 고객별 LTV(Life Time Value)를 계산해 주는 모델은?
  - 예: LTV = 현재 고객의 기여가치 + 추가구매확률에 의한 기여가치 – 이탈확률에 의한 손실 가치
- 담보대출 고객 중에서 누가 향후 90일 내에 조기상환할 것인가?
- 각 고객별로 클릭할 가능성이 가장 높은 광고는 어떤 것인가?
- 고객별로 어떤 할인쿠폰이 다음달에 사용될 것인가?
- 추후 VIP의 고객이 될 가능성이 높은 고객들은?
- 누가 차량사고를 낼 것인가?
- 누가 이직할 것인가?

지도학습이 발견한 이상하고 놀라운 사실

- 배너광고를 본 사람 중 61%는 그와 관련된 검색을 할 가능성이 높다.
- 맥 사용자들은 상대적으로 더 비싼 호텔을 예약한다.
- 금융 및 주식 사이트는 오후1시 직후가 접속 피크다.
- 이메일 주소는 그 사람의 충성도를 암시한다.
- 신용카드를 술집에서 자주 사용한다면 결제대금을 연체할 위험이 높다.
- 신용등급이 낮을수록 자동차 사고를 낼 확률이 높다.
- 이미 여러 개의 계좌를 개설한 고객들에게 우편물을 보내면 오히려 그들이 더 많은 계좌를 개설할 가능성을 낮춘다.
- 약정기간이 끝났는지 온라인으로 알아보는 고객은 경쟁업체로 넘어갈 가능성이 높다.
- 직무순환을 많이 한 직원일수록 회사에 더 오래 다니는 경향이 있다.
- 스테이플러는 일자리를 의미한다.

Source: "예측 분석이다 (빅데이터의 다음 단계는)", 에릭 시겔

머신러닝에서 요구되는 데이터구조



2610000101	010377	14		A	19.1		14 Spring . . .	TRUE
2610000102	103188	7		A	19.1		NULL	TRUE
2610000105	041598	1		B	21.2		71 W. 19 St.	FALSE
2610000171	040296	1		S	38.3		3562 Oak. . .	FALSE
2610000182	051990	22		C	56.1		9672 W. 142	FALSE
2610000183	111192	45		C	56.1		NULL	TRUE
2620000107	080891	6		A	19.1		P.O. Box 11	FALSE
2620000108	120398	3		D	10.0		560 Robson	TRUE
2620000220	022797	2		S	38.3		222 E. 11th	FALSE
2620000221	021797	3		A	19.1		10122 SW 9	FALSE
2620000230	060899	1		S	38.3		NULL	TRUE
2620000231	062099	10		S	38.3		RR 1729	TRUE
2620000300	032894	7		B	21.2		1920 S. 14th	FALSE

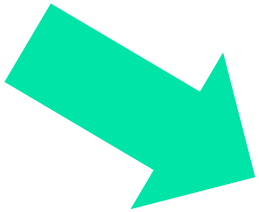
- ✓ 모든 데이터가 하나의 테이블에 존재해야 한다.
- ✓ 각 행은 기업과 관련 있는 한 개체(Ex: 고객)에 대응해야 한다.
- ✓ 하나의 값을 갖는 필드는 무시되어야 한다.
- ✓ 대부분이 한 값을 갖는 필드도 가급적 무시되어야 한다.
- ✓ 각 행마다 다른 값들을 가지는 필드는 무시되어야 한다.
- ✓ 목표필드와 지나치게 높은 상관관계를 갖는 필드는 제거되어야 한다.

Source: "경영을 위한 데이터마이닝", Michael J. A. Berry, Gordon S. Linoff

- A Clickstream Analysis Case

	CUS_ID	TIME_ID	SITE	SITE_CNT	ST_TIME	SITE_NM	BACT_NM	MACT_NM	ACT_NM
1	1	2012070905	search.naver.com	3	794	네이버 검색	인터넷/컴퓨터	검색	포털검색
2	1	2012072507	plus.google.com	1	1	구글 Plus	커뮤니티	블로그/SNS	SNS
3	1	2012081116	joongang.joinsmsn.com	2	5	중앙일보	뉴스/미디어	일간지	종합일간지
4	1	2012090304	news.naver.com	5	504	네이버 뉴스	뉴스/미디어	인터넷신문	포털뉴스
5	1	2012090506	news.nate.com	1	0	네이트 뉴스	뉴스/미디어	인터넷신문	포털뉴스
6	1	2012091004	plus.google.com	2	66	구글 Plus	커뮤니티	블로그/SNS	SNS
7	1	2012092017	plus.google.com	2	23	구글 Plus	커뮤니티	블로그/SNS	SNS
8	1	2012122801	news.naver.com	3	213	네이버 뉴스	뉴스/미디어	인터넷신문	포털뉴스
9	1	2012123114	search.naver.com	1	0	네이버 검색	인터넷/컴퓨터	검색	포털검색
10	1	2013061008	blog.naver.com	1	46	네이버 블로그	커뮤니티	블로그/SNS	포털블로그

	CUS_ID	QRY_STR	QRY_CNT
1	1	못내	1
2	1	배트맨 리부&acr=1&qdt=0&ie=utf8&query=배트맨 리부트	1
3	1	배트맨 비긴즈 명대사&sm=top_sug.pre&fbm=0&acr=9…	1
4	1	베수비&acr=1&qdt=0&ie=utf8&query=베수비오 화산	1
5	1	베이징올림픽 장미란	1
6	1	베이징올림픽 장미란 몸무게	1
7	1	베이징올림픽 장미란 코로브카	1
8	1	별이 빛나는 밤&sm=top_sug.pre&fbm=0&acr=2&acq=…	1
9	1	본 레거시 첫주 박스오피스	1
10	1	본 슈프리머시 첫주 박스오피스	1



	CUS_ID	AGE
1	1	40
2	2	20
3	3	20
4	4	30
5	5	40
6	6	40
7	7	40
8	8	30
9	9	40
10	10	30

Clickstream Data로부터 만들 수 있는 Feature

필드 명	필드 개수	내용
DWELLTIME	1	총 체류시간(ST_TIME의 총합)
PAGEVIEWS	1	총 페이지뷰(SITE_CNT의 총합)
VSITES	1	접속한 서로 다른 사이트(SITE_NM)의 수
COVERAGE	1	총 22개의 사이트 카테고리(BACT_NM)에 얼마나 다양하게 접속했는지에 대한 비율("서로 다른 카테고리 수/22"로 계산. 예: 22개 카테고리에 모두 접속한 경우는 1, 11개만 접속한 경우는 0.5)
SITECOV	1	사이트 카테고리(BACT_NM) 별 체류시간 변동계수(카테고리별 체류시간의 "표준편차/평균" 값)
VDAYS	1	총 접속 일수
DAYTIME	1	하루 평균 체류시간
DAYCOV	1	일별 변동계수(일일 체류시간의 "표준편차/평균" 값)
CT* (*: BACT_NM)	22	사이트 카테고리 별 체류시간 비율. 즉, 전체 체류시간(ST_TIME) 중 특정 카테고리(BACT_NM)에 얼마나 머물렀는가를 비율로 계산. BACT_NM은 총 22개임
DF* (*: 월요일 ~ 일요일)	7	요일 당 체류시간 비율
HF* (*: 0005, 0611, 1217, 1823)	4	시간대별(0-5시, 6-11시, 12-17시, 18-23시) 체류시간 비율

A Kaggle Competition Case

www.kaggle.com/c/talkingdata-adtracking-fraud-detection

File descriptions

- train.csv - the training set
- train_sample.csv - 100,000 randomly-selected rows of training data, to inspect data before downloading full set
- test.csv - the test set
- sampleSubmission.csv - a sample submission file in the correct format

Data fields

- ip: ip address of click.
- app: app id for marketing.
- device: device type id of user mobile phone (e.g., iphone 6 plus, iphone 7, huawei mate 7, etc.)
- os: os version id of user mobile phone
- channel: channel id of mobile ad publisher
- click_time: timestamp of click (UTC)
- attributed_time: if user download the app for after clicking an ad, this is the time of the app download
- is_attributed: the target that is to be predicted, indicating the app was downloaded
- Note that ip, app, device, os, and channel are encoded.

The test data is similar, with the following differences:

- click_id: reference for making predictions
- is_attributed: not included

How to derive features

- 한 값으로부터 특징들을 추출한다.
  - 날짜로부터 요일을 계산
  - 신용카드번호로부터 신용카드 발급사를 추출
- 한 레코드 내의 값들을 결합한다.
  - 멤버십 가입일과 첫 구매일로부터 경과를 계산
- 다른 테이블의 부가적인 정보를 참조한다.
  - 우편번호에 따른 인구와 평균가계수입
  - 상품코드에 대한 계층 구조
- 다수 필드 내에 시간 종속적인 데이터를 pivoting한다.
  - 월마다 한 행씩 저장되는 과금 데이터를 각각의 월에 대응하는 필드로 변환
- 거래 레코드들을 요약한다.
  - 년간 총 구매액
- Customer Signature 필드들을 요약한다.
  - 값의 표준화 및 서열화

Traditional ML vs. deep learning

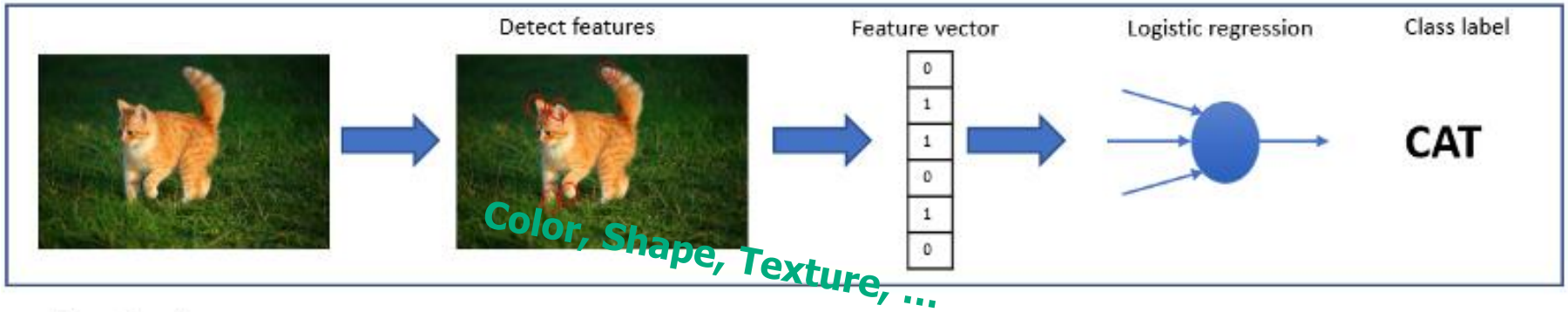
Traditional Machine Learning Flow

Deep Learning Flow

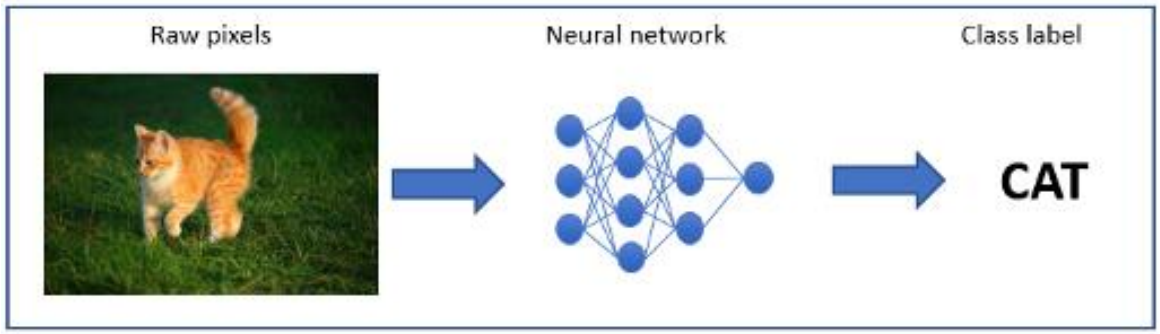
Coming up with features is difficult, time-consuming, requires expert knowledge."

Applied machine learning" is basically feature engineering. ***- Andrew Ng***

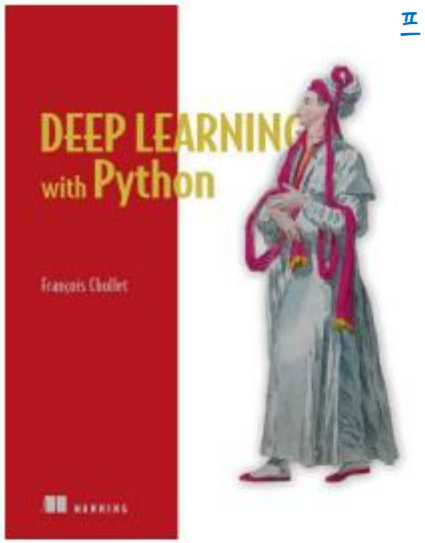
Traditional Computer Vison



Deep learning



Source: https://blog.esciencecenter.nl/mcfly-time-series-classification-made-easy-e47de8d29838

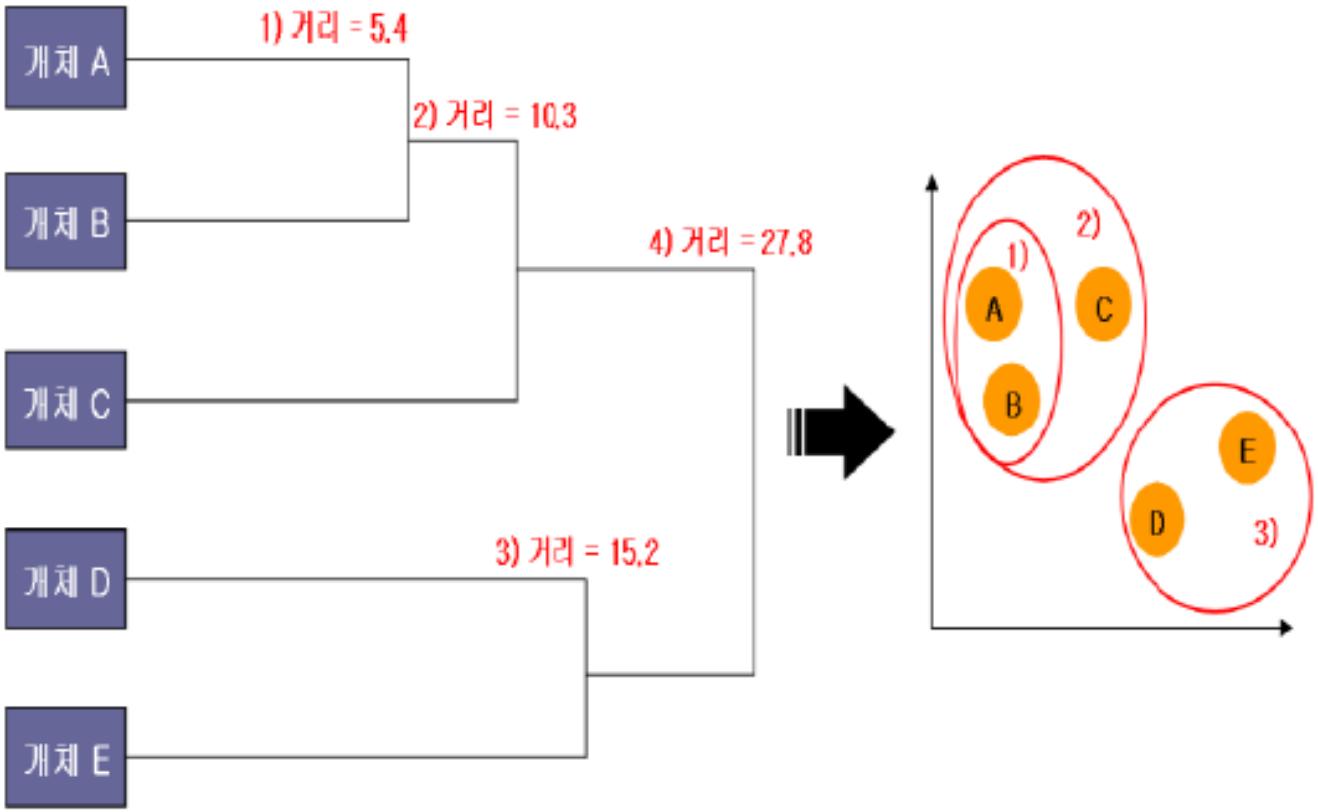


프랑스와 숄레 著

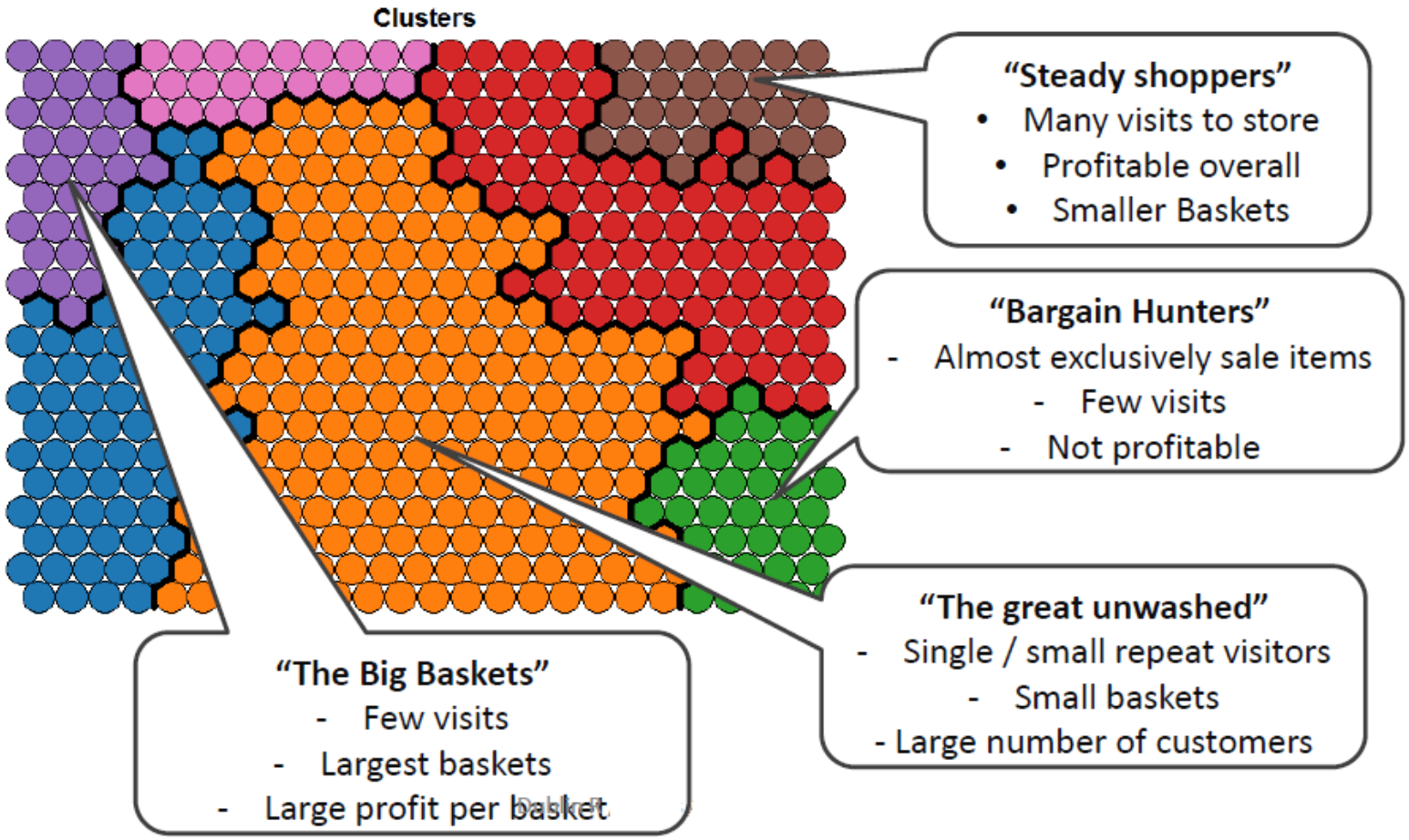
In 2016 and 2017, Kaggle was dominated by two approaches: gradient boosting machines and deep learning. Specifically, gradient boosting is used for problems where structured data is available, whereas deep learning is used for perceptual problems such as image classification. Practitioners of the former almost always use the excellent XGBoost library, which offers support for the two most popular languages of data science: Python and R. Meanwhile, most of the Kaggle entrants using deep learning use the Keras library, due to its ease of use, flexibility, and support of Python.

These are the two techniques you should be the most familiar with in order to be successful in applied machine learning today: gradient boosting machines, for shallow-learning problems; and deep learning, for perceptual problems. In technical terms, this means you'll need to be familiar with XGBoost and Keras—the two libraries that currently dominate Kaggle competitions. With this book in hand, you're already one big step closer.

군집화(Clustering)



- SOM을 이용한 식료품점 고객 세분화 사례



Source: Ta-Feng Grocery Shopping Analysis by Shane Lynn

연관규칙탐사(Association Rule Learning)

- 정의
  - 데이터 안에 존재하는 항목간의 종속 관계를 찾아내는 작업
- 장바구니 분석(market basket analysis)
  - 고객의 장바구니에 들어있는 품목 간의 관계를 발견
- 규칙의 표현
  - 항목 A와 품목 B를 구매한 고객은 품목 C를 구매한다.
  - (품목 A) & (품목 B) ⇒ (품목 C)
- 연관규칙의 활용
  - 제품이나 서비스의 교차판매
  - 매장진열, 첨부우편
  - 사기적발

ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}
…	…

market basket transactions

{Diapers, Beer} Example of a frequent itemset

{Diapers} → {Beer} Example of an association rule

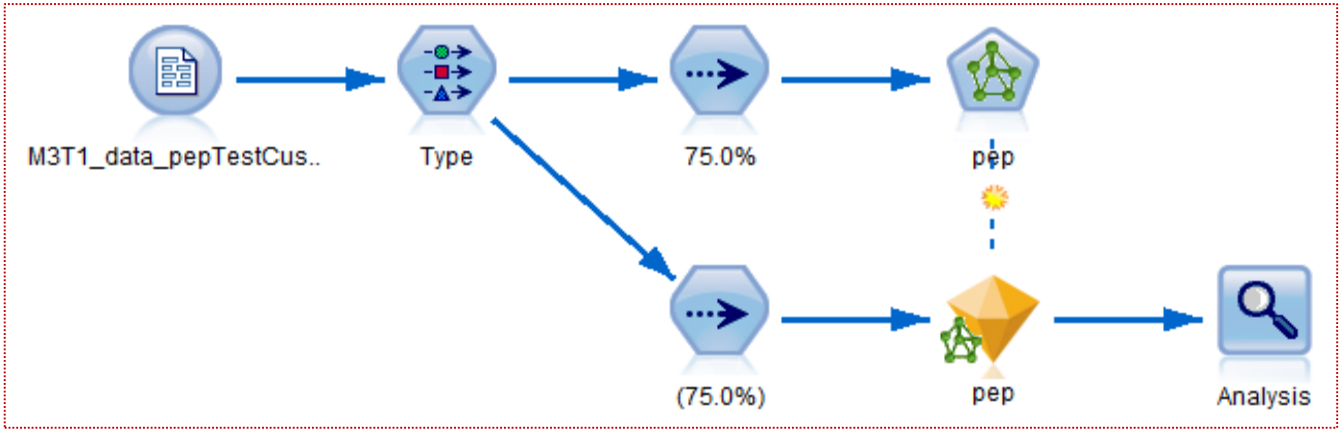
Source: http://www.big-data.tips/association-rules

Machine Learning Tools

구분	SAS Enterprise Miner	IBM SPSS Modeler	R ecosystem	Python Ecosystem
특징	사용이 간편한 GUI를 통해 모델 구축의 가속화가 가능하다. (비주얼 프로그래밍 기반)	Data 핸들링에 강하고 사용하기 쉬운 사용자 인터페이스를 가지고 있다. (비주얼 프로그래밍 기반)	통계학적 요소가 잘 스며들어있는 오픈 소스 데이터 프로그래밍 도구 (함수형 언어 기반)	문법이 간결하고 직관적인 오픈 소스 다목적 프로그래밍 도구 (객체지향 언어 기반)
장점	대용량 데이터분석이 가능하고 활용영역이 다양하다.	초보자가 배우기 쉽다. 체계적인 분석 프로세스를 지원한다.	코딩을 통해 효율적으로 분석할 수 있으며, 활용 가능한 패키지(특히 통계와 시각화)가 많다.	코딩을 통해 효율적으로 분석할 수 있으며, 딥러닝 등 최신의 머신러닝 라이브러리를 사용할 수 있다.
단점	사용법을 습득하는데 다소 시간이 걸린다. 데이터 처리의 유연성이 떨어지고 분석 알고리즘의수가 제한적이다.	분석을 위한 설정과 연결과정에 대한 프로세스가 다소 많은 편이다. 제공되는 데이터 처리와 분석 기능이 제한적이다.	배우는데 상당히 많은 시간이 소요되며, 딥러닝 지원이 부족하고 속도가 매우 느리다.	R 보다는 코딩이 쉽지만 제공되는 패키지가 상대적으로 적고, 시각화 기능이 약하며 속도가 느리다.
비용	매우 고가	고가	무료	무료

Machine Learning Tools

IBM SPSS Modeler



Python ecosystem

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
df = pd.read_csv('M3T1_data_pepTestCustomers.csv')
dfX = mdf.drop(['pep'], axis=1)
dfy = mdf['pep']
X_train, X_test, y_train, y_test = train_test_split(dfX, dfy, test_size=0.25, random_state=0)
tree = MLPClassifier(random_state=0)
tree.fit(X_train, y_train)
tree.score(X_test, y_test)
confusion_matrix(y_test, tree.predict(X_test))
```

Machine Learning Tools

R ecosystem

```
library(caret)
library(nnet)
df = read.csv('M3T1_data_pepTestCustomers.csv')
df$pep = factor(df$pep)
set.seed(0)
inTrain = createDataPartition(y=df$pep, p=0.75, list=FALSE)
train = df[inTrain,]
test = df[-inTrain,]
set.seed(0)
nn_model = nnet(pep ~ ., data=train)
confusionMatrix(predict(nn_model, newdata=test, type="class"), test$pep)
```

Python ecosystem

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
df = pd.read_csv('M3T1_data_pepTestCustomers.csv')
dfX = mdf.drop(['pep'], axis=1)
dfy = mdf['pep']
X_train, X_test, y_train, y_test = train_test_split(dfX, dfy, test_size=0.25, random_state=0)
tree = MLPClassifier(random_state=0)
tree.fit(X_train, y_train)
tree.score(X_test, y_test)
confusion_matrix(y_test, tree.predict(X_test))
```

머신러닝의 한계

- 과적합(over-fitting) 또는 과도한 일반화(Overgeneralization) 문제
- 활용할 수 있는 정확한 데이터가 충분하지 않을 경우
- 샘플링 잡음 및 편향(Sampling Noise or Bias) 문제
- 낮은 품질의 데이터
- 모델 성능 약화(model performance deterioration)

  Ex) 과거 현상들이 더 이상 미래에 유사하게 일어나지 않을 때 발생
- 영역 복잡성(Domain Complexity)

  Ex) 과거 학습된 내용(금융분야)을 다른 영역(법률분야)에 적용하지 못함
- 미래 데이터 유출(Data Leakage from the Future)

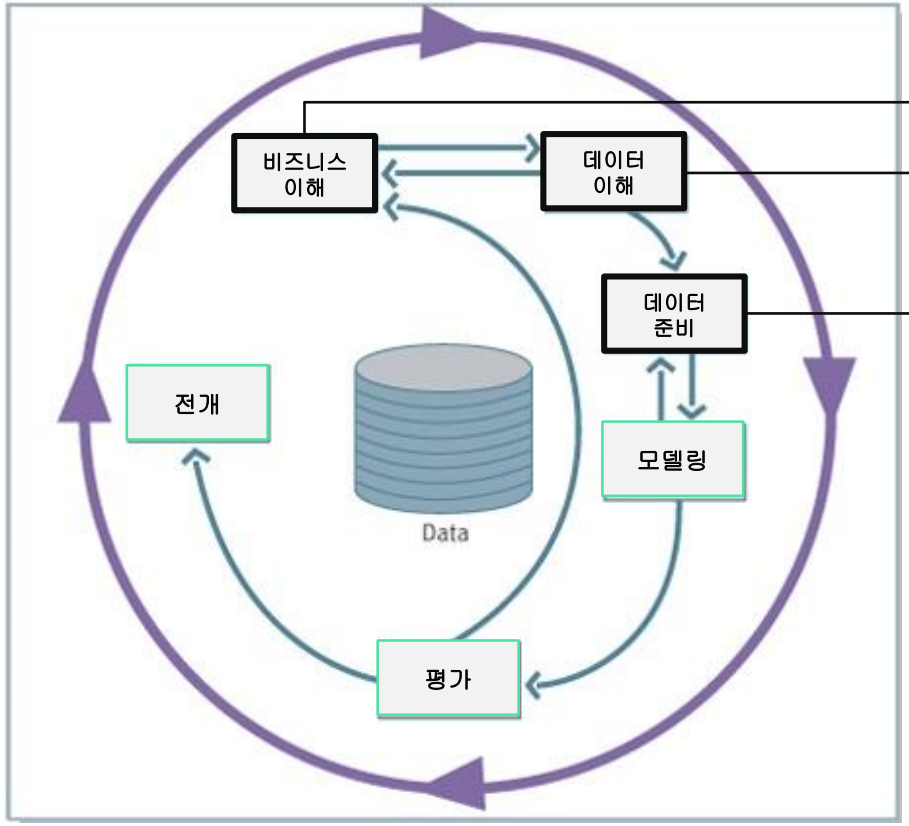
  Ex) 부동산 가격을 예측할 때 매입 이후 지불하는 인지세 및 부동산 수수료와 같은 데이터를 입력 데이터에 포함시키는 경우

End-to-End Machine Learning Project

General Machine Learning Process: CRISP-DM

CRISP-DM : SPSS에서 제시하는 프로세스 (1/2)

CRISP-DM(cross-industry standard process for data mining)은 머신러닝에 관련된 광범위한 업무의 범위를 다루고 있음.



단계 1: 비즈니스 이해(Business Understanding)

- 각종 참고 자료와 현업 책임자와의 커뮤니케이션을 통해 해당 비즈니스를 이해하는 단계

단계 2: 데이터 이해(Data Understanding)

- 레코드 수, 변수종류, 자료의 질 등, 현업이 보유 관리하고 있는 데이터를 이해하는 단계

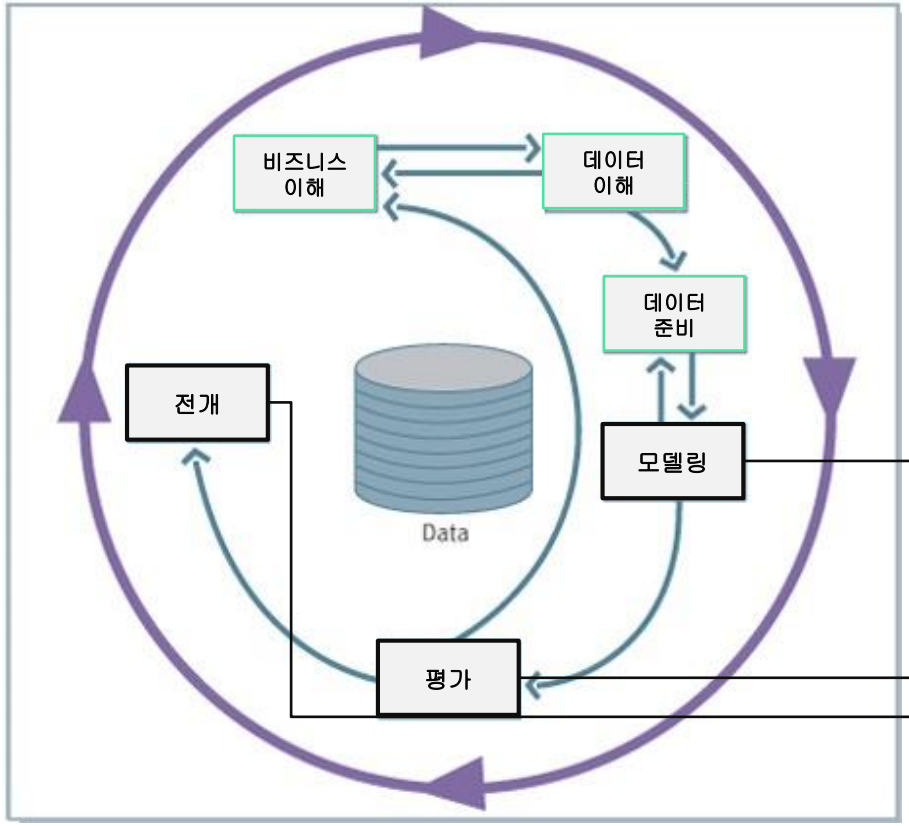
단계 3: 데이터 준비(Data Preparation)

- 데이터의 정제, 새로운 데이터 생성 등, 자료를 분석 가능한 상태로 만드는 단계

General Machine Learning Process: CRISP-DM

CRISP-DM : SPSS에서 제시하는 프로세스 (2/2)

CRISP-DM(cross-industry standard process for data mining)은 머신러닝에 관련된 광범위한 업무의 범위를 다루고 있음.



단계 4: 모델링 (Modeling)

- 자료 기술 및 탐색을 포함하여 필요한 각종 모델링을 하는 단계

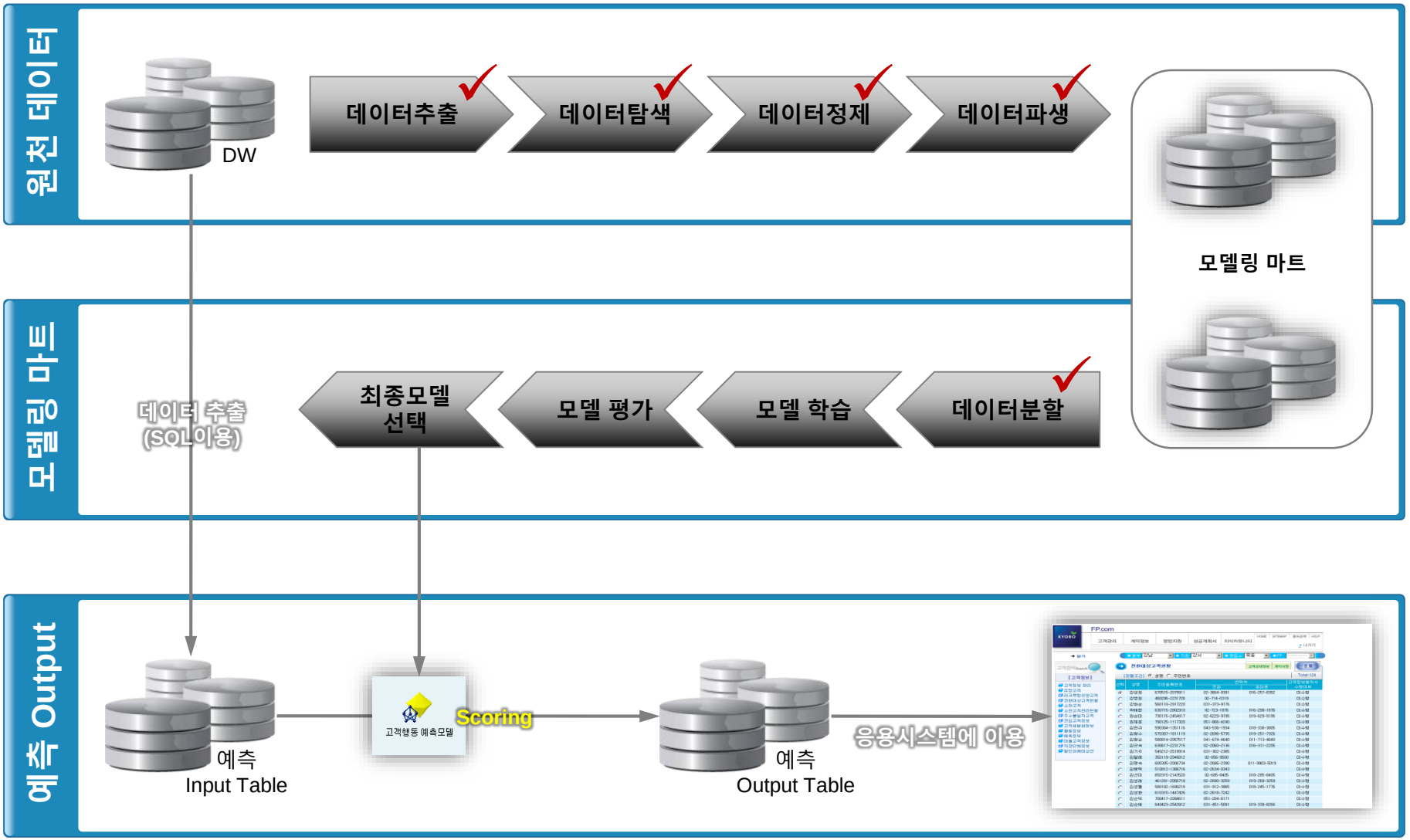
단계 5: 평가 (Evaluation)

- 모형의 해석 가능 여부, 독립적인 새 자료에 적용되는 경우에도 재현가능한지를 검토하는 단계

단계 6: 전개 (Deployment)

- 각 관리자에게 전달하여 필요한 조치를 취하는 등 검토가 끝난 모형을 실제 현업에 적용하는 단계

Predictive Analytics Process



Step 1: Business Understanding

- 사업 목적
  - 새로운 개인연금상품(PEP: Personal Equity Plan)을 개발하여 기존 고객들을 대상으로 가능한 많은 계좌를 유치

- 분석 목표
  - PEP 가입 예측모형 개발
  - 고객 프로파일 개발
  - 다이렉트 메일 광고 효율성 제고
  - 타겟 메일링에 의한 응답률 제고

Step 2: Data Understanding

- 데이터 획득 절차
  1) 기존고객 DB로부터 시험메일 발송을 위한 표본고객목록을 추출
  2) 새로운 금융상품(PEP)의 제안 메일을 발송
  3) 고객의 반응을 기록

- 분석 데이터
  - 학습용 데이터 600건 (M3T1_data_pepTestCustomers.csv)
  - 신규고객 데이터 200건 (M3T1_data_pepNewCustomers.csv)

Step 2: Data Understanding

- Frequently used libraries in Module 3 & 4

```
import pandas as pd
from pandas import Series, DataFrame
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

- Collect Initial Data

```
# Change working directory
%cd c:/+module3
```

```
# for modeling
df = pd.read_csv("M3T1_data_pepTestCustomers.csv")
# for deployment
new = pd.read_csv("M3T1_data_pepNewCustomers.csv")
```

Step 2: Data Understanding

- Describe Data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 12 columns):
id              600 non-null object
age             600 non-null int64
sex             600 non-null int64
region          600 non-null int64
income          600 non-null float64
married         600 non-null int64
children        600 non-null int64
car             600 non-null int64
save_act        600 non-null int64
current_act     600 non-null int64
mortgage        600 non-null int64
pep             600 non-null int64
dtypes: float64(1), int64(10), object(1)
memory usage: 56.3+ KB
```

Step 2: Data Understanding

```
df.head(10)
```

class label

	id	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep
0	ID12101	48	0	0	17546.00	0	1	0	0	0	0	1
1	ID12102	40	1	3	30085.10	1	3	1	0	1	1	0
2	ID12103	51	0	0	16575.40	1	0	1	1	1	0	0
3	ID12104	23	0	3	20375.40	1	3	0	0	1	0	0
4	ID12105	57	0	1	50576.30	1	0	0	1	0	0	0
5	ID12106	57	0	3	37869.60	1	2	0	1	1	0	1
6	ID12107	22	1	1	8877.07	0	0	0	0	1	0	1
7	ID12108	58	1	3	24946.60	1	0	1	1	1	0	0
8	ID12109	37	0	2	25304.30	1	2	1	0	0	0	0
9	ID12110	54	1	3	24212.10	1	2	1	1	1	0	0

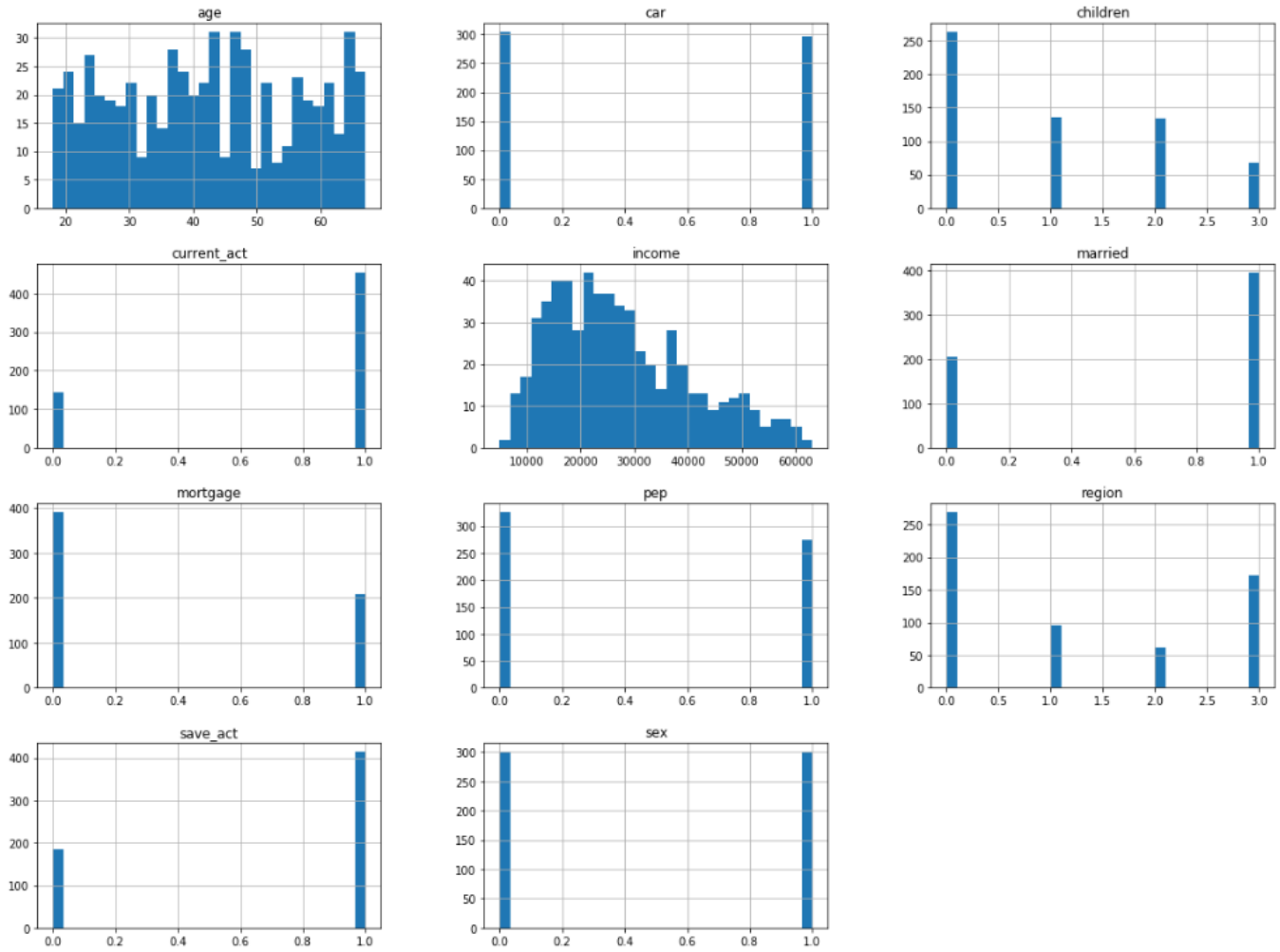
Step 2: Data Understanding

```
df.describe()
```

	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep
count	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000	600.000000
mean	42.395000	0.500000	1.231667	27524.031217	0.660000	1.011667	0.493333	0.690000	0.758333	0.348333	0.456667
std	14.424947	0.500417	1.286113	12899.468246	0.474104	1.056752	0.500373	0.462879	0.428451	0.476840	0.498534
min	18.000000	0.000000	0.000000	5014.210000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	30.000000	0.000000	0.000000	17264.500000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
50%	42.000000	0.500000	1.000000	24925.300000	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000	0.000000
75%	55.250000	1.000000	3.000000	36172.675000	1.000000	2.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	67.000000	1.000000	3.000000	63130.100000	1.000000	3.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Step 2: Data Understanding

```
df.hist(bins=30, figsize=(20,15))
```



Step 2: Data Understanding

```
print(new.shape)
new.tail()
```

```
(200, 11)
```

	id	age	sex	region	income	married	children	car	save_act	current_act	mortgage
195	ID12896	66	1	0	58792.6	0	1	1	1	1	1
196	ID12897	19	1	0	17906.8	0	2	1	1	0	0
197	ID12898	54	1	1	29348.8	0	0	0	0	1	1
198	ID12899	42	0	3	20552.5	1	0	1	1	1	0
199	ID12900	34	1	0	25843.1	0	2	0	1	1	0

Step 2: Data Understanding

- Explore Data – *Look for Correlations*

```
df.corr()
```

	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep
age	1.000000	-0.090081	0.011167	0.752726	0.010394	0.023572	0.077733	0.184389	-0.035312	-0.016154	0.173825
sex	-0.090081	1.000000	-0.035018	-0.023845	0.021110	-0.014206	0.006667	0.007207	-0.019466	0.066465	0.046843
region	0.011167	-0.035018	1.000000	-0.000212	0.006188	0.011520	0.021860	0.084382	-0.013356	0.026083	-0.027279
income	0.752726	-0.023845	-0.000212	1.000000	-0.008386	0.036761	0.081556	0.266164	0.031616	-0.014662	0.221991
married	0.010394	0.021110	0.006188	-0.008386	1.000000	-0.048716	-0.009571	0.028604	-0.059996	-0.021711	-0.189578
children	0.023572	-0.014206	0.011520	0.036761	-0.048716	1.000000	0.036455	0.041536	0.006238	-0.074339	-0.057663
car	0.077733	0.006667	0.021860	0.081556	-0.009571	0.036455	1.000000	0.034310	-0.034783	-0.007743	0.018917
save_act	0.184389	0.007207	0.084382	0.266164	0.028604	0.041536	0.034310	1.000000	0.042511	-0.001588	-0.072779
current_act	-0.035312	-0.019466	-0.013356	0.031616	-0.059996	0.006238	-0.034783	0.042511	1.000000	-0.036704	0.025141
mortgage	-0.016154	0.066465	0.026083	-0.014662	-0.021711	-0.074339	-0.007743	-0.001588	-0.036704	1.000000	-0.024182
pep	0.173825	0.046843	-0.027279	0.221991	-0.189578	-0.057663	0.018917	-0.072779	0.025141	-0.024182	1.000000

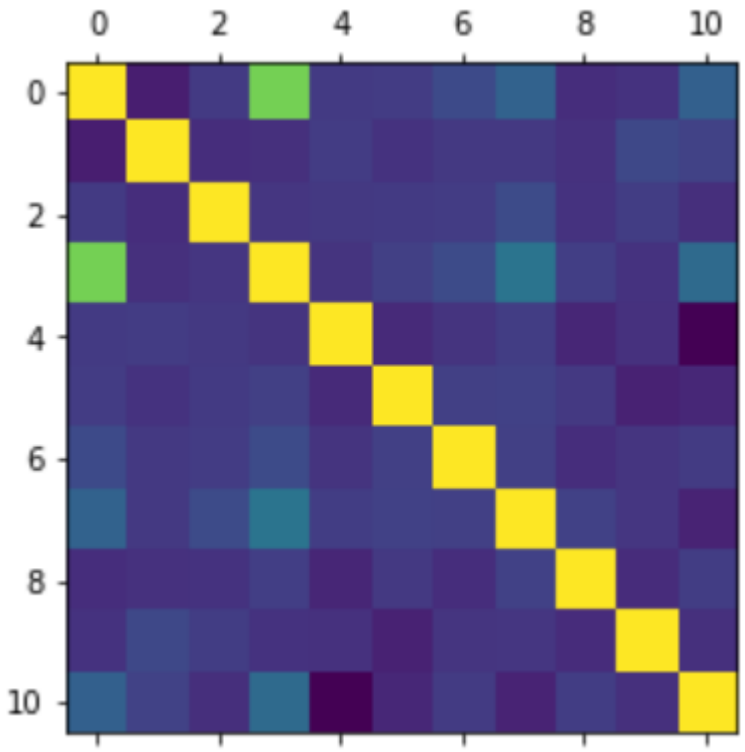
Step 2: Data Understanding

```
df.corr().pep.sort_values(ascending=False)
```

```
pep             1.000000
income          0.221991
age             0.173825
sex             0.046843
current_act     0.025141
car             0.018917
mortgage       -0.024182
region         -0.027279
children       -0.057663
save_act       -0.072779
married        -0.189578
Name: pep, dtype: float64
```

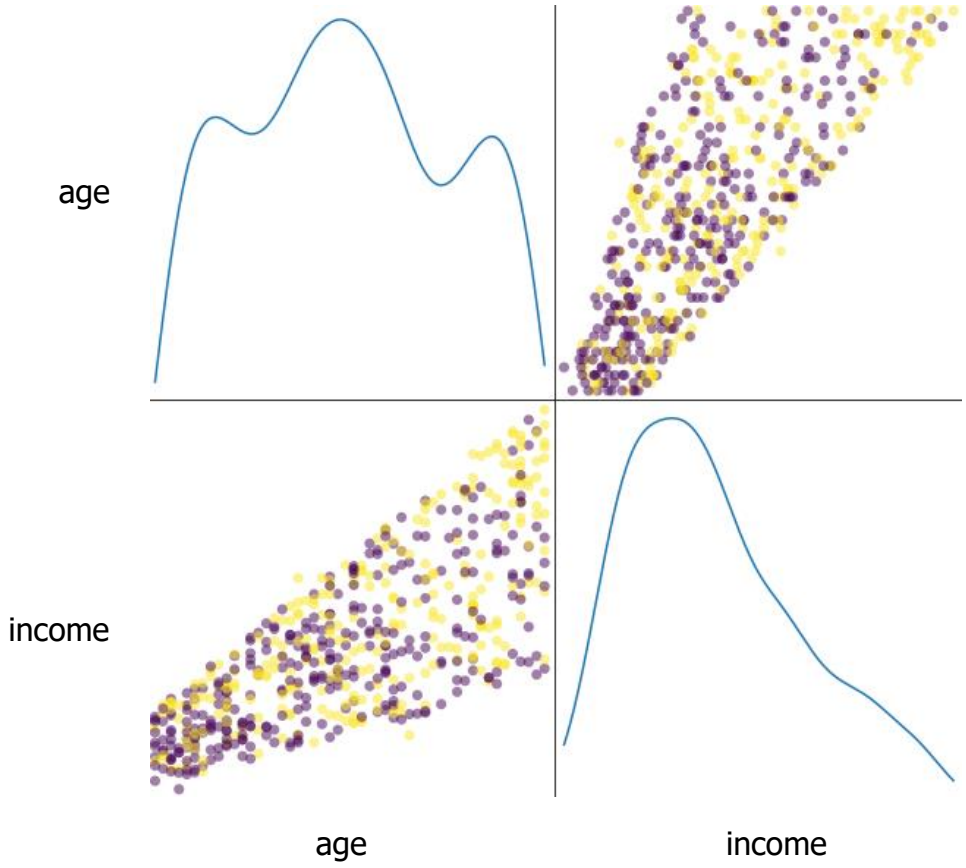
Step 2: Data Understanding

```
plt.matshow(df.corr())
```



Step 2: Data Understanding

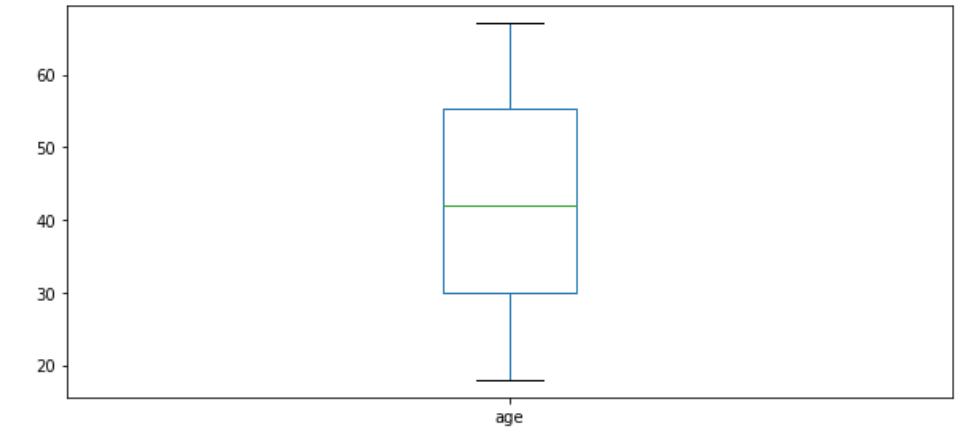
```
from pandas.plotting import scatter_matrix
scatter_matrix(df.iloc[:,[1,4]], c=df['pep'], figsize=(10,10), marker='o', s=50, diagonal='kde')
```

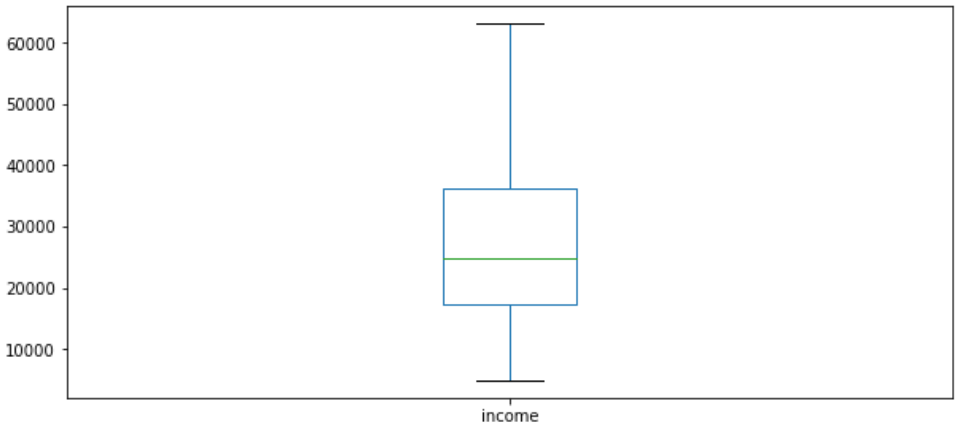


Step 2: Data Understanding

- Explore Data – *Detect Outliers*

```
df.loc[:,['age','income']].plot.box(subplots=True, layout=(2,1), figsize=(10,10))
```





Step 3: Data Preparation

- Construct Data – *Derive Attributes*

```
mdf = df.copy()
mdf['realincome'] = np.where(mdf['children']==0, mdf['income'], mdf['income']/mdf['children'])
mdf.head()
```

	id	age	sex	region	income	married	children	car	save_act	current_act	mortgage	pep	realincome
0	ID12101	48	0	0	17546.0	0	1	0	0	0	0	1	17546.000000
1	ID12102	40	1	3	30085.1	1	3	1	0	1	1	0	10028.366667
2	ID12103	51	0	0	16575.4	1	0	1	1	1	0	0	16575.400000
3	ID12104	23	0	3	20375.4	1	3	0	0	1	0	0	6791.800000
4	ID12105	57	0	1	50576.3	1	0	0	1	0	0	0	50576.300000

derived attribute

Step 3: Data Preparation

- Select Data – *Filter Attributes*

```
mdf = mdf.drop(['income','children'], axis=1)
mdf.head()
```

	id	age	sex	region	married	car	save_act	current_act	mortgage	pep	realincome
0	ID12101	48	0	0	0	0	0	0	0	1	17546.000000
1	ID12102	40	1	3	1	1	0	1	1	0	10028.366667
2	ID12103	51	0	0	1	1	1	1	0	0	16575.400000
3	ID12104	23	0	3	1	0	0	1	0	0	6791.800000
4	ID12105	57	0	1	1	0	1	0	0	0	50576.300000

Step 3: Data Preparation

- Split Data

```
from sklearn.model_selection import train_test_split  # for Hold-out validation
```

```
dfX = mdf.drop(['id','pep'], axis=1)  # exclude 'id' attribute & class variable
dfy = mdf['pep']                      # class variable
X_train, X_test, y_train, y_test = train_test_split(dfX, dfy, test_size=0.25, random_state=0)
```

```
print(X_train.shape, X_test.shape)
```

```
(450, 9) (150, 9)
```

```
X_train.head()
```

	age	sex	region	married	car	save_act	current_act	mortgage	realincome
46	50	0	0	0	1	1	1	0	13283.9
263	60	0	0	1	1	1	1	1	46358.4
458	18	1	2	0	0	1	1	0	13700.2
230	59	0	0	1	1	0	1	1	30189.4
107	23	1	0	1	0	0	1	0	13039.9

Step 4: Modeling

- Build Model – *Decision Tree*

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree = DecisionTreeClassifier(max_depth=6, random_state=0)
```

```
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=6,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=0, splitter='best')
```

```
pred_tree = tree.predict(X_test); pred_tree
```

```
array([0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0], dtype=int64)
```

Step 4: Modeling

- Assess Model

```
tree.score(X_train, y_train)
```

```
0.83555555555555561
```

```
tree.score(X_test, y_test)
```

```
0.67333333333333334
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, pred_tree)
```

```
array([[64, 14],
       [35, 37]])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred_tree, target_names=['not buy','buy']))
```

```
             precision    recall  f1-score   support

    not buy       0.65      0.82      0.72        78
        buy       0.73      0.51      0.60        72

avg / total       0.68      0.67      0.66       150
```

Step 5: Evaluation

- Which model is the best ?
- Is the model useful ?

```
best_model = tree    # Change this code if the best model is not decision tree.
best_model.score(X_test, y_test)
```

```
0.67333333333333334
```

```
from sklearn.dummy import DummyClassifier
print(y_test.value_counts())
DummyClassifier(strategy='most_frequent').fit(X_train, y_train).score(X_test, y_test)
```

```
0    78
1    72
Name: pep, dtype: int64

0.52000000000000002
```

Step 6: Deployment

- Preprocessing

```
# You must do the same preprocessing as the modeling data.
ndf = new.copy()
ndf['realincome'] = np.where(ndf['children']==0, ndf['income'], ndf['income']/ndf['children'])
ndf = ndf.drop(['income','children'], axis=1)
ndf.head()
```

	id	age	sex	region	married	car	save_act	current_act	mortgage	realincome
0	ID12701	23	1	0	1	1	1	0	1	18766.90
1	ID12702	30	1	1	0	0	1	0	1	9915.67
2	ID12703	45	0	1	0	1	1	1	0	21881.60
3	ID12704	50	1	3	1	0	1	0	1	23397.20
4	ID12705	41	0	0	1	1	1	1	0	20721.10

Step 6: Deployment

- Case I – Apply the best model to select target customers

```
ndf['pred'] = best_model.predict(ndf.loc[:,'age':'realincome'])
```

```
print(best_model.predict_proba(ndf.loc[:,'age':'realincome']))
ndf['pred_prob'] = best_model.predict_proba(ndf.loc[:,'age':'realincome'])[:,1]
```

```
[[ 0.16666667  0.83333333]
 [ 1.          0.        ]
 [ 0.          1.        ]
 [ 0.625       0.375     ]
 [ 0.5         0.5       ]
 [ 0.61111111  0.38888889]
 [ 0.85714286  0.14285714]
 [ 0.15        0.85      ]
 [ 1.          0.        ]
 [ 0.5         0.5       ]
 [ 0.85714286  0.14285714]
 [ 0.          1.        ]
 [ 0.5         0.5       ]
 [ 1.          0.        ]
 [ 0.58823529  0.41176471]
```

Step 6: Deployment

```
ndf.head()
```

예측 값과 확률

	id	age	sex	region	married	car	save_act	current_act	mortgage	realincome	pred	pred_prob
0	ID12701	23	1	0	1	1	1	0	1	18766.90	1	0.833333
1	ID12702	30	1	1	0	0	1	0	1	9915.67	0	0.000000
2	ID12703	45	0	1	0	1	1	1	0	21881.60	1	1.000000
3	ID12704	50	1	3	1	0	1	0	1	23397.20	0	0.375000
4	ID12705	41	0	0	1	1	1	1	0	20721.10	0	0.500000

```
target = ndf.query('pred == 1 & pred_prob > 0.7')  # PEP에 가입할 확률이 70%가 넘는 고객만 추출
target.sort_values(by="pred_prob", ascending=False).to_csv("target.csv", index=False)
pd.read_csv("target.csv").tail()
```

	id	age	sex	region	married	car	save_act	current_act	mortgage	realincome	pred	pred_prob
51	ID12860	31	0	0	1	0	0	1	1	24609.7	1	0.833333
52	ID12854	45	0	0	1	1	0	0	1	16385.4	1	0.833333
53	ID12844	40	0	0	1	0	1	1	1	23988.3	1	0.833333
54	ID12829	31	0	0	1	0	0	1	1	23025.4	1	0.833333
55	ID12701	23	1	0	1	1	1	0	1	18766.9	1	0.833333

Step 6: Deployment

- Case II – Export the best model for future use in other programs or systems

```
from sklearn.externals import joblib
```

```
# save the model to disk
joblib.dump(best_model, 'pep_model.sav')
```

```
['pep_model.sav']
```

```
# load the model from disk
loaded_model = joblib.load('pep_model.sav')
loaded_model.score(X_test, y_test)
```

```
0.67333333333333334
```