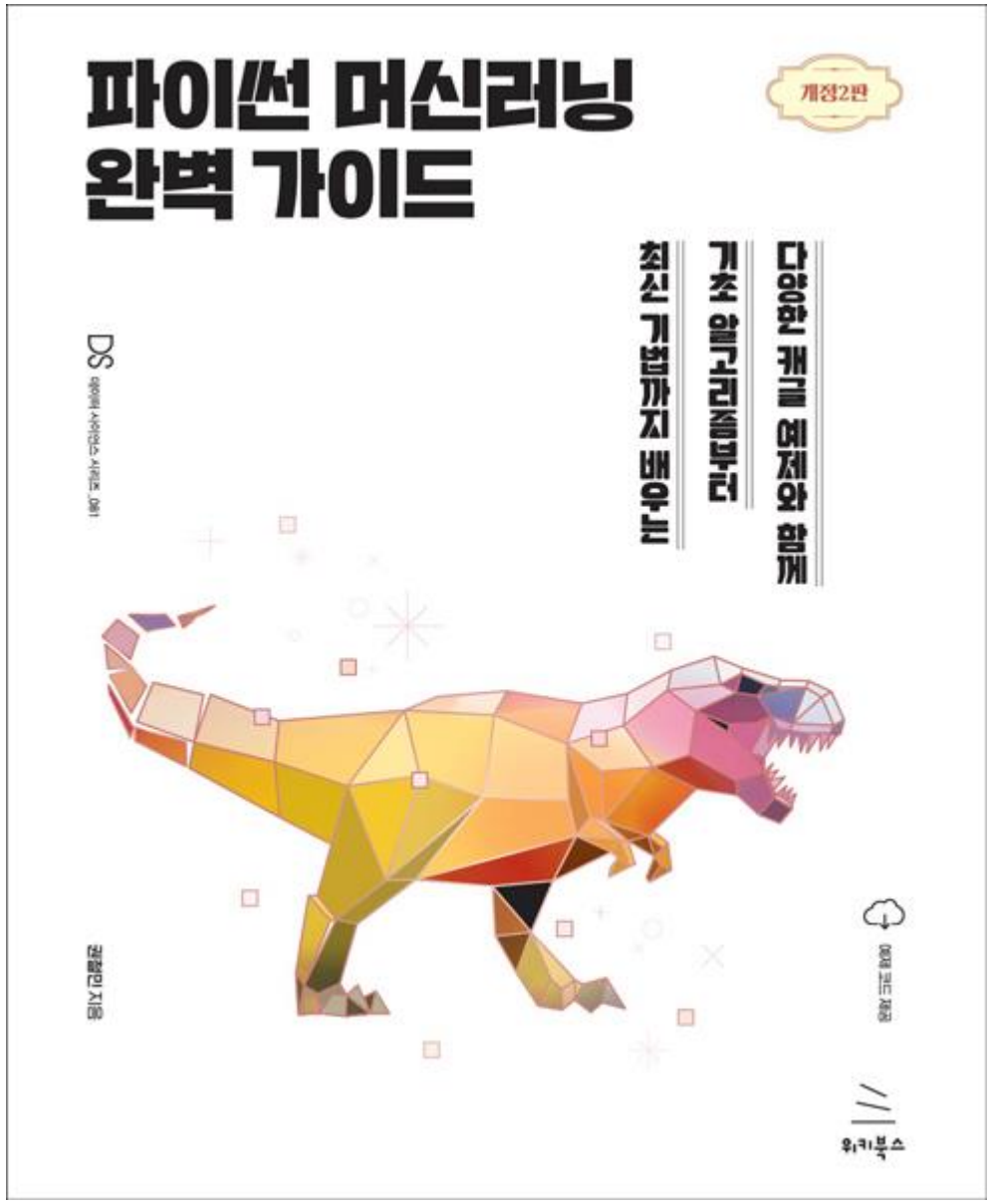
2023 Machine Learning Odyssey Pt. 1

2023.07.25.

Seungeun Lee

“2023 Machine Learning Odyssey”

Part 1	2023.07.25.	Tabular data, Data preprocessing, Evaluation metrics, Cross validation, imputation
Part 2	TBD (?)	AutoML, Feature selection/extraction, Data Imbalance, Data Preprocessing 2
Part 3	TBD	LightGBM, Hyperparameter tuning, Gridsearch
Part 4	TBD	SVM, RandomForest, Clustering, Dimension reduction
Part 5	TBD	XGBoost, Ensemble Models
Part 6	TBD	ML v.s. DL (TabNet, ⋯), Future of Machine Learning (Causal Inference, Bayesian)
Part 7	TBD	Meta Learning & Meta Reinforcement Learning
Part 8	TBD	MLOps, Multi-modal analysis

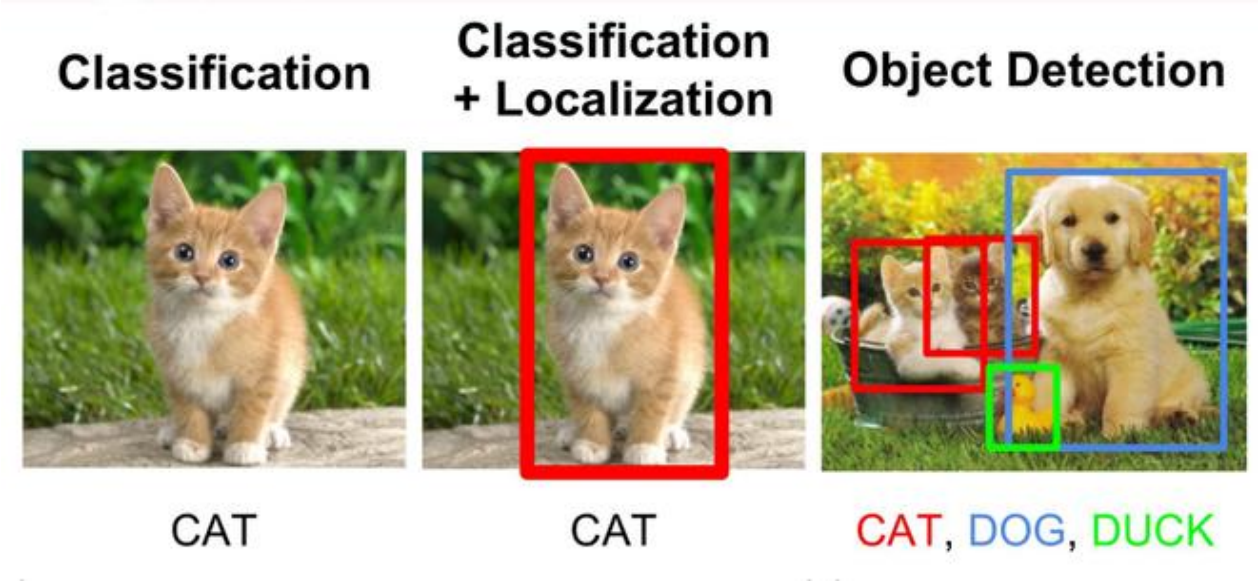


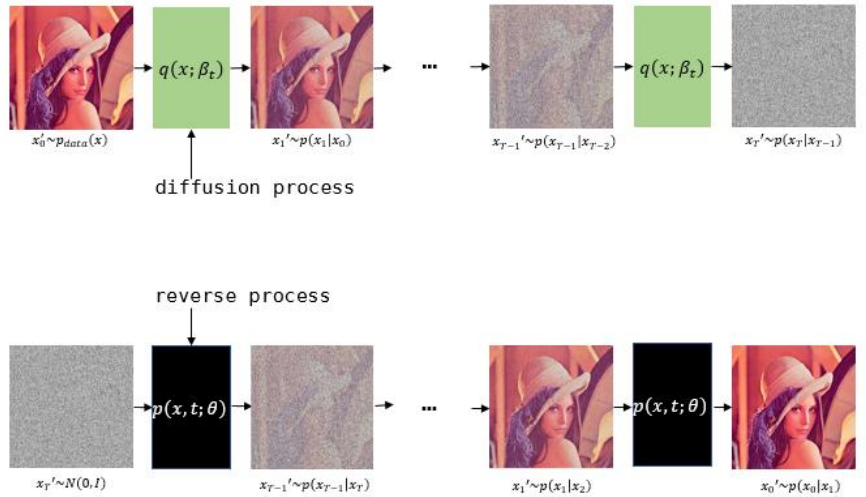
https://github.com/wikibook/pymlrev2

Research Interest

- **Computer Vision**

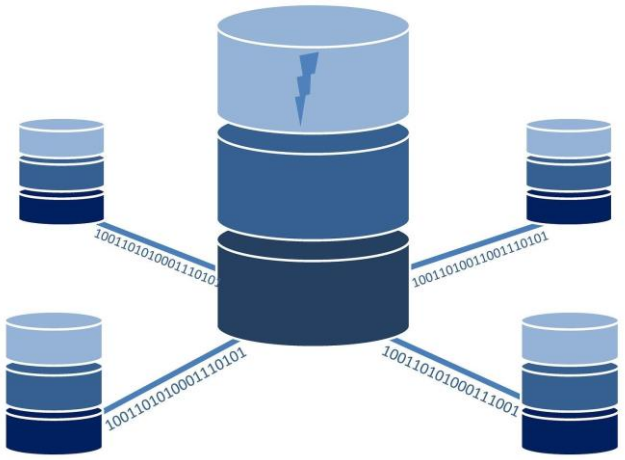
(Image Manipulation, Latent Editing, 3D, Medical Images)





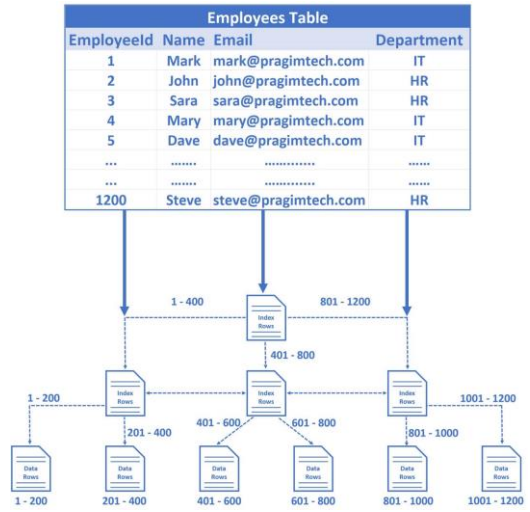
Tabular Data

- Structured Data [정형 데이터]
- 데이터베이스에서 추출한 table 형태의 행과 열로 표현되는 데이터



DataBase

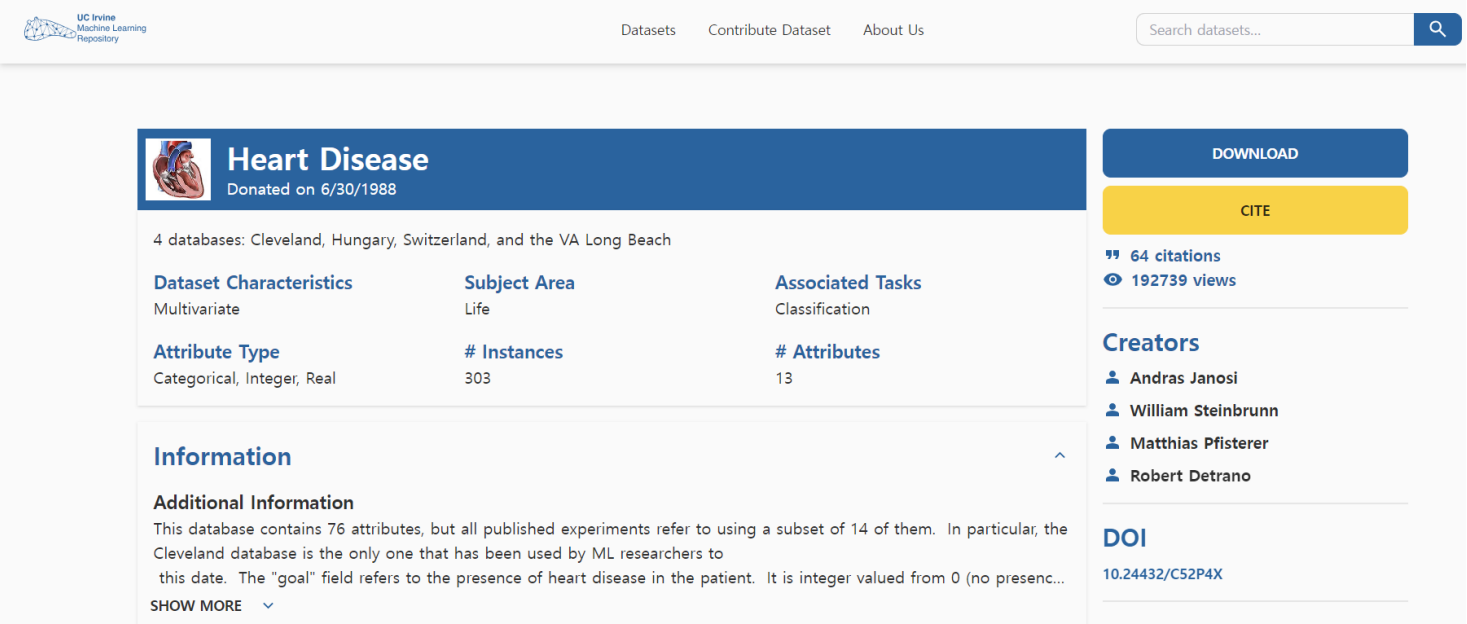
SQL Query



Data

UCI Heart Disease

- UC Irvine Machine Learning Repository
- https://archive.ics.uci.edu/dataset/45/heart+disease



UCI Heart Disease

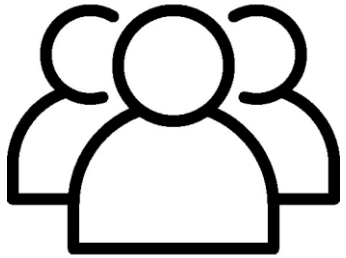
Features

Attribute Name	Role	Type	Demographic	Description	Units	Missing Values
age	Feature	Discrete			years	false
sex	Feature	Categorical				false
cp	Feature	Categorical				false
trestbps	Feature	Discrete		resting blood pressure (on admission to the hospital)	mm Hg	false
chol	Feature	Discrete		serum cholestoral	mg/dl	false
fbs	Feature	Categorical		fasting blood sugar > 120 mg/dl		false
restecg	Feature	Categorical				false
thalach	Feature	Discrete		maximum heart rate achieved		false
exang	Feature	Categorical		exercise induced angina		false
oldpeak	Feature	Discrete		ST depression induced by exercise relative to rest		false

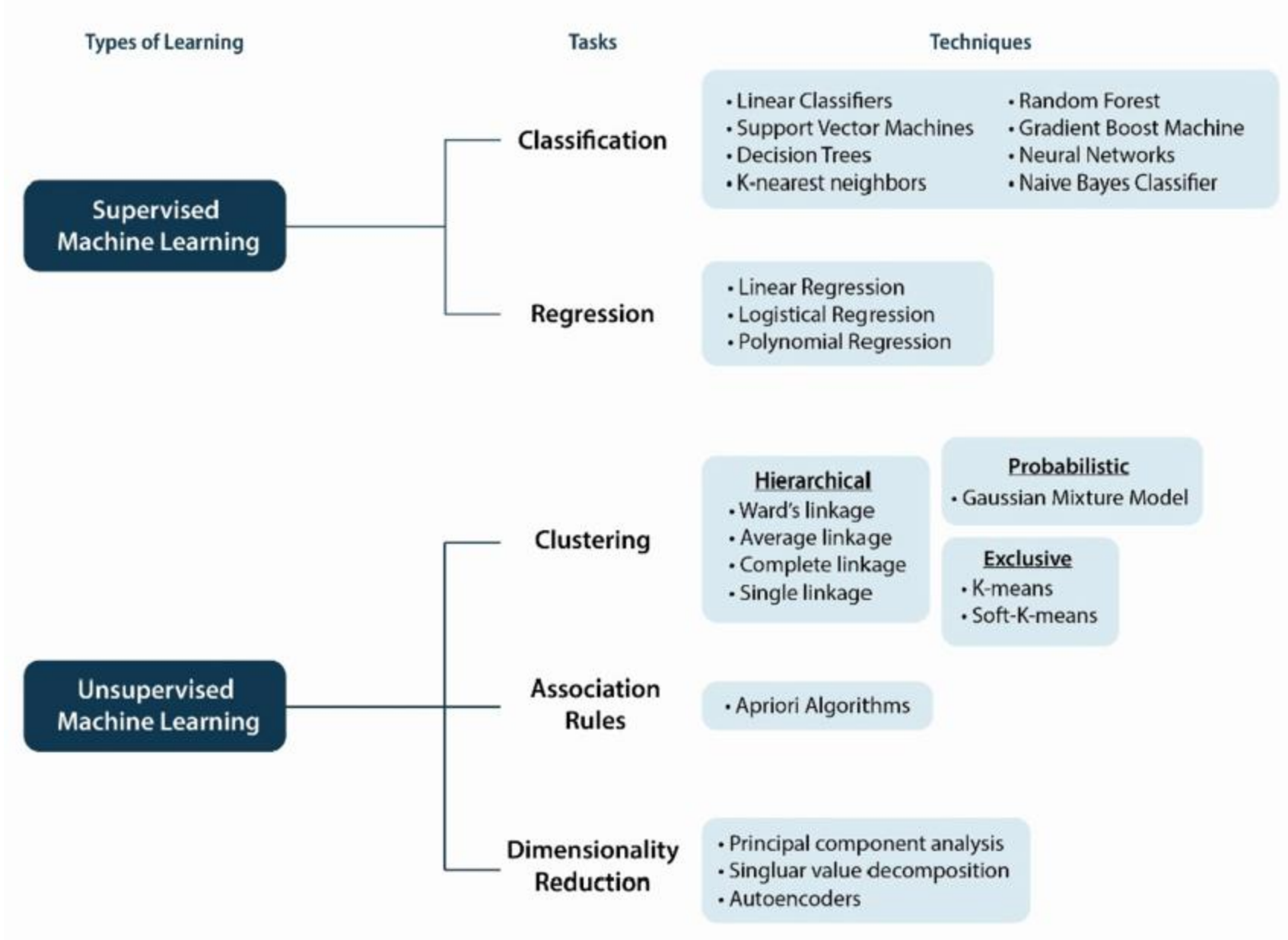
UCI Heart Disease

- https://drive.google.com/drive/folders/1OPBBz-FA4IRkEEodf9kmfwJBz7fzazPu?usp=sharing
- or 첨부한 Jupyter Notebook 파일

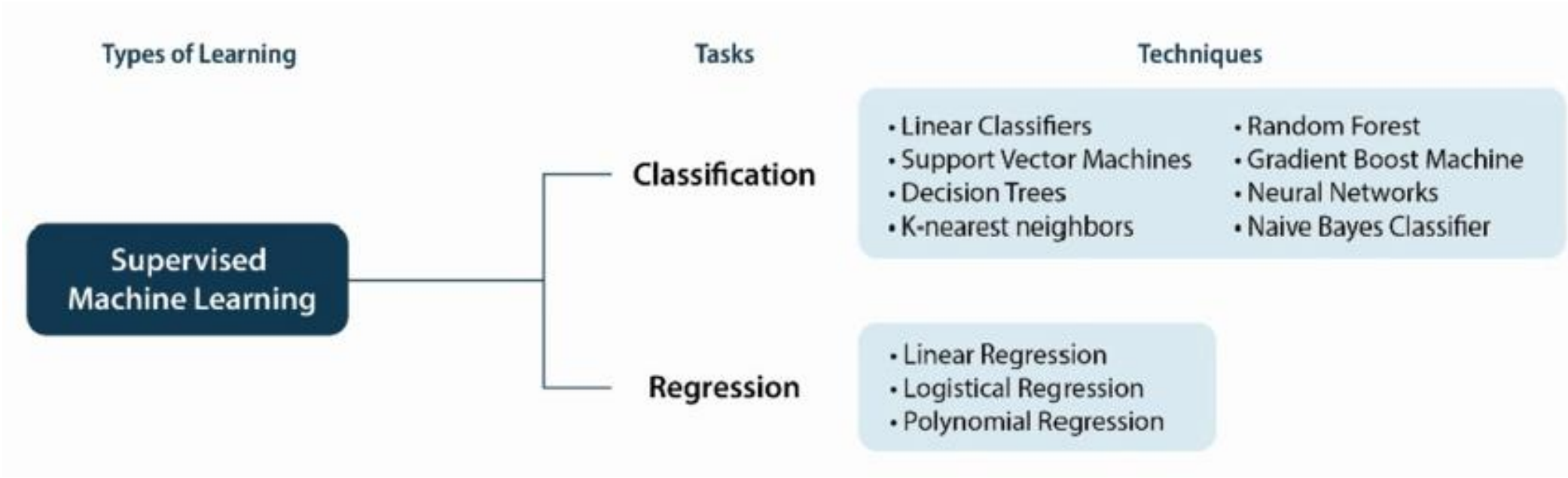
Why is Tabular Data important?

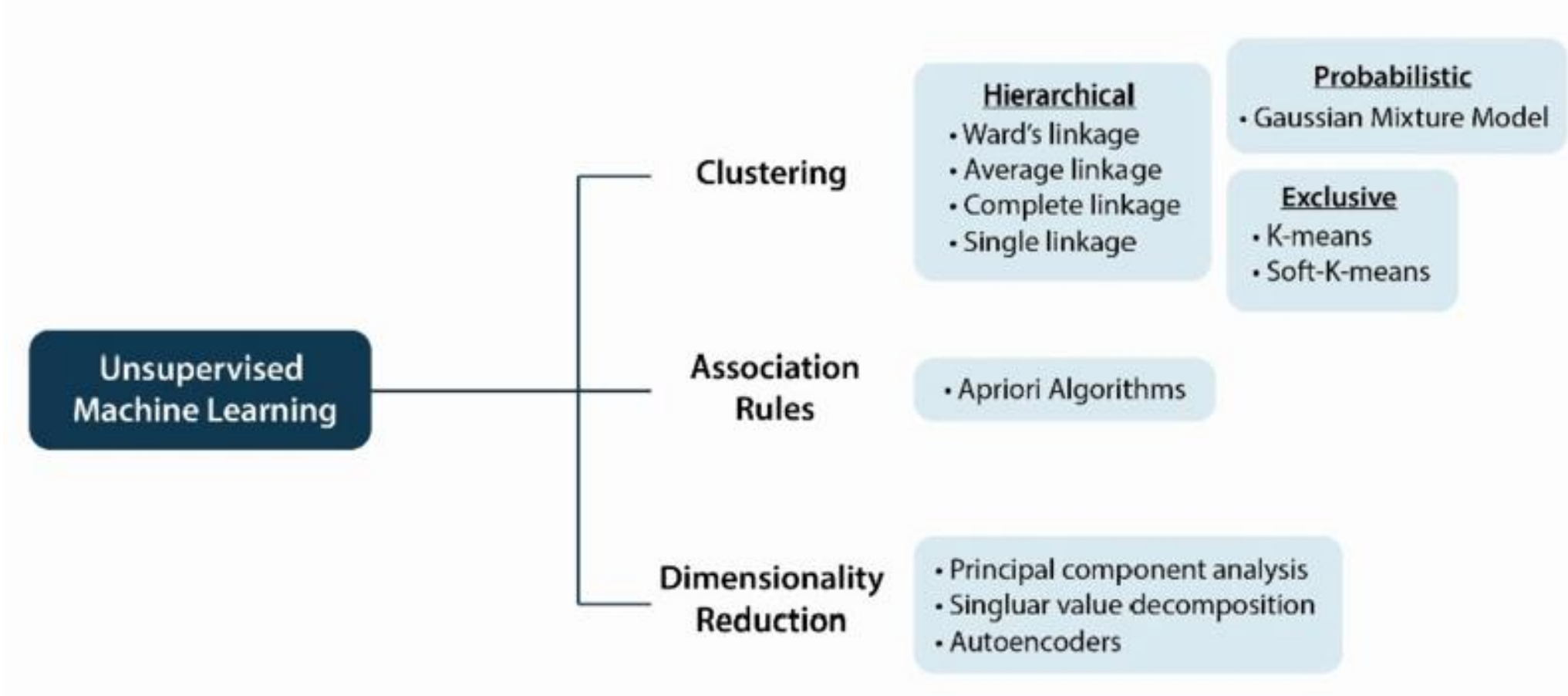


"현직에서는 비정형데이터보다 정형데이터 더 많이 사용"
"비정형데이터 (이미지, 음성, 텍스트) 구하기는
아직까지 쉽지 않아 클라우드 개선 작업 중"
"ML 도움으로 기기 파라미터 튜닝"
"ML & 통계학적 지식이 필요할 때가 많음"


Types of Learning
Tasks
Techniques
Supervised Machine Learning
Classification
• Linear Classifiers
• Support Vector Machines
• Decision Trees
• K-nearest neighbors
• Random Forest
• Gradient Boost Machine
• Neural Networks
• Naive Bayes Classifier
Regression
• Linear Regression
• Logistical Regression
• Polynomial Regression
Unsupervised Machine Learning
Clustering
Hierarchical
• Ward's linkage
• Average linkage
• Complete linkage
• Single linkage
Probabilistic
• Gaussian Mixture Model
Exclusive
• K-means
• Soft-K-means
Association Rules
• Apriori Algorithms
Dimensionality Reduction
• Principal component analysis
• Singluar value decomposition
• Autoencoders

Random Forest
XGBoost




Unsupervised Machine Learning
Clustering
Hierarchical
• Ward's linkage
• Average linkage
• Complete linkage
• Single linkage
Probabilistic
• Gaussian Mixture Model
Exclusive
• K-means
• Soft-K-means
Association Rules
• Apriori Algorithms
Dimensionality Reduction
• Principal component analysis
• Singluar value decomposition
• Autoencoders

Missing feature (NA, Not Available)

- **Missing Values [결측치] 처리** (DL에서는 결측치라는 concept이 없음. Tabular data는 부분적 결측치 발생)

(1) Do nothing: XGBoost, LightGBM 같은 결측치를 handling 해주는 모델 사용 (추후 발표)

결측치가 왜 발생했을까? (Random? 특별한 이유?)

(2) Deletion / Drop (누락된 데이터 제거): 조심! (중요한 데이터 손실)

Row no	State	Salary	Yrs of Experience
1	NY	57400	Mid
2	TX		Entry
3	NJ	90000	High
4	VT	36900	Entry
5	TX		Mid
6	CA	76600	High
7	NY	85000	High
8	CA		Entry
9	CT	45000	Entry

Missing values

Row no	State	Salary	Yrs of Experience
1	NY	57400	Mid
2	TX		Entry
3	NJ	90000	High
4	VT	36900	Entry
5	TX		Mid
6	CA	76600	High
7	NY	85000	High
8	CA		Entry
9	CT	45000	Entry

Missing values

Missing feature (NA, Not Available)

(3) Imputation

- Mean, Median imputation
- Most Frequent Value / Zero / Constant Imputation
- 결측치가 왜 발생했을까? (Random? 특별한 이유?) -> Domain Knowledge를 바탕으로 대치
- 단점: 다른 feature (column)과의 상관관계를 고려하지 않으며, bias 생길 수 있음 -> 다른 방식?

< 결측치 처리 가이드 라인 >
10% 미만: 삭제 or 대치
10% - 50%: Regression or
Model-based imputation -> *대치*
50% 이상: 해당 column (variable) 자체 제거

(절대적인 rule은 아니며, 경험적인 수치)

Missing feature (NA, Not Available)

- KNN Imputation
- MICE (Multivariate Imputation by Chained Equation) Imputation
- DL 활용

```
from impyute.imputation.cs import fast_knn
np_imputed=fast_knn(df_null.values, k=5)
df_imputed = pd.DataFrame(np_imputed)
```

```
from impyute.imputation.cs import mice
np_imputed=mice(df_null.values)
df_imputed = pd.DataFrame(np_imputed)
```

- 각각이 하나의 논문 -> 자세한 설명은 생략
- 단점: imputation 하는데 시간이 매우 오래 걸림 ($\mathbb{R}^{321\times70}$ 크기: 2시간(?))
  메모리가 많이 필요, outlier에 민감

Missing feature (NA, Not Available)

- **KNN Imputation**

  feature similarity를 이용해 가장 근접한 데이터 K개를 찾는 방식
  KDTree를 생성해 가장 가까운 이웃 (NN) 찾기 -> 거리에 따라 가중 평균 부여

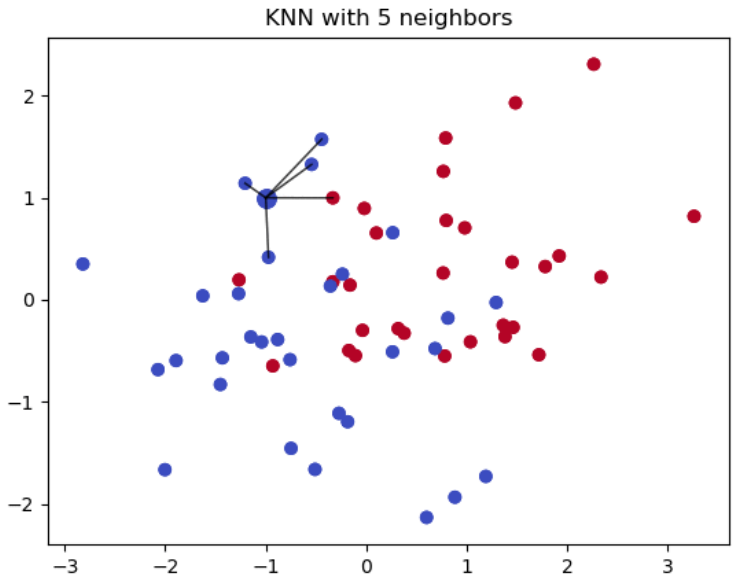
- **MICE (Multivariate Imputation by Chained Equation) Imputation**

  누락된 데이터를 여러 번 채우는 방식: Chain 형식의 접근법
  (연속형, 이진형, 범위형, survey skip 패턴도 처리 가능)
  Single Imputation 보다 Multiple Imputation이 불확실성을 잘 처리

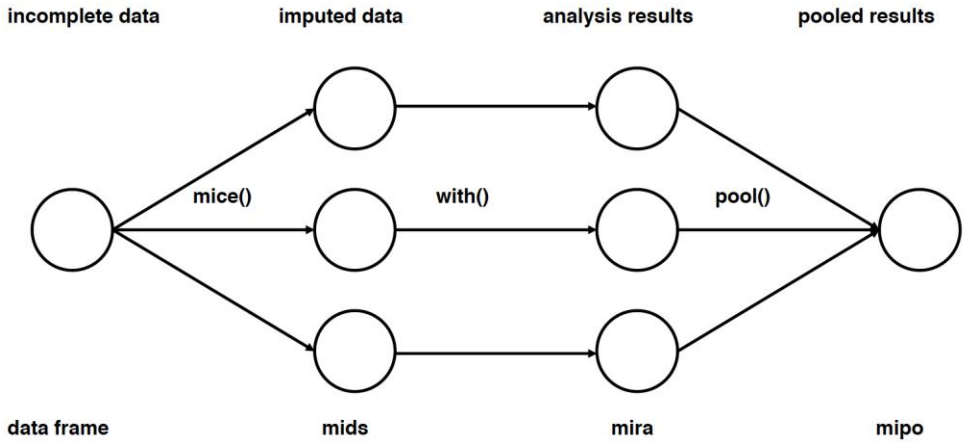
  ① Imputation: imputation 방식 선택 (option: mean, median 등); m가지 방법으로 대치
  ② Analysis: m개의 완성된 데이터셋을 분석
  ③ Pooling: 평균, 분산, 신뢰 구간을 계산하여 결과 통합

- **DL 활용** (ML 문제를 풀기 위해 DL을 활용하는 것은 overload…?)

Missing feature (NA, Not Available)



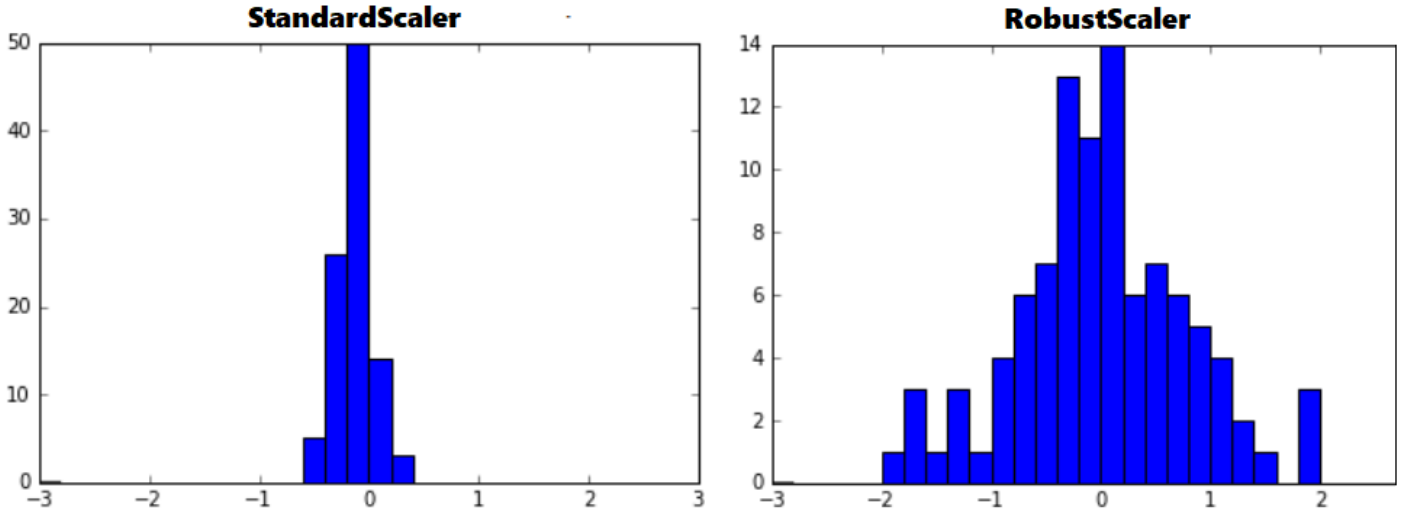
KNN Imputation



MICE Imputation

Data preprocessing (Scikit-learn)

- **StandardScaler**: 평균을 제거하고 데이터를 단위 분산으로 조정. Outlier에 민감.
- **MinMaxScaler**: 모든 feature 값이 0-1사이에 있도록 데이터를 조정. Outlier에 민감.
- **MaxAbsScaler**: 절대값을 0-1사이로 조정. (i.e. -1과 1 사이로 조정)
  양수 데이터로만 구성된 데이터셋에서는 MinMaxScaler와 유사하게 동작. 큰 outlier에 민감.
- **RobustScaler**: Outlier 영향 최소화. 중앙값 (median)과 IQR (interquartile range)을 사용하기에 StandardScaler와 비교해보면 표준화 후 동일한 값을 더 넓게 분포 시키고 있음



Data preprocessing

- 스케일링시 Feature별 크기를 유사하게 만드는 것은 중요하지만,
  그렇다고 모든 Feature의 분포를 동일하게 만들 필요는 없음.

- 특성에 따라 어떤 항목은 원본 데이터의 분포를 유지하는 것이 유의미할 수 있음.
- e.g. 데이터가 거의 한 곳에 집중되어 있는 Feature를 표준화시켜 분포를 같게 만들었을 때 작은 단위의 변화가 큰 차이를 나타내는 것으로 반영될 수 있기 때문.
- 경험상 원본 데이터를 그대로 사용한 실험이 성능이 더 높게 나올 때가 많음.

- Data Scaling을 크게 고민할 필요가 없다!!
- 그래도 scaling을 하고자 할 때는 StandardScaler 사용 추천.

Data preprocessing

- 주의점

- 학습 데이터 세트와 테스트 데이터 세트로 분리한 후 학습 데이터 세트에 스케일링을 적용하고 같은 method로 테스트 데이터 세트에 scaling 적용

- 학습 데이터와 테스트 데이터 세트로 분리한 후 학습 데이터와 테스트 데이터 각각에만 biased된 방식으로 scaling하면 bias 부여 가능성 있음

- 전체 데이터셋에 동일한 스케일링을 적용하는 것은 cheating이 발생할 가능성이 있음..! (테스트 데이터에도 이미 학습 데이터의 정보가 영향을 주었을 가능성이 있음)

- Real-time Validation, External Validation 필요

- 하지만 아까 언급했듯이 data scaling에 크게 신경 쓸 필요가 없다!!

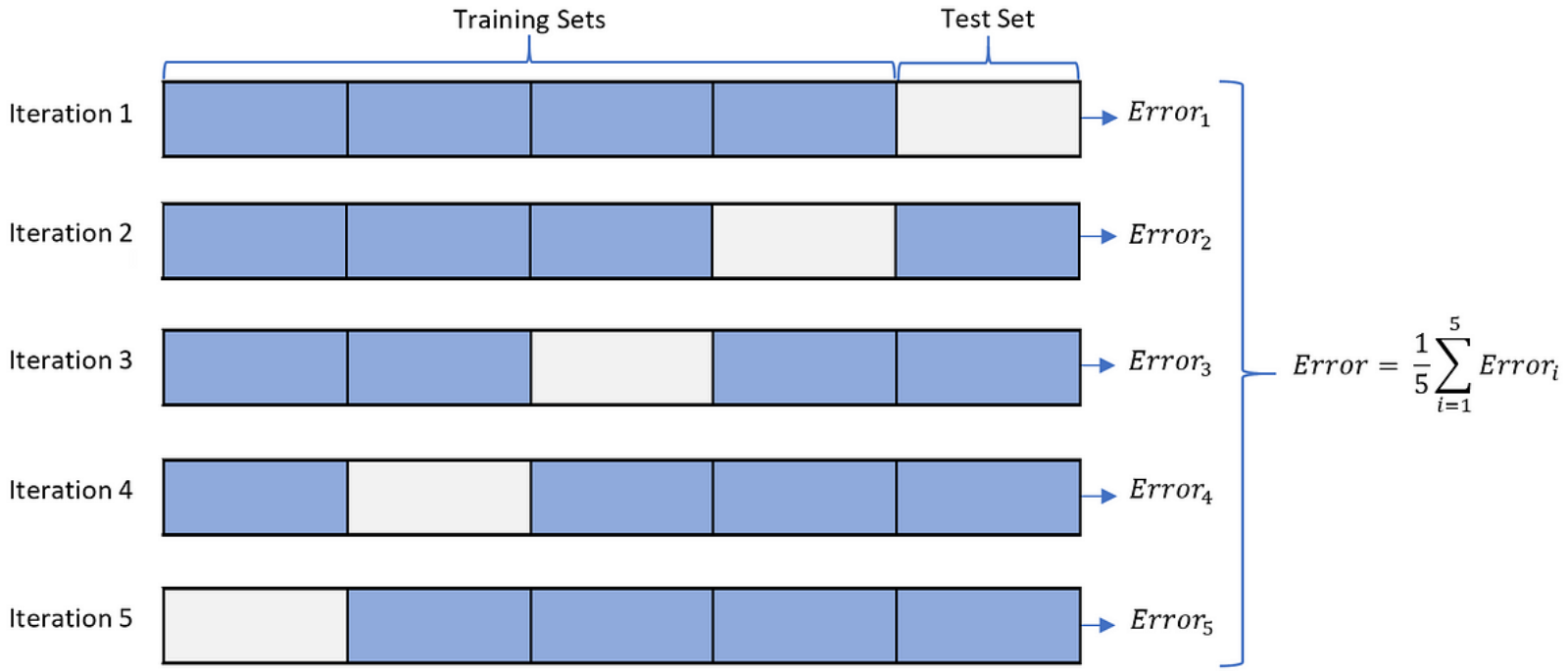
Data preprocessing

(+)

(1) Outlier 제거 (Data Distribution 확인 후)
(2) SMOTE 기반 Over Sampling 기법 … Data Augmentation

COMING UP NEXT…!
Data Preprocessing 2:
Data Imbalance (해결 방법, 논문)
Multi-center 학습 및 노이즈 제거

Cross Validation (2/5/10 Fold CV)



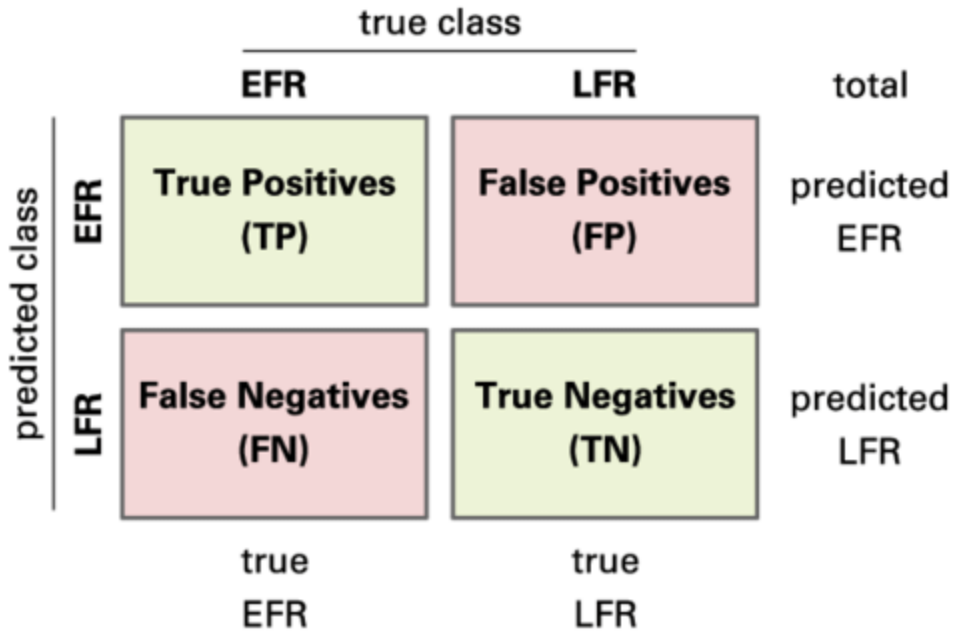
- 학습의 안정성을 위해 사용 (ML의 작은 데이터셋에서 안정적인 학습을 위해 필요)
- Scikit-learn API 사용할 경우 데이터의 분포를 고려해서 Train/Validation set 분할

Evaluation Metrics (Jupyter Notebook)

(1) Accuracy [정확도]
(2) Confusion Matrix

$$Accuracy = \frac{Correct\ prediction}{Total\ cases} * 100\%$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} * 100\%$$



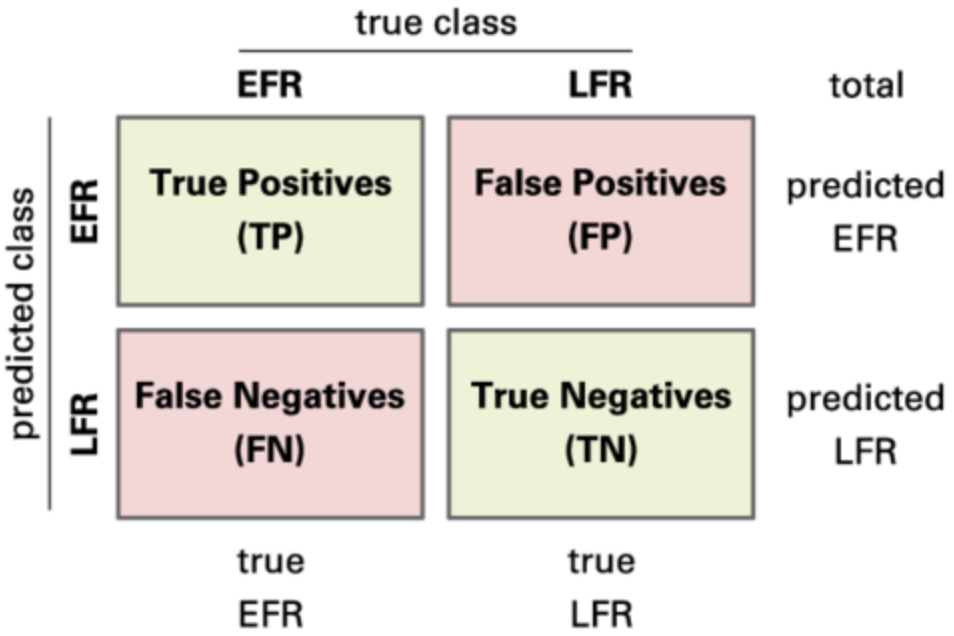
$$PR = \frac{TP}{TP+FP}$$

$$RE = \frac{TP}{TP+FN}$$

$$CA = \frac{TP+TN}{TP+TN+FP+FN}$$

$$F_1 = \frac{2TP}{2TP+FP+FN}$$

Evaluation Metrics



$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

$$\mathrm{CA} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$

$$\mathrm{F_1} = \frac{2\mathrm{TP}}{2\mathrm{TP+FP+FN}}$$

TN: 예측값을 Negative 값 (0)으로 예측했고 실제 값 역시 Negative 값 (0)
FP: 예측값을 Positive 값 (1)으로 예측했는데 실제 값은 Negative 값 (0)
FN: 예측값을 Negative 값 (0)으로 예측했는데 실제 값은 Positive 값 (1)
TP: 예측값을 Positive 값 (1) 로 예측했고 실제 값 역시 Positive 값 (1)

Evaluation Metrics

(3) F1 score

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Precision [정밀도]: 모델이 Positive라고 예측한 것 중에서 실제 Positive인 것의 비율
== PPV [Positive Predictive Value]

Recall [재현율]: 실제 Positive인 것 중에서 모델이 Positive라고 예측한 것의 비율
== sensitivity, hit rate, TPR

F1 score: Precision과 Recall을 균형 있게 분석 (어느 한 쪽으로 치우치지 않을 때 상대적으로 높은 값을 가짐)

Evaluation Metrics

(+)

Specificity: 실제 Negative인 것 중에서 모델이 Negative라고 예측한 것의 비율
NPV [Negative Predictive Value]: 모델이 Negative라고 예측한 것 중에서 실제 Negative인 것의 비율

>> Accuracy, AUC, Sensitivity, Specificity, PPV, NPV

Data preprocessing

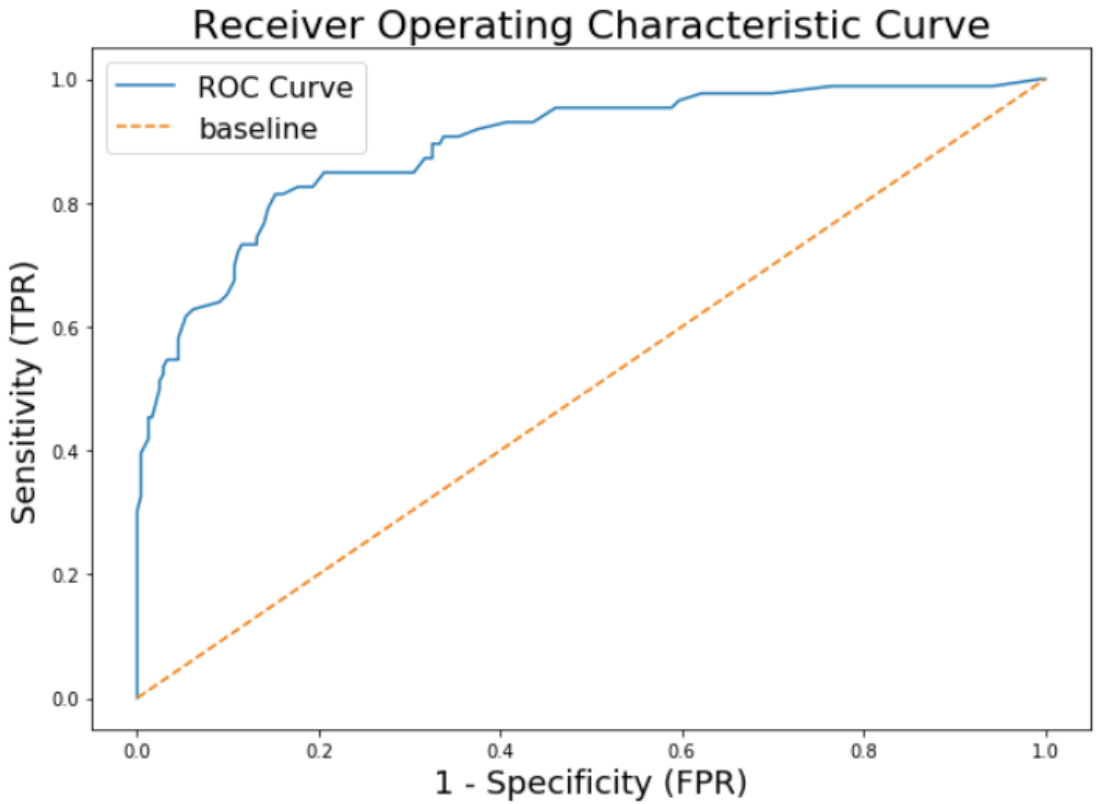
- 주의점

- Threshold를 설정할 수 있는 모델의 경우,
- e.g. 0으로 예측할 것인지 1로 예측할 것인지 -> threshold를 custom하게 조정 가능
- 단순히 성능 지표를 늘리기 위한 수단으로 사용해서는 안됨

- Question
- 모든 threshold에 대해서 평가할 수는 없을까?

Evaluation Metrics

(4) ROC Curve & AUC [Area Under the Curve]



- AUC는 클수록 좋으며 FPR이 작은 상태에서 얼마나 큰 TPR을 얻을 수 있는지가 중요

* FPR: False Positive Rate
* TPR: True Positive Rate

(AUC는 0.5이상, 일반적인 분류기에서 0.5이하의 값이 나온다면 계산 오류)

Data preprocessing

- Question
- Multi-class classification의 경우 ROC Curve는 어떻게?

- https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
- One-vs-One [OvO] multiclass ROC / One-vs-Rest [OvR] multiclass ROC
- https://github.com/duneag2/vit-xgboost-imaging-genomics

Evaluation Metrics

(5) MCC

Matthews correlation coefficient (Phi coefficient)
Binary classification (Class Imbalance가 있을 때 사용하기 좋음)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Data preprocessing

- Regression 성능 지표

RMSE, R Square, MAE, MSLE, RMSLE, Pearson Correlation Coefficient

Reference

(1) 파이썬 머신러닝 완벽 가이드
(2) https://inhovation97.tistory.com/65
(3) https://dining-developer.tistory.com/19
(4) https://wooono.tistory.com/103
(5) https://mkjjo.github.io/python/2019/01/10/scaler.html
(6) https://ivoryrabbit.github.io/%EC%88%98%ED%95%99/2021/03/12/%EB%A7%A4%ED%8A%9C%EC%83%81%EA%B4%80%EA%B3%84%EC%88%98.html

COMING UP NEXT…!

오늘 설명한 개념들을 모두 직접 코딩할 필요가 없다...?!! (feat. AutoML)

Q & A