빅데이터분석 실습

평가지표와 측정

데이터 사이언스 전공

담당교수: 곽철완

강의 내용

- 이진 분류의 평가 지표

▪ 평가 지표와 목표

- 어떤 평가 지표를 선택할까?
  - 특정 머신러닝 알고리즘을 선택한 이유
  - 데이터를 통해 미래 예측 뿐만 아니라 기업(혹은 기관)의 중요한 의사결정의 일부로 사용될 수 있다
  - 특정 머신러닝 알고리즘을 선택하여 나타난 결과를 비즈니스 임팩트라 한다
- 목적에 적합한 머신러닝 알고리즘 평가 지표를 선택해야 한다

이진 분류의 평가 지표

▪ 에러의 종류

- 암 조기 발견 애플리케이션
  - 일반적인 이진 분류는 양성 클래스와 음성 클래스로 구분한다
    - 보통 양성 클래스에 관심을 갖는다
  - 암 테스트에서 음성은 암이 활성화되지 않아 건강하다는 의미이다
  - 건강한 사람을 양성으로 판단한 경우 → 거짓 양성(타입 I 에러)
    - 추가 검진 등으로 비용이 들어간다
  - 암에 걸린 사람을 음성으로 판단한 경우 → 거짓 음성(타입 II 에러)
    - 암을 발견하지 못해 치명적인 문제가 발생할 수 있다
  - 애플리케이션은 거짓 음성을 최대한 피해야 한다

▪ 불균형 데이터 세트

- 에러의 원인 중 하나는 데이터 세트의 불균형의 결과이다

▪ 사례

- 인터넷에서 상품을 사용자에 보여주면 사용자가 보여진 노출 데이터로 클릭을 예측하는 애플리케이션을 개발한다
- 목표는 상품을 보여주면 사용자가 클릭할 지(관심 대상인지) 예측하는 것이다
  - 필요하다면 사용자가 클릭할 때까지 100개의 글이나 광고를 보여주어야 한다
  - 이때 클릭이 아닌 데이터 99개와 클릭 데이터 1개가 데이터 세트로 만들어진다 → 불균형 데이터 세트

- 불균형 데이터 세트를 사용하면 발생하는 결과 예시
  - digits 데이터 세트를 이용해서 숫자 9를 다른 숫자와 구분하여 9:1의 불균형 데이터 세트를 만들었다

```
from sklearn.datasets import load_digits

digits = load_digits()
y = digits.target == 9

X_train, X_test, y_train, y_test = train_test_split(digits.data, y, random_state=0)
```

  - 항상 다수의 클래스를 예측값으로 제시하는 DummyClassifier를 사용하여 정확도를 계산한다

```
from sklearn.dummy import DummyClassifier
dummy_majority = DummyClassifier(strategy='most_frequent').fit(X_train, y_train)
pred_most_frequent = dummy_majority.predict(X_test)
print("예측된 레이블의 고유값:", np.unique(pred_most_frequent))
print("테스트 점수: {:.2f}".format(dummy_majority.score(X_test, y_test)))
```

```
예측된 레이블의 고유값: [False]
테스트 점수: 0.90
```

- 아무런 학습을 하지 않았어도 90%의 정확도를 얻었다

◦ 실제 분류기를 사용하여 정확도를 측정한다

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=2).fit(X_train, y_train)
pred_tree = tree.predict(X_test)
print("테스트 점수: {:.2f}".format(tree.score(X_test, y_test)))
```

```
테스트 점수: 0.92
```

◦ 정확도가 dummy 분류기보다 조금 나아졌다
◦ 그렇다면, 이 문제에서는 '정확도 높은 측정 방법 ' 을 선택한 것이 적합하지 않다고 추정할 수 있다

- 다른 사례를 살펴본다
- dummy 분류기와 로지스틱 회귀 분석을 비교한다
  - dummy: 82%
  - 회귀분석: 98%
- dummy가 80%를 맞추었다
이는 불균형 데이터 세트
의 문제이다
- 즉, 정확도가 적절한 방법이
아니다

```
from sklearn.linear_model import LogisticRegression

dummy = DummyClassifier().fit(X_train, y_train)
pred_dummy = dummy.predict(X_test)
print("dummy 점수: {:.2f}".format(dummy.score(X_test, y_test)))

logreg = LogisticRegression(C=0.1).fit(X_train, y_train)
pred_logreg = logreg.predict(X_test)
print("logreg 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

```
dummy 점수: 0.82
logreg 점수: 0.98
```

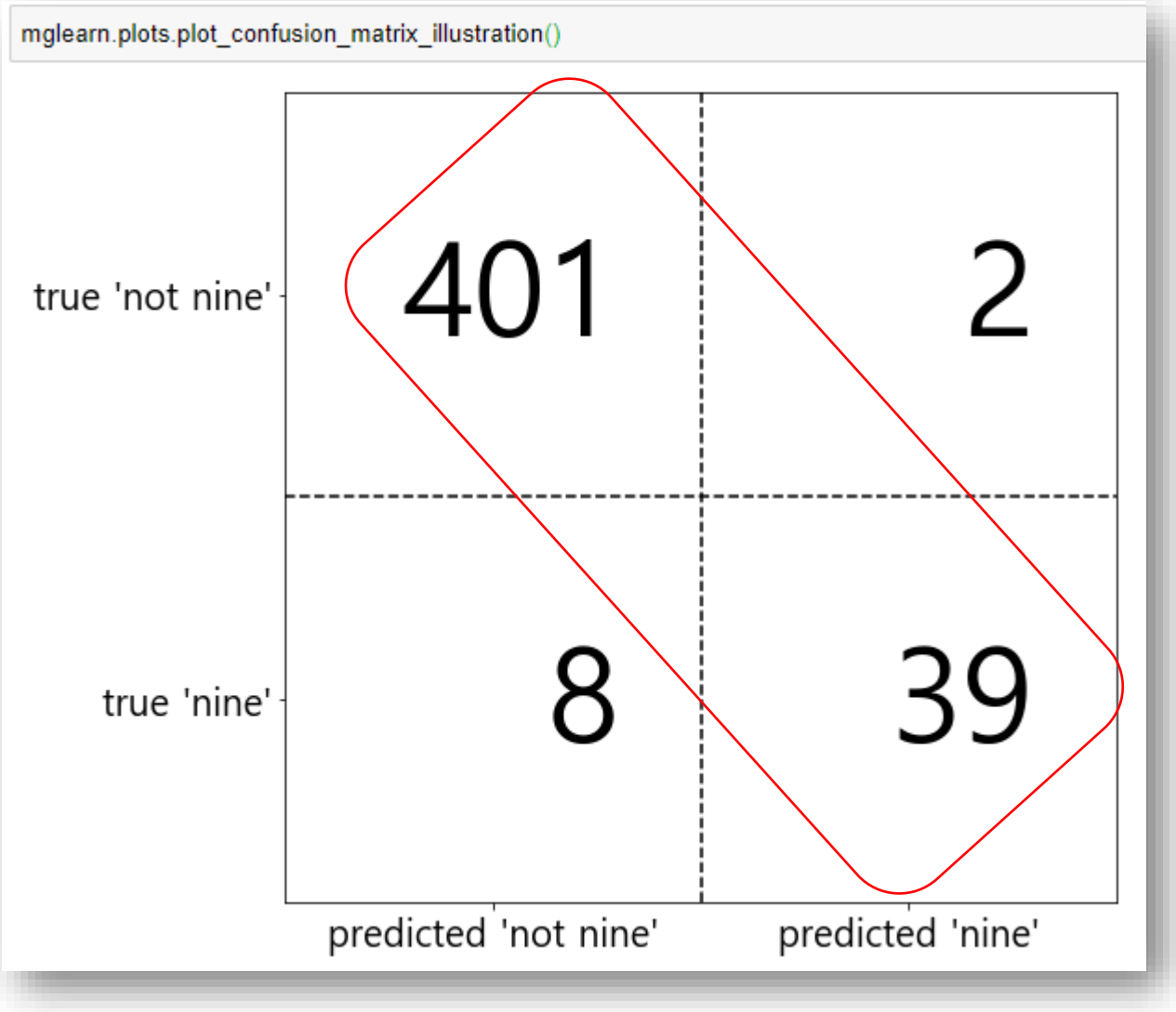
▪ 오차 행렬

- 이진 분류 평가 결과를 나타낼 때 가장 많이 사용하는 방법 중 하나이다
- 앞의 로지스틱 회귀분석의 예측 결과를 confusion_matrix 함수를 사용해서 확인한다

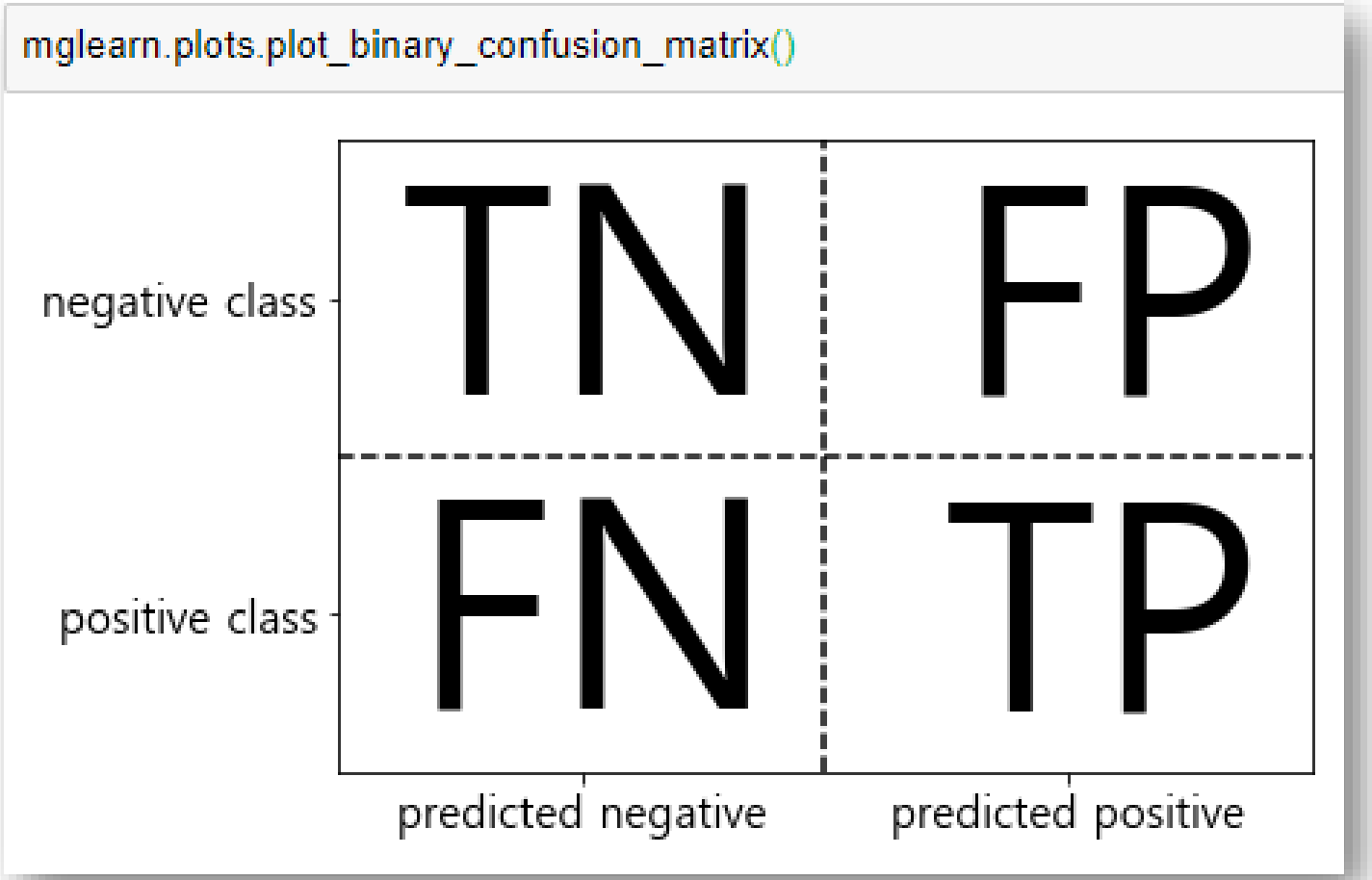
```
from sklearn.metrics import confusion_matrix

confusion = confusion_matrix(y_test, pred_logreg)
print("오차 행렬:\n", confusion)
```

```
오차 행렬:
 [[401   2]
 [  8  39]]
```

- 9가 아님 → 정확한 예측 401개(잘못 예측 8)
- 9 → 정확한 예측 39개(잘못 예측 2)
- 대각선이 정확히 분류된 것이다



- 암 예측과 연계하면, 정확하게 분류된 것을 TN, TP이라 한다
- 잘못 분류된 것을 PN, FP라 한다
- 이 오차 행렬을 이용하여 dummy, 결정 트리, 로지스틱 회귀를 비교한다



- 빈도 기반 더미 모델
  - 동일한 클래스를 예측한다
- 무작위 더미 모델
  - 진짜 양성 수가 너무 적다
- 결정 트리
- 로지스틱 회귀
  - 모든 면에서 결정 트리보다 낫다

```
print("빈도 기반 더미 모델:")
print(confusion_matrix(y_test, pred_most_frequent))
print("\n무작위 더미 모델:")
print(confusion_matrix(y_test, pred_dummy))
print("\n결정 트리:")
print(confusion_matrix(y_test, pred_tree))
print("\n로지스틱 회귀")
print(confusion_matrix(y_test, pred_logreg))
```

```
빈도 기반 더미 모델:
[[403   0]
 [ 47   0]]

무작위 더미 모델:
[[367  36]
 [ 44   3]]

결정 트리:
[[390  13]
 [ 24  23]]

로지스틱 회귀
[[401   2]
 [  8  39]]
```

■ 정확도와의 관계

- 정확도(accuracy)

$$\text{정확도} = \frac{TP + TN}{TP + TN + FP + FN}$$

  - 정확히 예측한 수를 전체 샘플 수로 나눈 것이다

- 정확률(precision)
  - 양성으로 예측된 샘플 중 진짜 양성으로 분류된 샘플의 비율
  - '양성 예측도'라고도 한다

$$\text{정확률} = \frac{TP}{TP + FP}$$

- 재현률(racall)
  - 진짜 양성 샘플 중에서 양성으로 분류된 샘플의 비율
  - 진짜 양성 비율이라고도 한다

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

- f1-score

$$\text{F} = 2 \cdot \frac{\text{정확률} \cdot \text{재현율}}{\text{정확률} + \text{재현율}}$$

◦ 숫자 9와 9가 아님 데이터 세트를 이용한 예측 사례

```
from sklearn.metrics import f1_score
print("빈도 기반 더미 모델의 f1 score: {:.2f}".format(
    f1_score(y_test, pred_most_frequent)))
print("무작위 더미 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_dummy)))
print("트리 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_tree)))
print("로지스틱 회귀 모델의 f1 score: {:.2f}".format(
    f1_score(y_test, pred_logreg)))
```

```
빈도 기반 더미 모델의 f1 score: 0.00
무작위 더미 모델의 f1 score: 0.11
트리 모델의 f1 score: 0.55
로지스틱 회귀 모델의 f1 score: 0.92
```

◦ f1-score를 사용하여 평가하면 직관적으로 좋은 모델을 파악할 수 있으나, 이해 혹은 설명이 힘들다

- classification_report 함수
  - 빈도 기반 더미 분류기 평가
  - 9 를 양성으로 했다
  - support: 클래스에 있는 진짜 샘플 수
  - micro avg: 클래스별 정확률 평균 수치
  - macro avg: 클래스별 점수의 평균 수치
  - weighted avg: 클래스의 샘플 수로 가중 평균한 수치

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred_most_frequent,
                            target_names=["9 아님", "9"]))
```

```
              precision    recall  f1-score   support

       9 아님       0.90      1.00      0.94       403
           9       0.00      0.00      0.00        47

   micro avg       0.90      0.90      0.90       450
   macro avg       0.45      0.50      0.47       450
weighted avg       0.80      0.90      0.85       450
```

- 무작위 더미 분류기 평가
  - '9' 클래스의 f1-score는 7% 이지만 '9 아님' 클래스는 90%이다
- 로지스틱 회귀
  - 9, 9 아님의 f1-scroe가 89%, 99%로 우수하다

```
print(classification_report(y_test, pred_dummy,
                            target_names=["9 아님", "9"]))
```

	precision	recall	f1-score	support
9 아님	0.89	0.91	0.90	403
9	0.08	0.06	0.07	47
micro avg	0.82	0.82	0.82	450
macro avg	0.48	0.49	0.49	450
weighted avg	0.81	0.82	0.81	450

```
print(classification_report(y_test, pred_logreg,
                            target_names=["9 아님", "9"]))
```

	precision	recall	f1-score	support
9 아님	0.98	1.00	0.99	403
9	0.95	0.83	0.89	47
micro avg	0.98	0.98	0.98	450
macro avg	0.97	0.91	0.94	450
weighted avg	0.98	0.98	0.98	450

▪ 불확실성 고려

- 분류기는 decision_function 이나 predict_proba 메서드를 제공한다
  - 이진 탐색에서 decision_function은 0, predict_proba는 0.5를 임계값으로 사용한다

- 불균형한 이진 분류 문제 사례
  - 음성 클래스: 400, 양성 클래스: 50

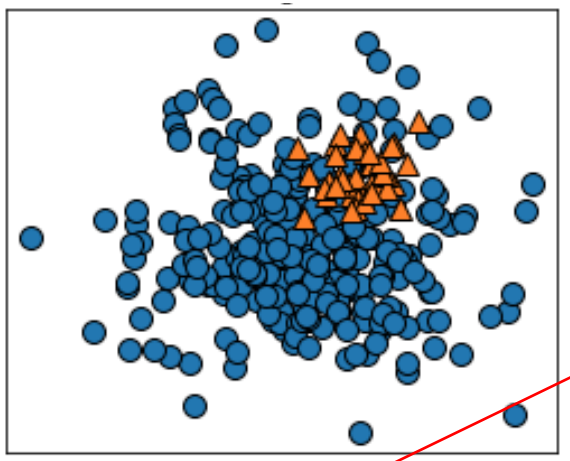
```
X, y = make_blobs(n_samples=(400, 50), cluster_std=[7.0, 2], random_state=22)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
svc = SVC(gamma=.05).fit(X_train, y_train)
```

  - SVM으로 학습시킨 후, 결정 함수 값을 히트맵으로 표현했다

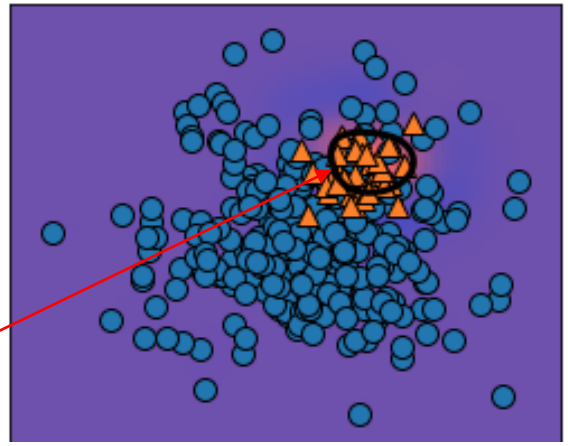
```
mglearn.plots.plot_decision_threshold()
```

결정 임계값

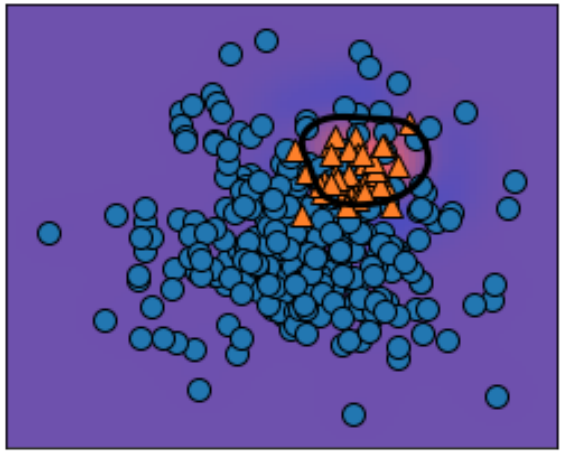
학습용 데이터



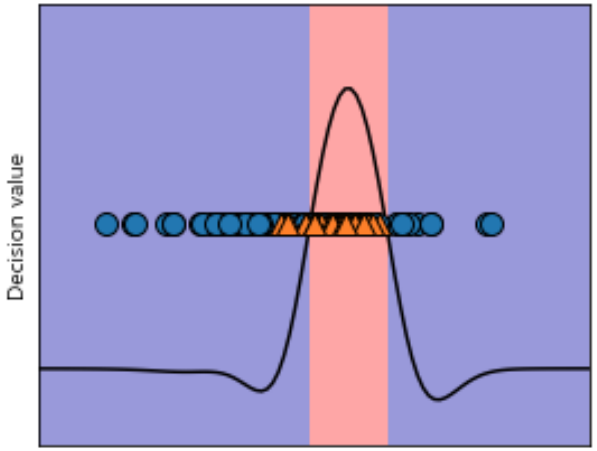
임계값이 0일때



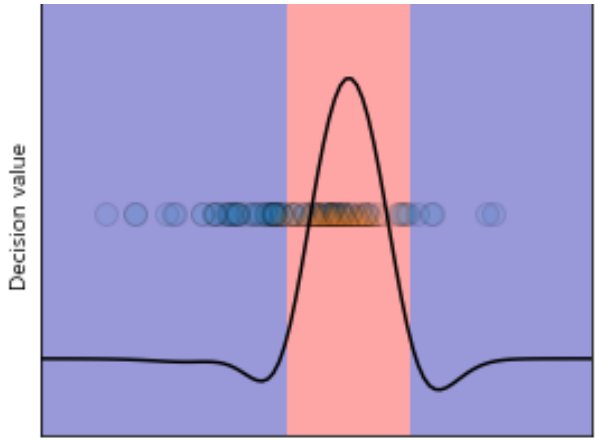
임계값이 -0.8일때



임계값이 8일때 단면도



임계값이 -0.8일때 단면도



검은 원이 decision_function이
정확히 0일때 임계점이다
이 원 안의 포인트는 양성 클래스로
분류된다

◦ classification_report 함수를 이용한 정확률과 재현율 평가

```
print(classification_report(y_test, svc.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.97	0.89	0.93	104
1	0.35	0.67	0.46	9
micro avg	0.88	0.88	0.88	113
macro avg	0.66	0.78	0.70	113
weighted avg	0.92	0.88	0.89	113

- 클래스 0의 샘플 수가 많으므로 분류기는 클래스 0에 초점을 맞추고 있다
- 암 진단의 예에서 본다면 클래스 1의 재현율을 높이는 게 중요할 수 있다

◦ 임계값을 조정하여 클래스 1의 재현율을 높이도록 예측을 조정

```
y_pred_lower_threshold = svc.decision_function(X_test) > -.8
```

```
print(classification_report(y_test, y_pred_lower_threshold))
```

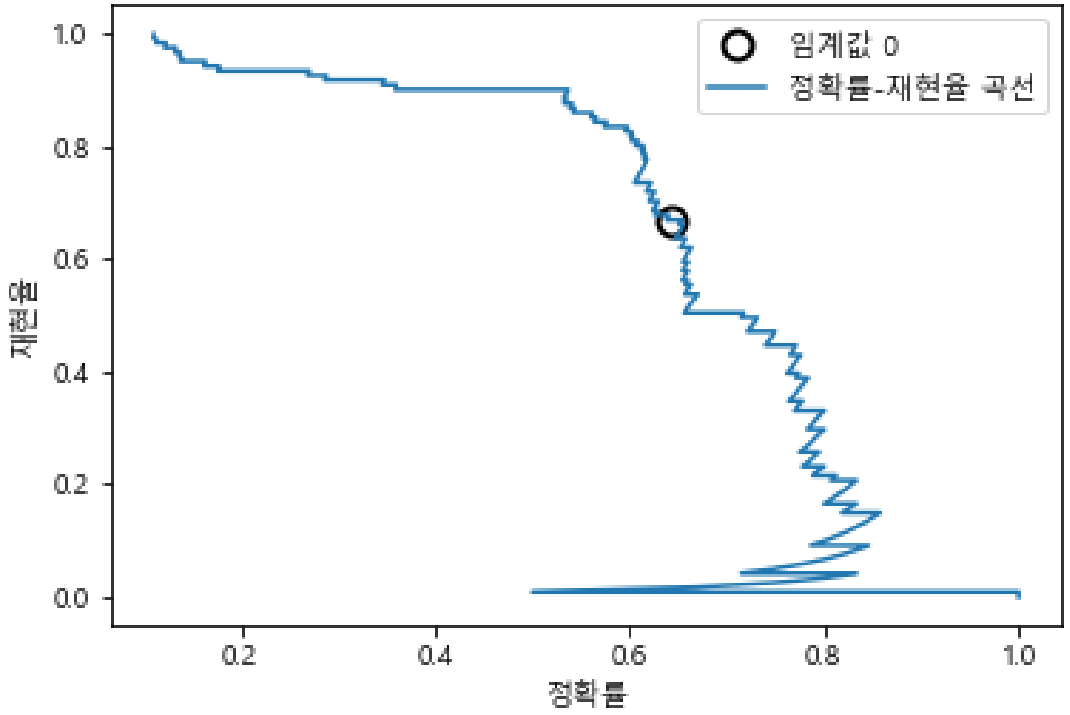
	precision	recall	f1-score	support
0	1.00	0.82	0.90	104
1	0.32	1.00	0.49	9
accuracy			0.83	113
macro avg	0.66	0.91	0.69	113
weighted avg	0.95	0.83	0.87	113

◦ 클래스 1의 재현율이 높아졌고, 정확률은 낮아졌다
◦ 임계값 수정을 통해 재현율 혹은 정확률 조정이 가능하다

- 정밀도-재현율 곡선
  - 모델 분류에서 임계값을 조정하는 것은 분류기의 정확률과 재현율의 관계를 조정하는 것이다
  - 이는 애플리케이션에 따라 다르며, 데이터 분석의 목표에 따라 달라진다
  - 이처럼 분류기의 필요조건을 지정하는 것을 운영포인트를 지정한다고 한다
  - 하지만 새로운 모델을 만들 때는 운영 포인트가 명확하지 않을 수 있다 → 이때 문제 이해를 위해 모든 임계값을 조사하거나 정확률, 재현율의 모든 장단점을 살펴볼 수 있다 → 정확률-재현율 곡선 사용

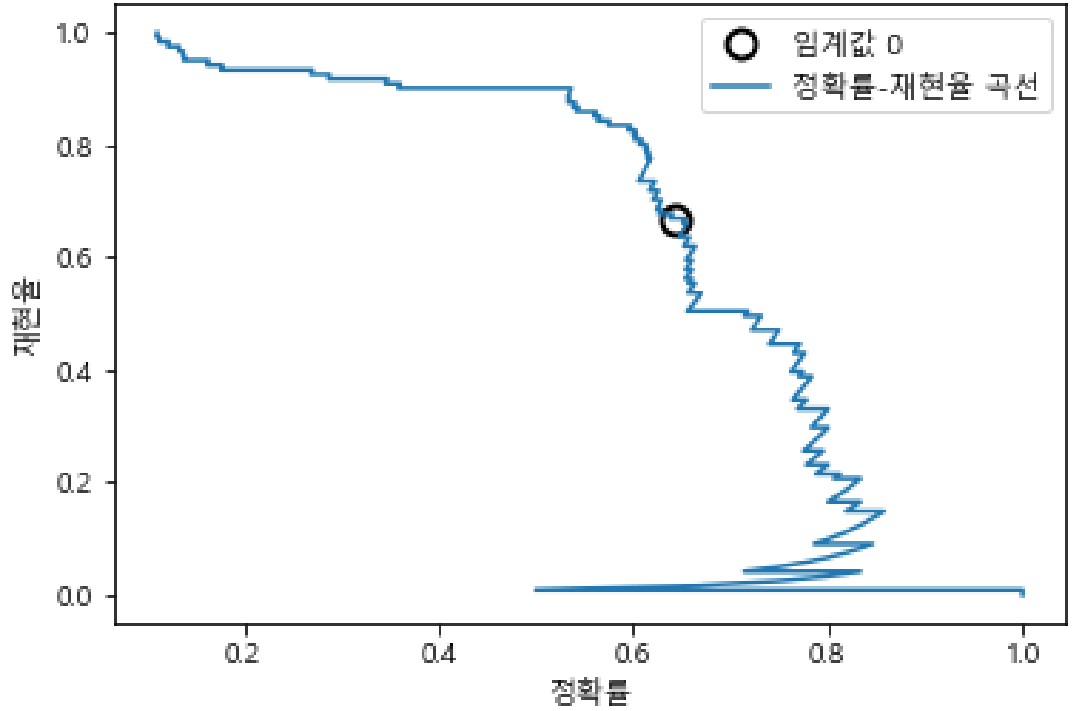
```
from sklearn.metrics import precision_recall_curve
precision, recall, thresholds = precision_recall_curve( y_test, svc.decision_function(X_test))
```

```
# 부드러운 곡선을 위해 데이터 포인트 수를 늘립니다
X, y = make_blobs(n_samples=(4000, 500), cluster_std=[7.0, 2], random_state=22)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

svc = SVC(gamma=.05).fit(X_train, y_train)

precision, recall, thresholds = precision_recall_curve(
    y_test, svc.decision_function(X_test))
# 0에 가까운 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
plt.plot(precision[close_zero], recall[close_zero], 'o', markersize=10,
         label="임계값 0", fillstyle="none", c='k', mew=2)

plt.plot(precision, recall, label="정확률-재현율 곡선")
plt.xlabel("정확률")
plt.ylabel("재현율")
plt.legend(loc="best")
```



```
from sklearn.metrics import precision_recall_curve
precision, recall, thresholds = precision_recall_curve( y_test, svc.decision_function(X_test))
```

```
# 부드러운 곡선을 위해 데이터 포인트 수를 늘립니다
X, y = make_blobs(n_samples=(4000, 500), cluster_std=[7.0, 2], random_state=22)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

svc = SVC(gamma=.05).fit(X_train, y_train)

precision, recall, thresholds = precision_recall_curve(
    y_test, svc.decision_function(X_test))
# 0에 가까운 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
plt.plot(precision[close_zero], recall[close_zero], 'o', markersize=10,
         label="임계값 0", fillstyle="none", c='k', mew=2)

plt.plot(precision, recall, label="정확률-재현율 곡선")
plt.xlabel("정확률")
plt.ylabel("재현율")
plt.legend(loc="best")
```



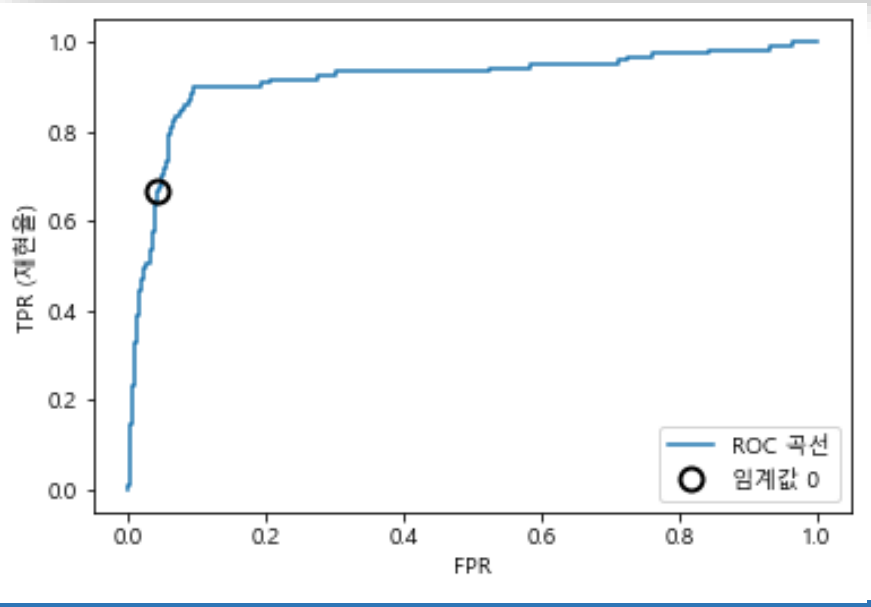
- 임계값 0은 decision_function의 기본 값이다(predict 메서드를 사용할 때 사용되는 임계값이다)
- 곡선이 오른쪽 위로 올라갈 수록 좋은 분류기이다

- ROC
  - 여러 임계값에서 분류기의 특성을 분석하는데 많이 사용하는 도구이다
  - 진짜 양성 비율에 대한 거짓 양성 비율
    - 재현율: 진짜 양성 비율
    - 거짓 양성 비율: 전체 음성 샘플에서 양성으로 잘못 분류한 비율

$$\text{FPR}(\text{거짓양성비율}) = \frac{\text{FP}(\text{음성인데 양성으로 예측된 샘플 수})}{\text{FP} + \text{TN}(\text{음성으로 음성에 예측된 샘플 수})}$$

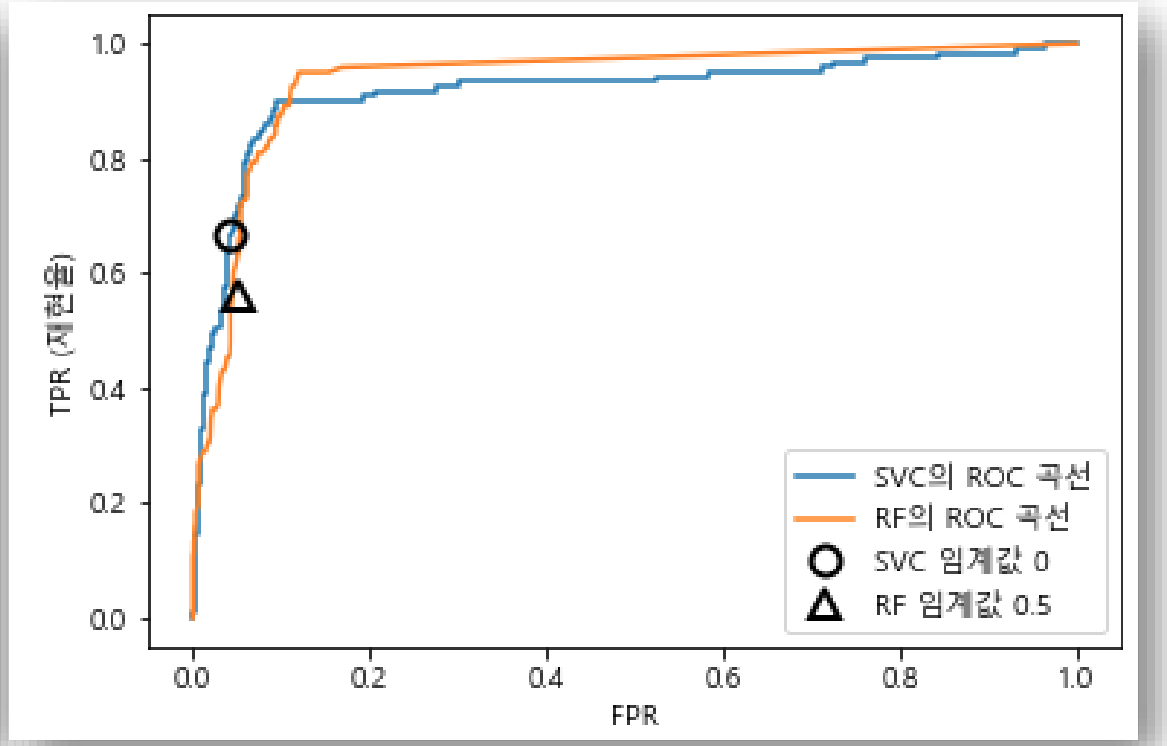
- ROC 곡선은 왼쪽 위에 가까울수록 이상적이다
  - 거짓 양성 비율(FRP)이 낮게 유지되면서 제현율이 높은 분류기가 좋은 것이다
- 현재 임계점을 볼 때, FRP를 조금 늘리면 재현율을 크게 높일 수 있다

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, svc.decision_function(X_test))

plt.plot(fpr, tpr, label="ROC 곡선")
plt.xlabel("FPR")
plt.ylabel("TPR (재현율)")
# 0 근처의 임계값을 찾습니다
close_zero = np.argmin(np.abs(thresholds))
plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
        label="임계값 0", fillstyle="none", c='k', mew=2)
plt.legend(loc=4)
```



◦ SVM과 랜덤포레스트 ROC 곡선 비교

```
from sklearn.metrics import roc_curve
fpr_rf, tpr_rf, thresholds_rf = roc_curve(y_test, rf.predict_proba(X_test)[:, 1])

plt.plot(fpr, tpr, label="SVC의 ROC 곡선")
plt.plot(fpr_rf, tpr_rf, label="RF의 ROC 곡선")

plt.xlabel("FPR")
plt.ylabel("TPR (재현율)")
plt.plot(fpr[close_zero], tpr[close_zero], 'o', markersize=10,
        label="SVC 임계값 0", fillstyle="none", c='k', mew=2)
close_default_rf = np.argmin(np.abs(thresholds_rf - 0.5))
plt.plot(fpr_rf[close_default_rf], tpr[close_default_rf], '^', markersize=10,
        label="RF 임계값 0.5", fillstyle="none", c='k', mew=2)

plt.legend(loc=4)
```



요약

- 머신러닝 모델 선택과 평가에 사용하는 평가 지표와 방법이 중요하다
- 머신러닝 작업의 최종 목적이 높은 정확도의 모델을 만드는 것이 아니다
- 모델은 실제 상황을 잘 대변할 수 있어야 한다
- 그러므로, 평가 지표를 잘 이해하여 적절한 평가 지표를 선택해야 한다

데이터 분석 요약

- 분석방법론
  - 데이터 분석 기획이란?
    - 분석에 앞서 과제 정의 및 결과 도출이 가능하도록 방안 계획
    - 목표 + 데이터 + 방법
  - 데이터 분석 기획 시 고려사항
    - 분석 가능한 데이터 이해: 정형데이터 비정형데이터
    - 활용 가능한 USE CASE: 고객분석, 소셜미디어 분석, 파이프라인 관리
    - 장애 요소 처리 방안: 비용, 단순성, 실행 문화 등

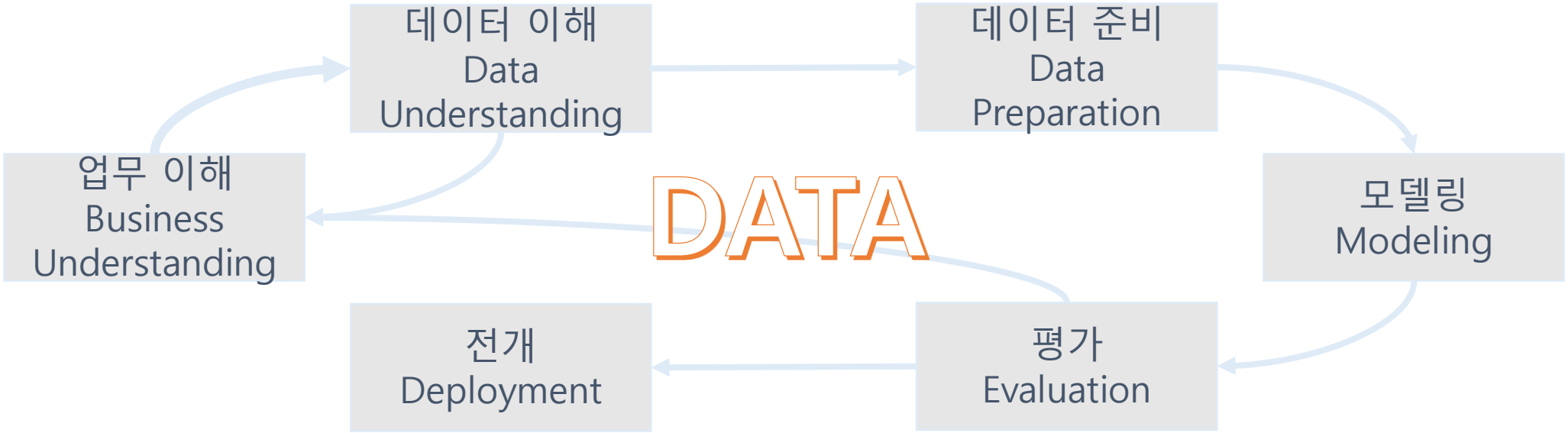
- KDD 분석 방법론
  - 특징
    - 데이터로부터 패턴이나 지식을 발견하기 위한 데이터 마이닝 프로세스
    - 머신러닝, 인공지능 등에서 활용
  - 9개 프로세스

1. 분석 대상 도메인의 이해
2. 분석 대상 데이터셋 선택과 생성
3. 데이터 정제 작업 혹은 전처리 작업
4. 목적에 맞는 피처(변수)를 찾거나 피처 수 축소
5. 목적에 적합한 데이터마이닝 기법 선택
6. 목적에 적합한 알고리즘(예, K-means clustering) 선택
7. 데이터마이닝 실행
8. 데이터마이닝 결과 해석
9. 데이터마이닝에서 발견된 지식 활용

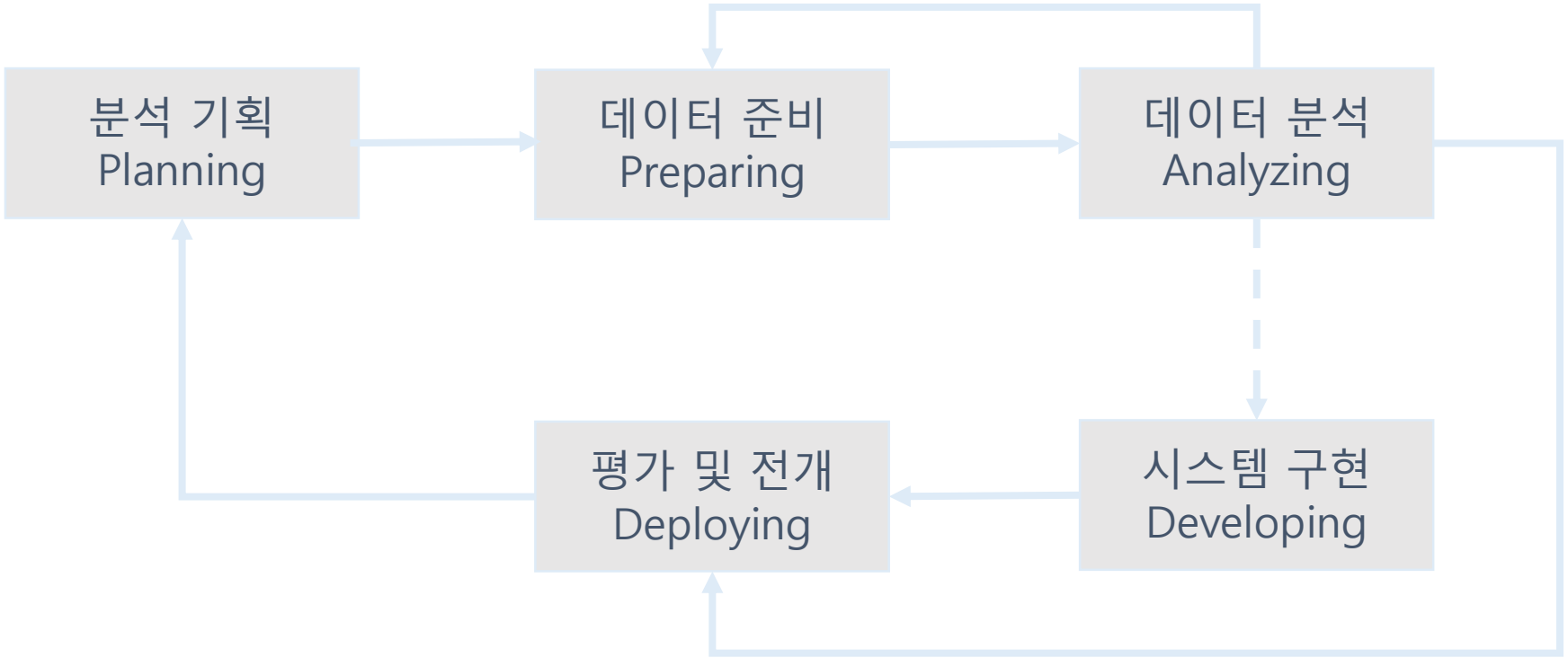
- KDD 분석 절차
  1) 데이터셋 선택
  2) 데이터 전처리(데이터 클리닝)
  3) 데이터 변환(학습용 데이터, 평가용 데이터)
  4) 데이터마이닝(기법 및 알고리즘 선택)
  5) 데이터마이닝 결과 평가

■ CRISP-DM 분석 방법론
- 특징
  - CRoss Industrial Standard Process for Data Mining
  - 계층적 프로세스 모델로 4개 레벨로 구성

- CRISP-DM 프로세스
  - 단계 간 피드백을 통하여 단계별 완성도를 높힘



- 빅데이터 분석 방법론
  - 분석 방법론 5 단계

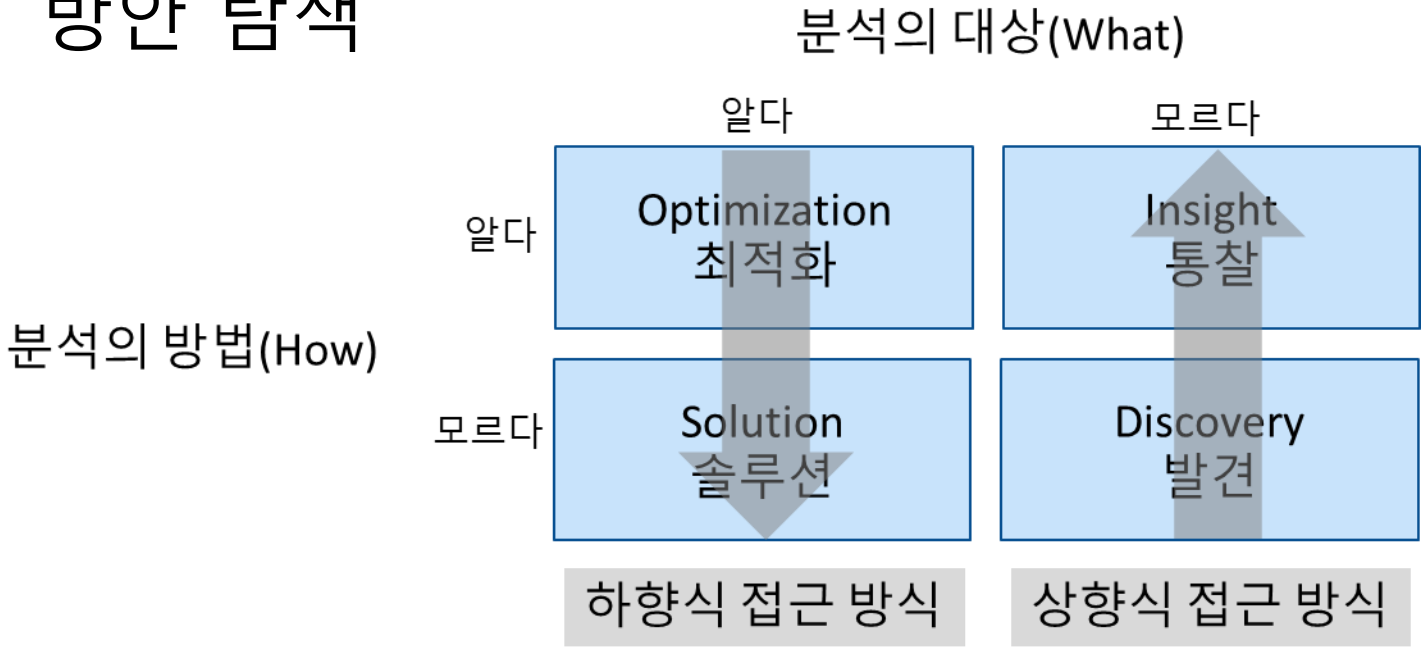


- 분석 기획
  - 비즈니스 이해 및 범위 설정
  - 프로젝트 정의 및 계획 수립
  - 프로젝트 위험 계획 수립
- 데이터 준비
  - 필요 데이터 정의
  - 데이터 스토어 설계(데이터 큐레이션 센터)
  - 데이터 수집 및 정합성 점검
- 데이터 분석(Analyzing)
  - 분석용 데이터 준비
  - 분석용 데이터 준비: a bag-of-words, a set of sequences

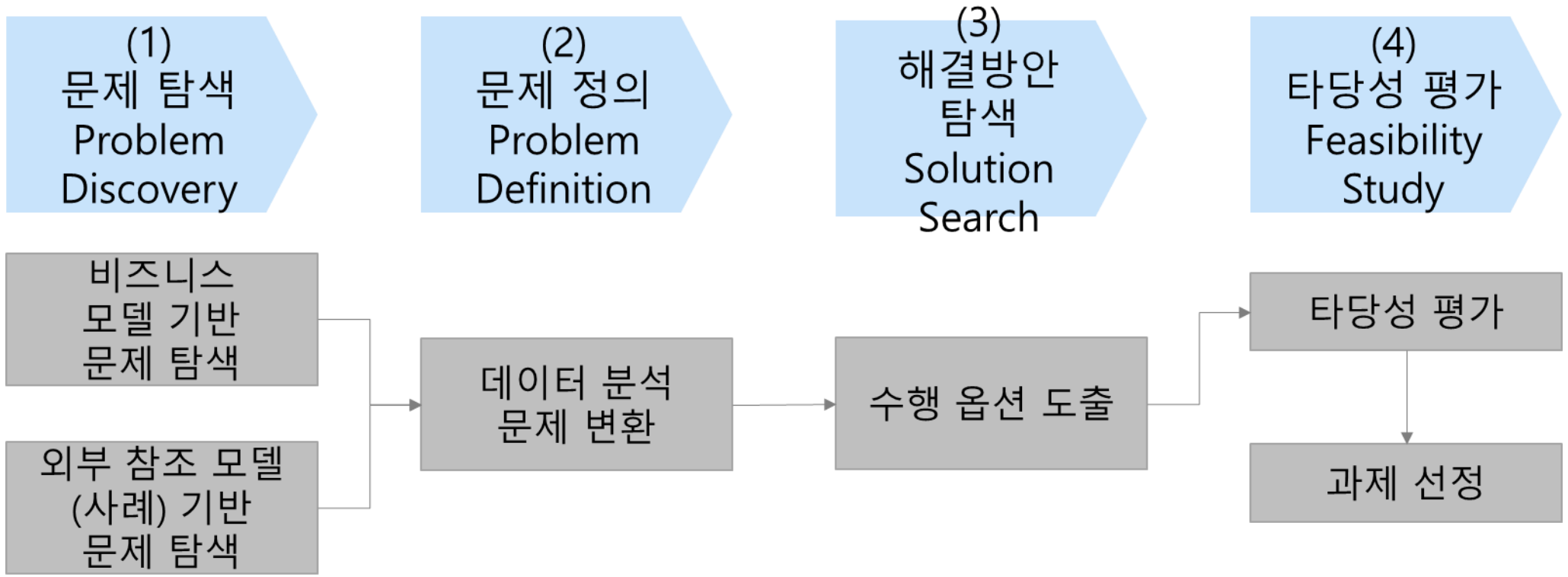
- 탐색적 분석: 통계분석, 연관성 분석, 데이터 시각화
- 모델링: 머신러닝 알고리즘 이용
- 모델 평가 및 검증

- 시스템 구현
  - 설계 및 구현
  - 시스템 테스트 및 운영
- 평가 및 전개
  - 모델 발전 계획 수립
  - 프로젝트 평가 및 보고

- 분석 과제 발굴
  - 과제 발굴 접근 방식
    - 하향식 접근 방식(Top Down Approach)
    - 상향식 접근 방식(Bottom Up Approach): 문제를 정의하고 해결 방안 탐색



◦ 하향식 접근법


(1)
문제 탐색
Problem
Discovery
(2)
문제 정의
Problem
Definition
(3)
해결방안
탐색
Solution
Search
(4)
타당성 평가
Feasibility
Study
비즈니스
모델 기반
문제 탐색
외부 참조 모델
(사례) 기반
문제 탐색
데이터 분석
문제 변환
수행 옵션 도출
타당성 평가
과제 선정

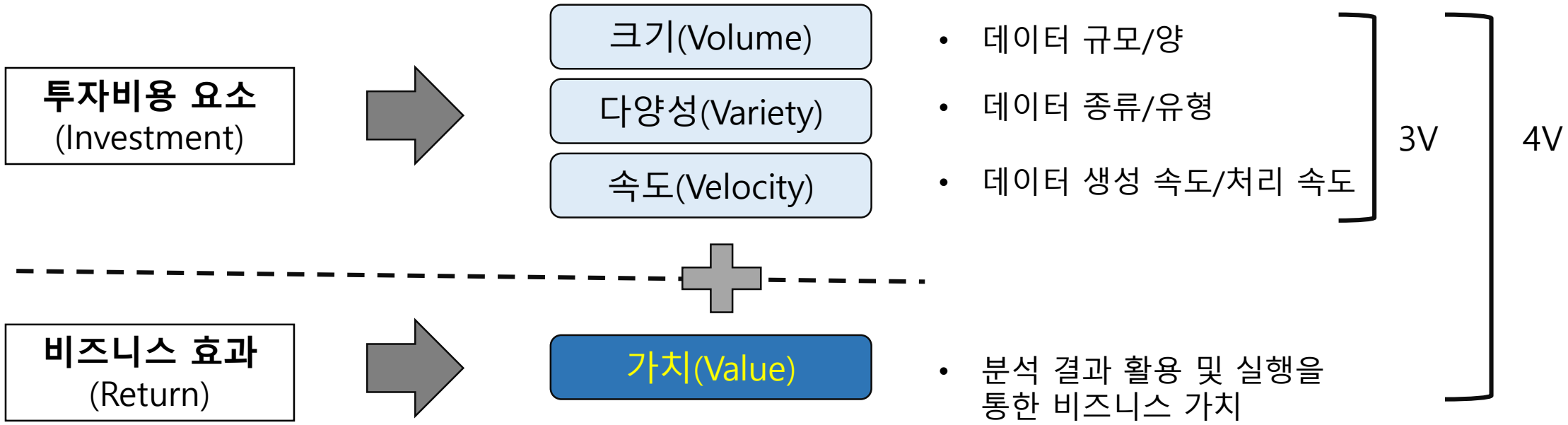
- 상향식 접근법
  - 특징
    - 하향식 접근법의 한계를 극복하기 위한 분석 방법론
      - 하향식 접근법인 논리적인 단계별 접근법은 문제의 구조가 분명하고, 해결책 도출을 위해 데이터 분석가 및 의사결정 권자가 책임을 가지고 있어, 해결책 도출은 유리하지만,새로운 문제 탐색에는 한계가 있어서, 최근 복잡하고 다양한 환경에서 발생하는 문제에는 적합하지 않을 수 있음
    - 디자인 사고(Design Thinking) 접근법을 통해 문제점 해결
    - 사물을 분석적 관점에서 인식하는 것이 아니라, 있는 그대로 'What' 관점에서 접근

- 데이터 분석은 일반적으로 머신러닝의 비지도 학습 (Unsupervised Learning) 적용
- 시행착오를 통한 문제 해결
  - 프로토타이핑(prototyping) 접근법
    - 먼저 분석을 시도하여 프로토타입 결과를 산출 한 후, 반복적으로 개선하는 방법
    - 프로토타입이 완전하지 못하지만, 신속하게 해결책을 제시한 후, 이를 바탕으로 문제를 좀 더 명확하게 인식하고 필요한 데이터를 식별하여 구체화하는 접근 방식

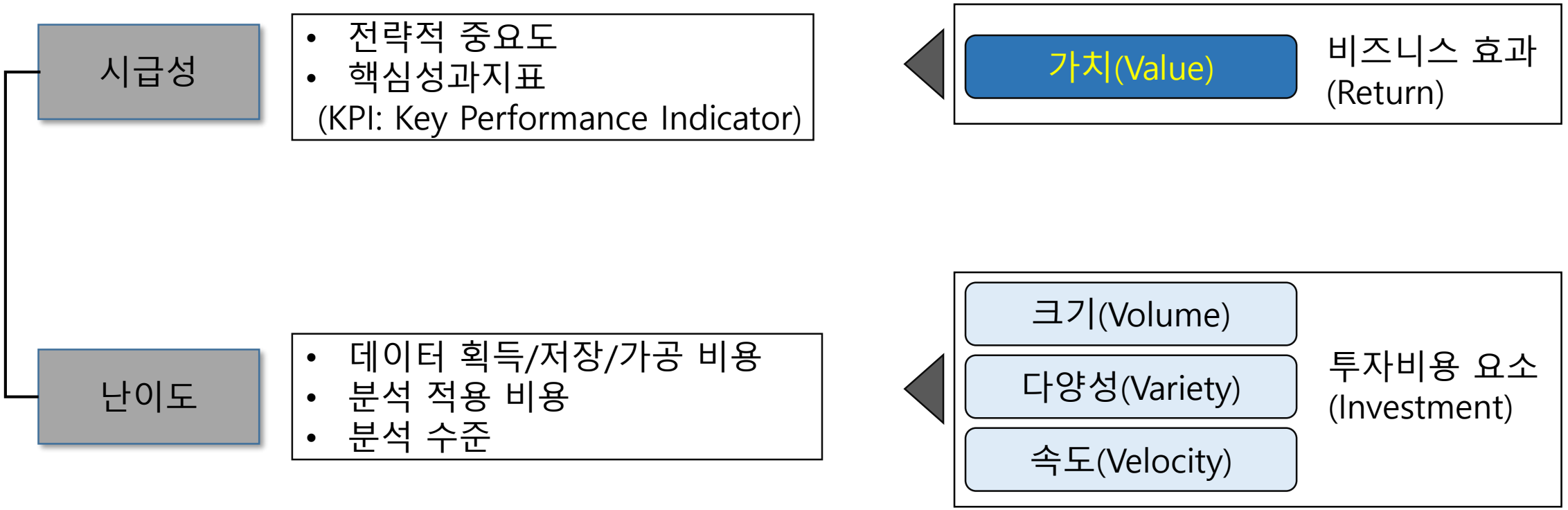
- 분석 프로젝트 관리 방안
  - 분석 과제 관리를 위한 5대 영역
    - Data Size: 데이터 양을 고려한 관리 방안
    - Data Complexity: 비정형 데이터에 대한 분석 모델 선정
    - Speed: 결과가 도출 되었을 때, 이를 활용하는 측면에서 속도 고려(실시간 수행)
    - Analytic Complexity: 복잡도와 정확도는 trade off 관계. 해석이 가능하면서 정확도를 올릴 수 있는 모델 파악
    - Accuracy & Precision
      - 정확도: 모델값과 실제값 차이
      - 정확률: 모델을 지속적으로 반복했을 때 편차 수준의 일관성

- 마스터 플랜 수립
  - 데이터 기반 구축을 위해 적용 우선 순위 설정
  - 데이터 분석 구현을 위한 로드맵 수립
- 우선순위 평가 방법 및 절차
  - 평가 기준
    - 전략적 중요도: 필요성 및 시급성
    - 실행 용이성: 투자 용이성 및 기술 용이성

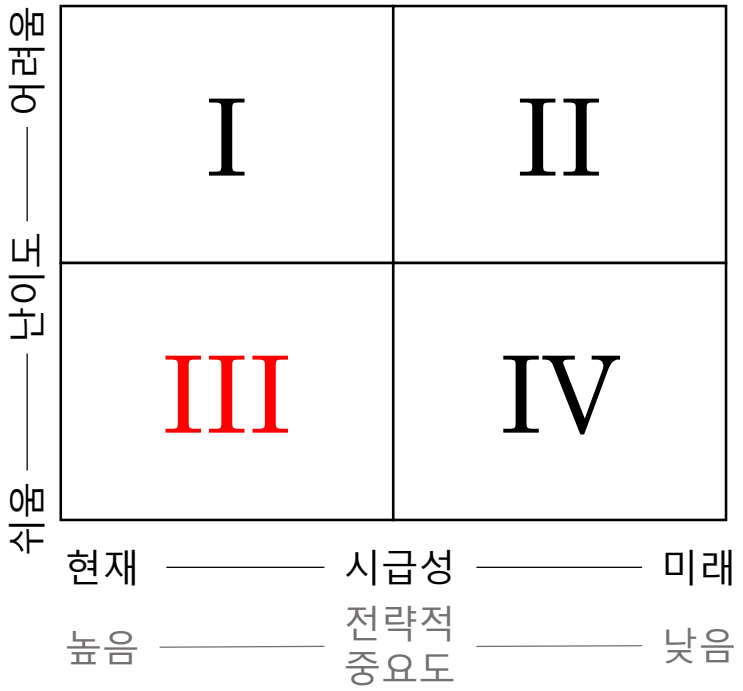
◦ 투자 수익율 관점에서 빅데이터 핵심 특징



◦ 데이터 분석 과제의 우선순위 평가 기준



◦ 포트폴리오 사분면 분석을 통해 과제 우선순위 선정



I	• 전략적으로 중요도가 높아 경영에 미치는 영향이 크므로 현재 시급하게 추진이 필요 • 난이도가 높아 현재 수준에서 과제를 곧바로 적용하기 어려움
II	• 현재 시점에서는 전략적 중요가 높지 않지만 중장기적 관점에서 반드시 추진되어야 함 • 분석 과제를 바로 적용하기에 난이도가 높음
III	• 전략적 중요도가 높아 현재 시점에 전략적 가치를 두고 있음 • 과제 추진의 난이도가 어렵지 않아 우선적으로 곧바로 적용 가능할 필요성이 있음
IV	• 전략적 중요도가 높지 않아 중장기적 관점에서 과제 추진이 바람직함 • 과제를 바로 적용하는 것이 어렵지 않음

- 분석 거버넌스 체계 수립
  - 필요성
    - 기업 운영에 데이터 중요성이 강조될 수록 데이터 분석·활용을 위한 체계적인 관리 필요성 증대
    - 어떤 데이터를, 어떤 목적으로, 어떻게 분석에 활용할 것인가 계획
    - 데이터 분석 문화 정착 및 데이터 분석 업무의 지속적인 고도화

◦ 조직 및 인력 방안 수립

◦ 데이터 분석 조직 구조 유형

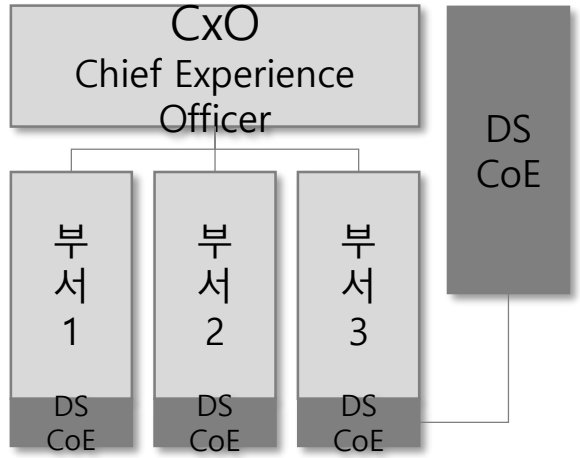
집중 구조

•전담 분석 조직
•우선 순위 운영
•이원화 가능성

기능 구조

•해당 부서에서 분석 업무 담당
•기업 전체의 핵심 분석이 어려움

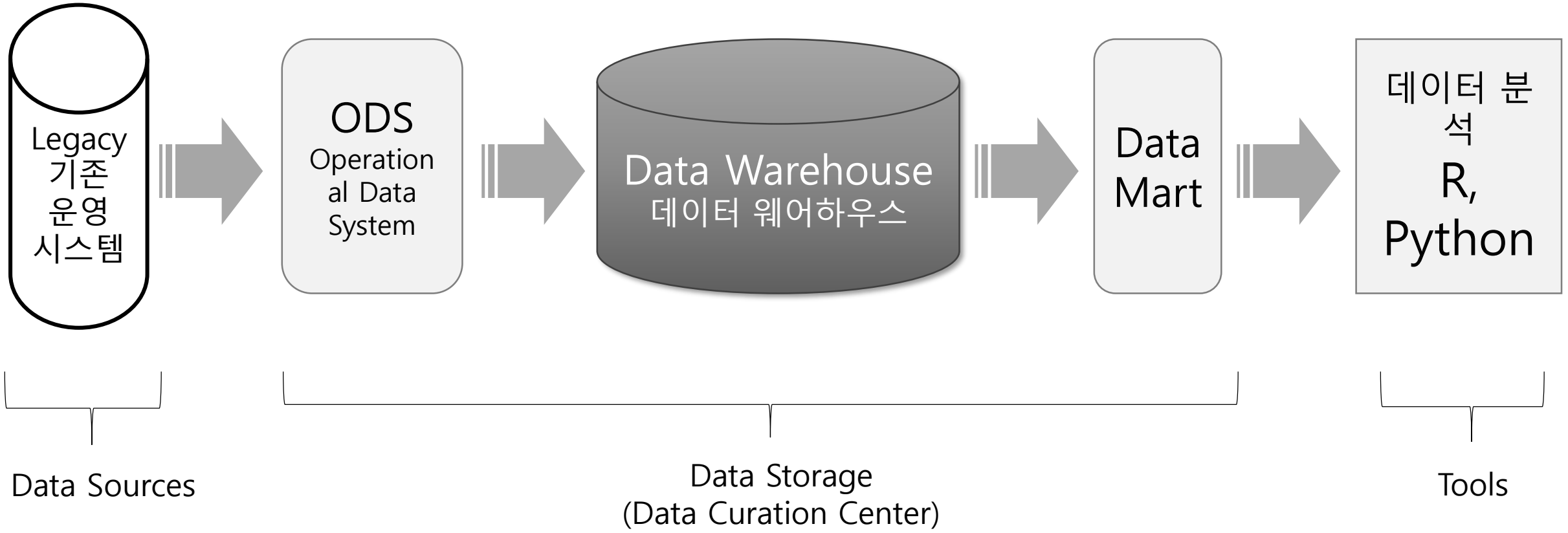
기능 구조



•분석 조직 인력을 현업 부서에 직접 배치 운영
•분석 결과를 신속히 처리 가능

※ DS CoE: Data Science Center of Excellence

- 데이터 분석 기법 소개


Legacy
기존
운영
시스템
ODS
Operation
al Data
System
Data Warehouse
데이터 웨어하우스
Data
Mart
데이터 분
석
R,
Python
Data Sources
Data Storage
(Data Curation Center)
Tools

한 학기 동안 고생 많았습니다