Python Algorithms

- 유니정보 뉴아틀란-

7월 19일

정 준 수

과정 목표

프로그래밍 언어 Python를 활용하여 SW 알고리즘의 구현기술 습득

1. Python 언어특성과 기본 문법 익히기

2. Python 라이브러리를 이용한 고급 알고리즘 구현

3. Tree Algorithm와 Graph 구조 등을 이용한 다양한 머신러닝 알고리즘 응용 사례 구현


크리스찬 베일
스티브 카렐
라이언 고슬링
브래드 피트
AN ADAM McKAY FILM
빅 쇼트
월스트리트를 물먹인 4명의 괴짜 천재들!
믿을 수 없는 실화
〈머니볼〉〈블라인드 사이드〉
작가 원작
PARAMOUNT PICTURES AND REGENCY ENTERPRISES PRESENT
A PLAN B ENTERTAINMENT PRODUCTION AN ADAM McKAY FILM CHRISTIAN BALE
STEVE CARELL RYAN GOSLING BRAD PITT "THE BIG SHORT"
COSTUME DESIGNER SUSAN MATHESON EDITED BY HANK CORWIN, ACE PRODUCTION DESIGNER CLAYTON HARTLEY
DIRECTOR OF PHOTOGRAPHY BARRY ACKROYD, B.S.C. EXECUTIVE PRODUCERS LOUISE ROSNER-MEYER KEVIN MESSICK
PRODUCED BY BRAD PITT, p.g.a. DEDE GARDNER, p.g.a. JEREMY KLEINER, p.g.a. ARNON MILCHAN
BASED UPON THE BOOK BY MICHAEL LEWIS SCREENPLAY BY CHARLES RANDOLPH AND ADAM McKAY DIRECTED BY ADAM McKAY
REGENCY
PLAN B
LOTTE ENTERTAINMENT
1월 21일 대개봉
청소년 관람불가

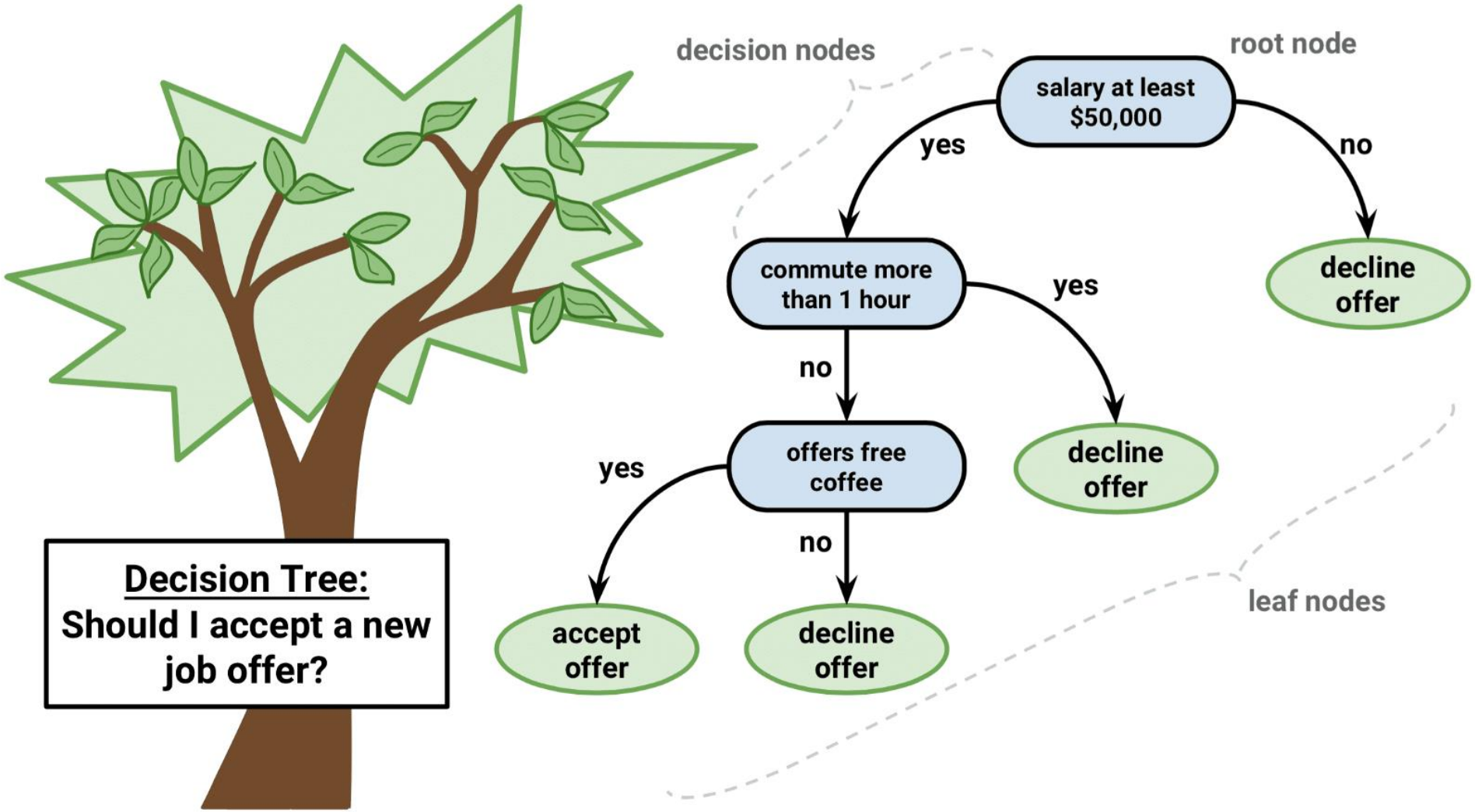
나무에서 돈이 자란다!

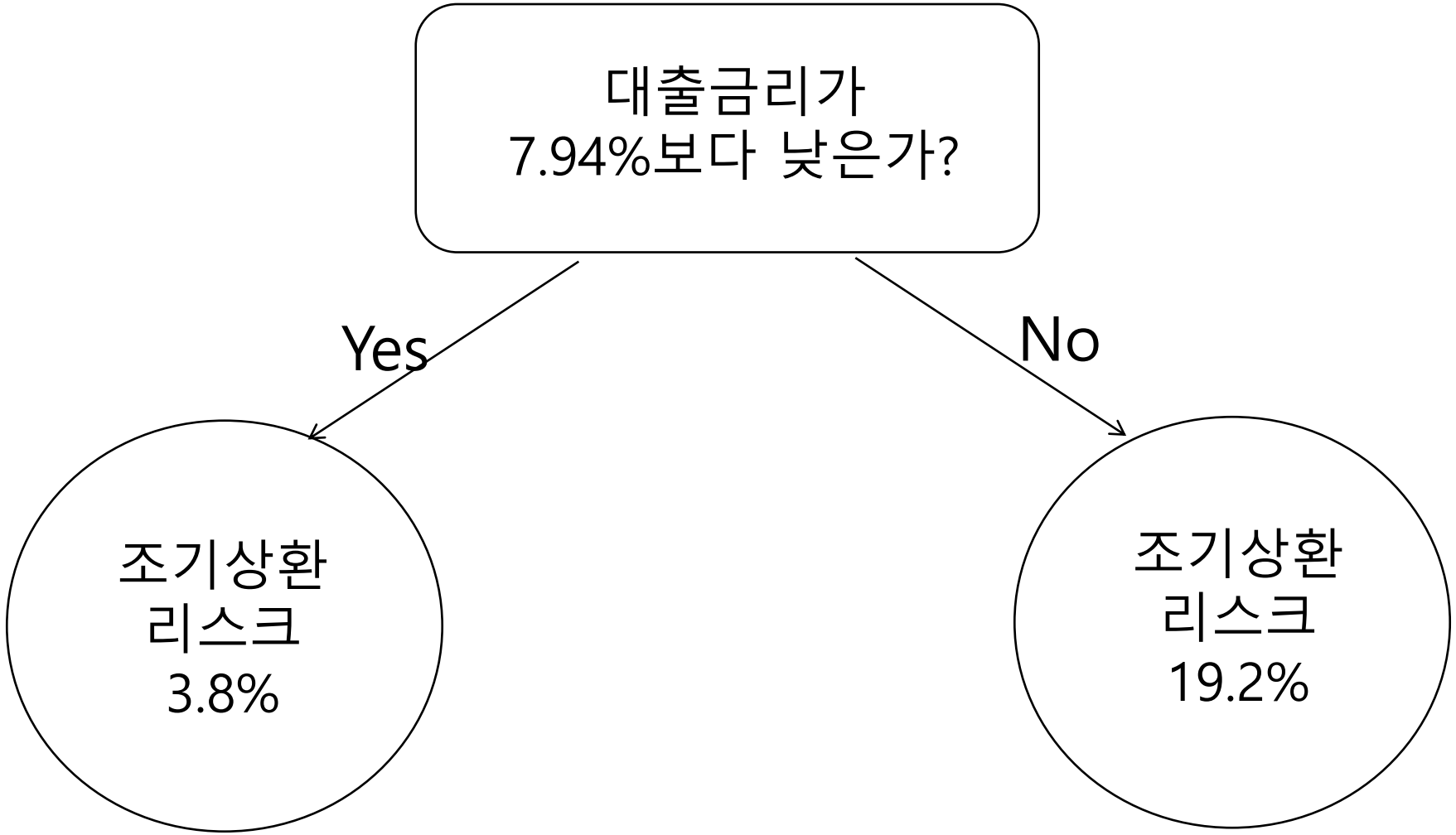
체이스 은행 예측모델은 부동산 담보대출 중에서 실제로

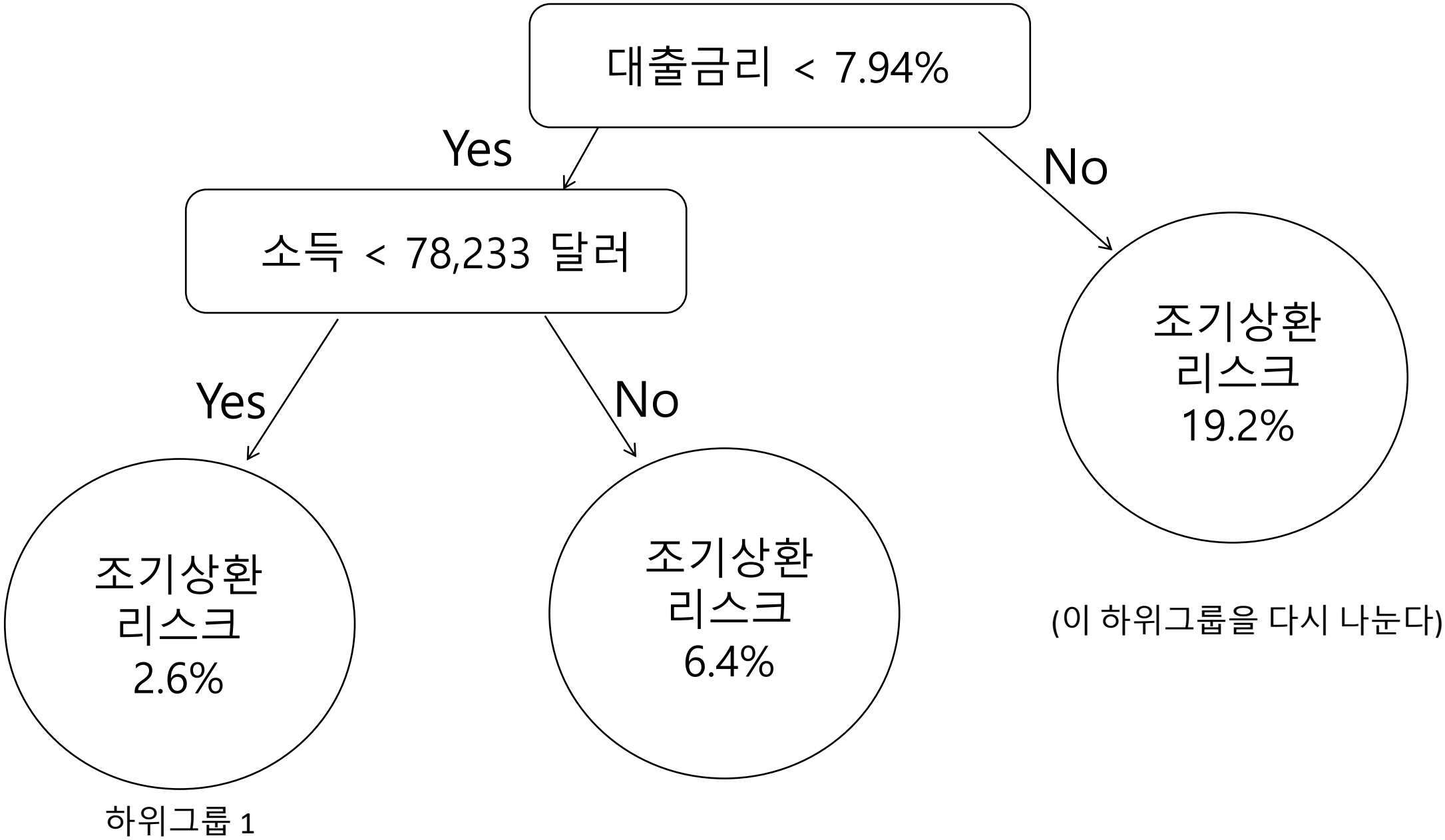
조기상환된 대출 건들 중 74%를 정확하게 인식해 내어서

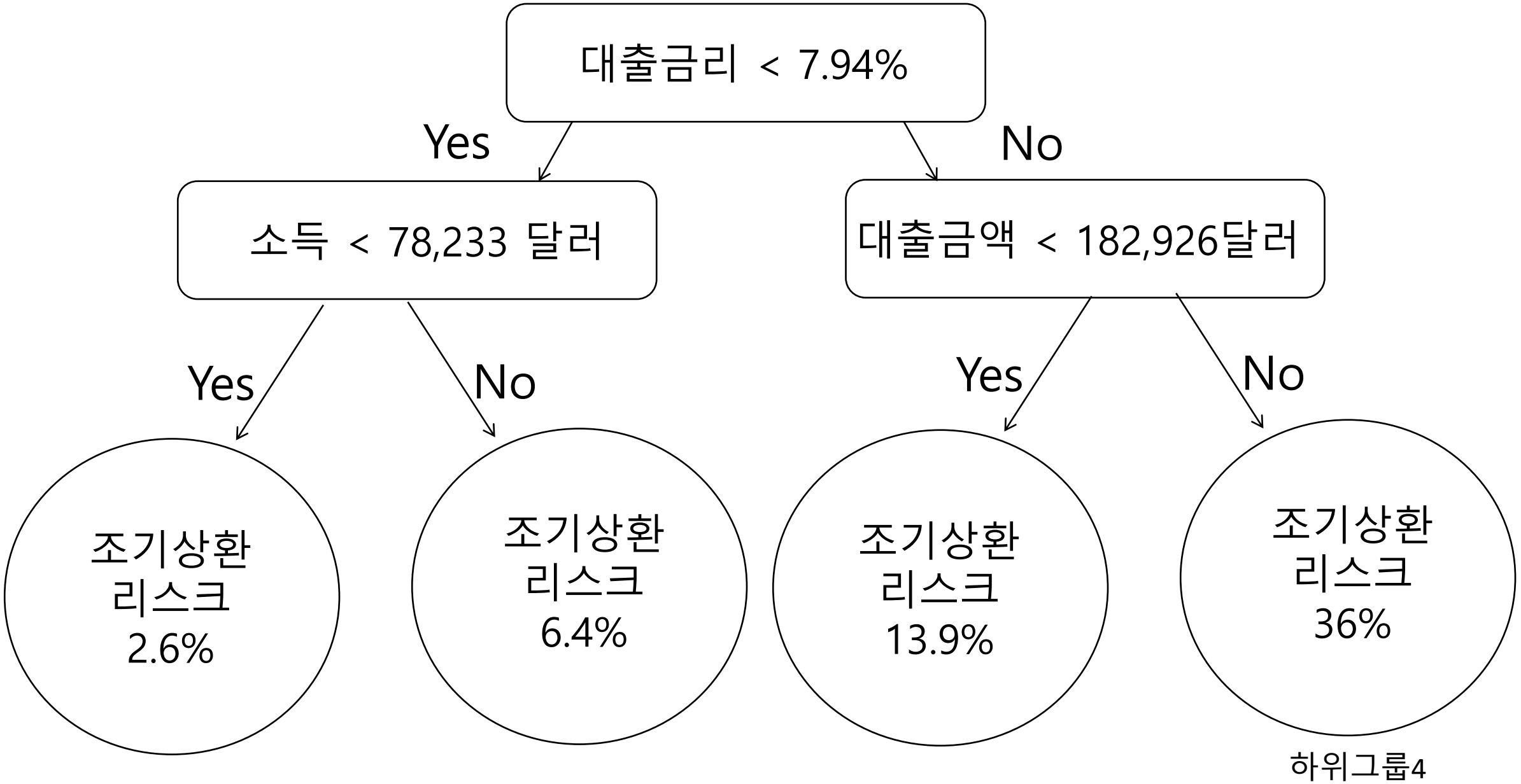
부동산 담보대출 포트폴리오를 성공적으로 관리함.

Decision Tree

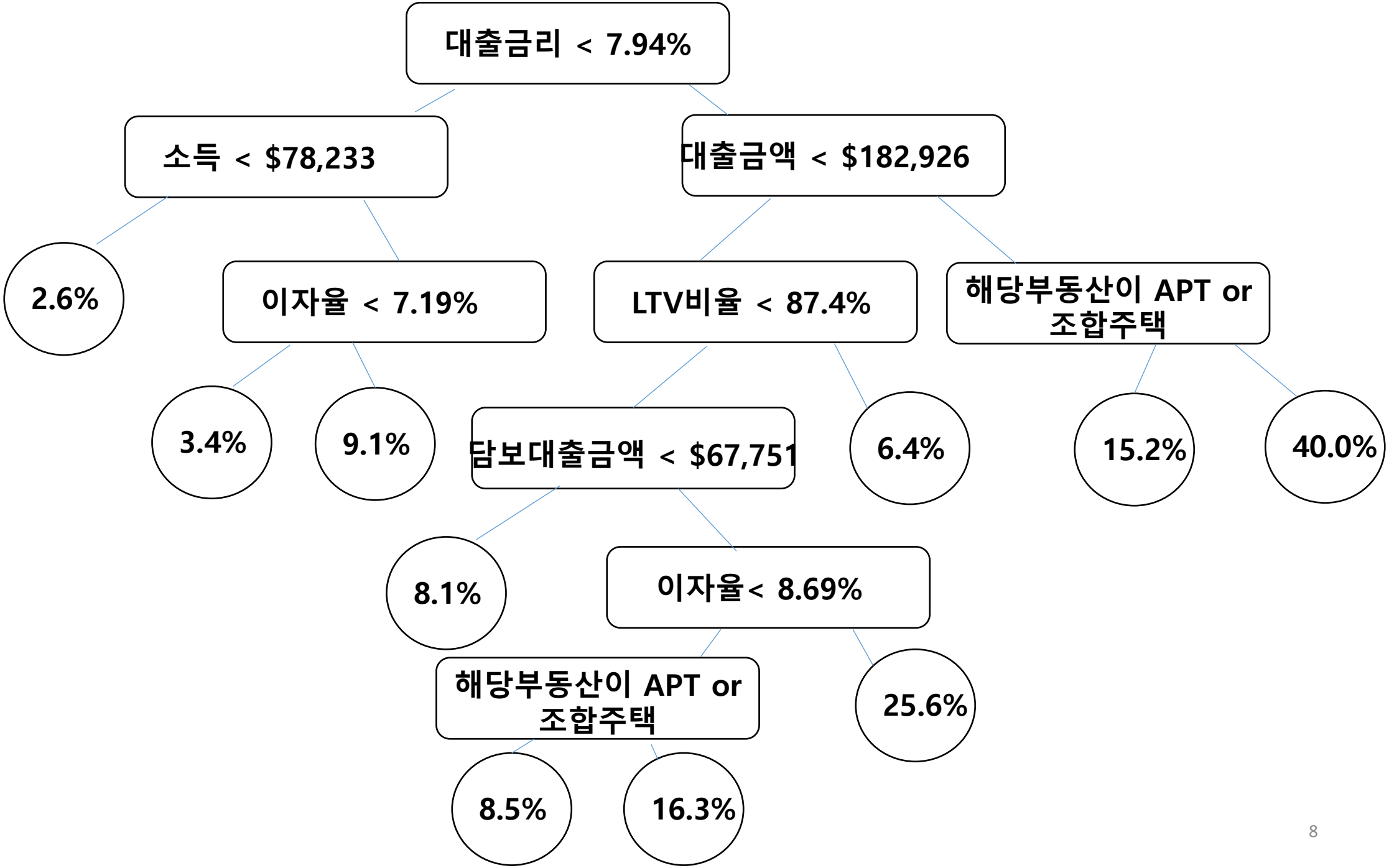



대출금리가
7.94%보다 낮은가?
Yes
No
조기상환
리스크
3.8%
조기상환
리스크
19.2%


대출금리 < 7.94%
Yes
No
소득 < 78,233 달러
Yes
No
조기상환
리스크
19.2%
조기상환
리스크
2.6%
조기상환
리스크
6.4%
(이 하위그룹을 다시 나눈다)
하위그룹 1


대출금리 < 7.94%
Yes
No
소득 < 78,233 달러
대출금액 < 182,926달러
Yes
No
Yes
No
조기상환 리스크 2.6%
조기상환 리스크 6.4%
조기상환 리스크 13.9%
조기상환 리스크 36%

하위그룹4


대출금리 < 7.94%
소득 < $78,233
대출금액 < $182,926
2.6%
이자율 < 7.19%
LTV비율 < 87.4%
해당부동산이 APT or 조합주택
3.4%
9.1%
담보대출금액 < $67,751
6.4%
15.2%
40.0%
8.1%
이자율< 8.69%
해당부동산이 APT or 조합주택
25.6%
8.5%
16.3%

만약(IF):

부동산 담보대출 금액이 67,751 달러와 같거나 그보다 더 많고 182,926 달러보다 작다.

그리고(AND):

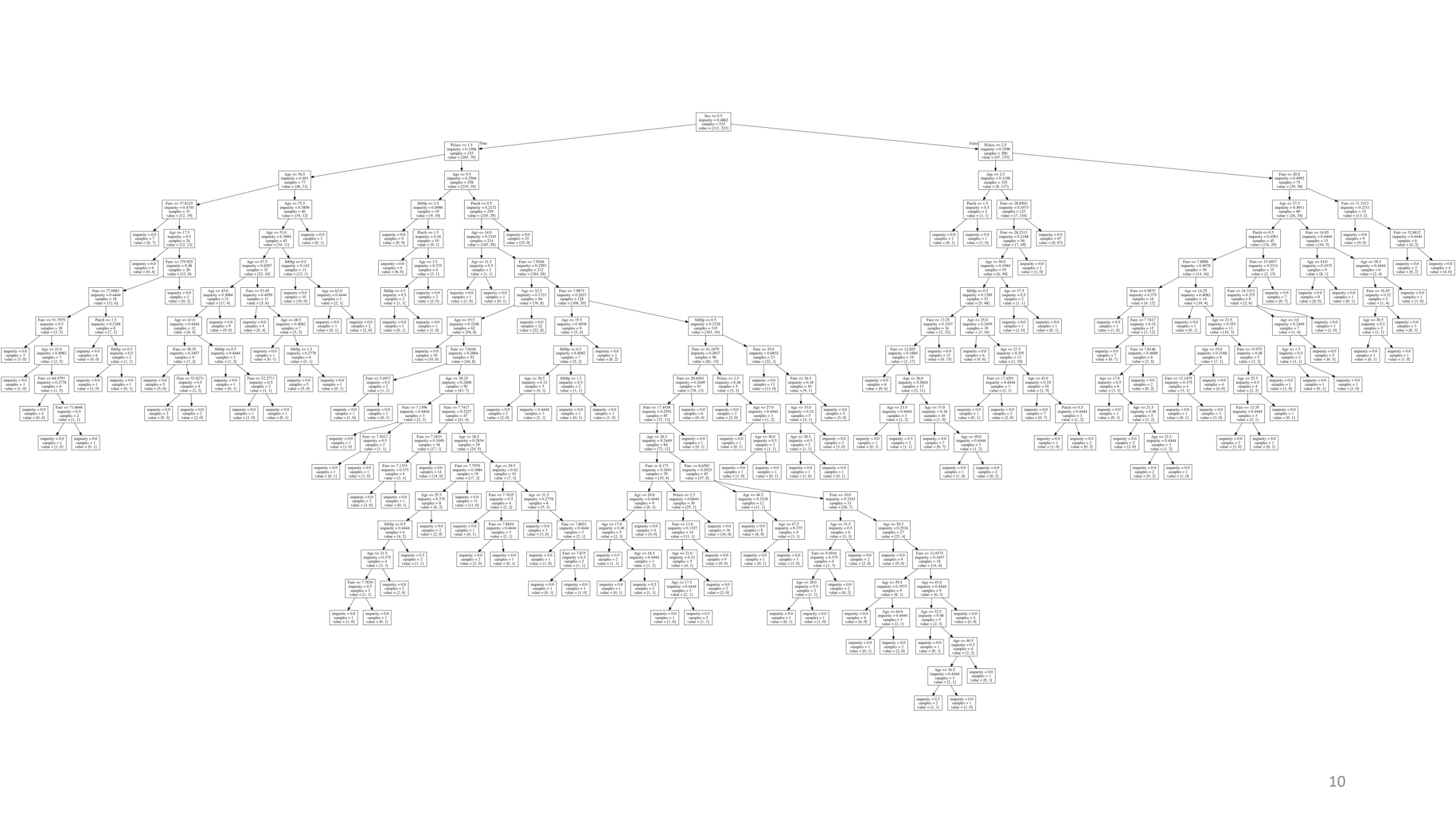
이자율이 8.69%와 같거나 그보다 더 높다.

그리고(AND):

부동산 자산가치 대비 대출금액의 비율이 87.4% 보다 작다.

그러면(THEN):

조기상환 확률은 25.6% 이다.


Sex <= 0.5
impurity = 0.4862
samples = 535
value = [312, 223]
True
False
Pclass <= 1.5
impurity = 0.3306
samples = 335
value = [265, 70]
Pclass <= 2.5
impurity = 0.3596
samples = 200
value = [47, 153]
Age <= 36.5
impurity = 0.481
samples = 77
value = [46, 31]
Age <= 9.5
impurity = 0.2566
samples = 258
value = [219, 39]
Age <= 2.5
impurity = 0.1198
samples = 125
value = [8, 117]
Fare <= 20.8
impurity = 0.4992
samples = 75
value = [39, 36]
Fare <= 37.8125
impurity = 0.4745
samples = 31
value = [12, 19]
Age <= 75.5
impurity = 0.3856
samples = 46
value = [34, 12]
SibSp <= 2.5
impurity = 0.4986
samples = 19
value = [9, 10]
Parch <= 0.5
impurity = 0.2132
samples = 239
value = [210, 29]
Parch <= 1.5
impurity = 0.5
samples = 2
value = [1, 1]
Fare <= 28.8563
impurity = 0.1073
samples = 123
value = [7, 116]
Age <= 27.5
impurity = 0.4911
samples = 60
value = [26, 34]
Fare <= 31.3312
impurity = 0.2311
samples = 15
value = [13, 2]
impurity = 0.0
samples = 7
value = [0, 7]
Age <= 17.5
impurity = 0.15
samples = 24
value = [12, 12]
Age <= 53.0
impurity = 0.3694
samples = 45
value = [34, 11]
impurity = 0.0
samples = 1
value = [0, 1]
impurity = 0.0
samples = 9
value = [0, 9]
Parch <= 1.5
impurity = 0.18
samples = 10
value = [9, 1]
Age <= 14.0
impurity = 0.2343
samples = 214
value = [185, 29]
impurity = 0.0
samples = 25
value = [25, 0]
impurity = 0.0
samples = 1
value = [0, 1]
impurity = 0.0
samples = 1
value = [1, 0]
Fare <= 28.2313
impurity = 0.2188
samples = 56
value = [7, 49]
impurity = 0.0
samples = 67
value = [0, 67]
Parch <= 0.5
impurity = 0.4583
samples = 45
value = [16, 29]
Age <= 14.85
impurity = 0.4444
samples = 15
value = [10, 5]
Fare <= 32.8812
impurity = 0.4444
samples = 6
value = [4, 2]
impurity = 0.0
samples = 2
value = [0, 2]
impurity = 0.0
samples = 4
value = [4, 0]
Fare <= 379.925
impurity = 0.48
samples = 20
value = [12, 8]
Age <= 47.5
impurity = 0.4297
samples = 32
value = [22, 10]
SibSp <= 0.5
impurity = 0.142
samples = 13
value = [12, 1]
impurity = 0.0
samples = 2
value = [0, 2]
Age <= 3.5
impurity = 0.375
samples = 4
value = [3, 1]
Age <= 11.5
impurity = 0.5
samples = 2
value = [1, 1]
Fare <= 7.9104
impurity = 0.2293
samples = 212
value = [184, 28]
Age <= 32.5
impurity = 0.1723
samples = 84
value = [76, 8]
Fare <= 7.9875
impurity = 0.2637
samples = 128
value = [108, 20]
Age <= 56.0
impurity = 0.1944
samples = 55
value = [6, 49]
impurity = 0.0
samples = 1
value = [1, 0]
Age <= 54.0
impurity = 0.1975
samples = 9
value = [8, 1]
Age <= 38.5
impurity = 0.4444
samples = 6
value = [2, 4]
impurity = 0.0
samples = 2
value = [0, 2]
impurity = 0.0
samples = 4
value = [4, 0]
Fare <= 77.0083
impurity = 0.4444
samples = 18
value = [12, 6]
Age <= 43.0
impurity = 0.3084
samples = 21
value = [17, 4]
Fare <= 53.95
impurity = 0.4959
samples = 11
value = [5, 6]
SibSp <= 0.5
impurity = 0.4082
samples = 7
value = [5, 2]
impurity = 0.0
samples = 10
value = [10, 0]
Age <= 62.0
impurity = 0.4444
samples = 3
value = [2, 1]
Age <= 19.5
impurity = 0.2248
samples = 62
value = [54, 8]
Age <= 35.5
impurity = 0.4938
samples = 9
value = [5, 4]
SibSp <= 0.5
impurity = 0.2328
samples = 119
value = [103, 16]
Fare <= 13.25
impurity = 0.1107
samples = 34
value = [2, 32]
Age <= 25.0
impurity = 0.2659
samples = 19
value = [3, 16]
SibSp <= 0.5
impurity = 0.1709
samples = 53
value = [5, 48]
Age <= 57.5
impurity = 0.5
samples = 2
value = [1, 1]
Fare <= 6.9875
impurity = 0.375
samples = 16
value = [4, 12]
Age <= 14.25
impurity = 0.4082
samples = 14
value = [10, 4]
Fare <= 15.4937
impurity = 0.2311
samples = 15
value = [2, 13]
Age <= 54.0
impurity = 0.1975
samples = 9
value = [8, 1]
Age <= 16.45
impurity = 0.32
samples = 5
value = [1, 4]
Fare <= 51.7979
impurity = 0.5
samples = 10
value = [5, 5]
Parch <= 1.5
impurity = 0.2188
samples = 8
value = [7, 1]
Age <= 41.0
impurity = 0.4444
samples = 12
value = [8, 4]
Age <= 48.5
impurity = 0.4082
samples = 7
value = [5, 2]
Age <= 20.25
impurity = 0.2408
samples = 50
value = [43, 7]
Fare <= 12.825
impurity = 0.1884
samples = 19
value = [2, 17]
Age <= 26.0
impurity = 0.2604
samples = 13
value = [2, 11]
Age <= 27.5
impurity = 0.355
samples = 13
value = [3, 10]
Fare <= 7.8896
impurity = 0.4978
samples = 30
value = [14, 16]
Age <= 23.5
impurity = 0.355
samples = 13
value = [10, 3]
Age <= 30.5
impurity = 0.5
samples = 2
value = [1, 1]
impurity = 0.0
samples = 1
value = [0, 1]
impurity = 0.0
samples = 1
value = [1, 0]
impurity = 0.0
samples = 1
value = [0, 1]
impurity = 0.0
samples = 1
value = [1, 0]
Fare <= 7.9104
impurity = 0.4938
samples = 7
value = [5, 4]
impurity = 0.0
samples = 2
value = [0, 2]
Fare <= 26.25
impurity = 0.0832
samples = 23
value = [22, 1]
Fare <= 26.5
impurity = 0.18
samples = 10
value = [9, 1]
Age <= 37.0
impurity = 0.18
samples = 10
value = [1, 9]
Fare <= 7.7417
impurity = 0.32
samples = 15
value = [3, 12]
Fare <= 7.8146
impurity = 0.4688
samples = 8
value = [3, 5]
Age <= 21.5
impurity = 0.48
samples = 5
value = [3, 2]
Fare <= 12.1479
impurity = 0.375
samples = 4
value = [3, 1]
Fare <= 15.975
impurity = 0.48
samples = 5
value = [3, 2]
Age <= 1.5
impurity = 0.5
samples = 2
value = [1, 1]
Age <= 30.5
impurity = 0.5
samples = 2
value = [1, 1]
impurity = 0.0
samples = 1
value = [0, 1]
impurity = 0.0
samples = 1
value = [1, 0]
Age <= 5.4937
impurity = 0.5
samples = 2
value = [1, 1]
Age <= 20.25
impurity = 0.2408
samples = 50
value = [43, 7]
Age <= 41.2479
impurity = 0.2637
samples = 96
value = [81, 15]
Pclass <= 2.5
impurity = 0.48
samples = 5
value = [3, 2]
impurity = 0.0
samples = 13
value = [13, 0]
Age <= 33.0
impurity = 0.32
samples = 5
value = [4, 1]
impurity = 0.0
samples = 6
value = [0, 6]
Age <= 27.5
impurity = 0.355
samples = 13
value = [3, 10]
Age <= 43.0
impurity = 0.18
samples = 10
value = [1, 9]
Age <= 23.5
impurity = 0.4444
samples = 3
value = [2, 1]
Fare <= 15.975
impurity = 0.48
samples = 5
value = [3, 2]
Age <= 10.0
impurity = 0.2449
samples = 7
value = [1, 6]
impurity = 0.0
samples = 5
value = [0, 5]
impurity = 0.0
samples = 1
value = [1, 0]
impurity = 0.0
samples = 1
value = [0, 1

인간이 의사결정을 할 때에 이성과 감정의 조화를 잘 이뤄가면서 의사결정을 할 것이라고 생각하겠지만, 인간은 결국 감정의 동물이라는 것이 임상실험을 통해 의학적으로 밝혀진 사실입니다. 즉 감정에 기반하여 잘못된 의사결정을 할 수 있는 위험이 항상 있다는 것입니다. 하지만 감정을 이기는 것이 있습니다. 바로 데이터입니다. **흘러가는 무형의 시간은 데이터로 그 모습을 남깁니다.** 즉 **데이터는 시간의 다른 이름입니다.** 이 데이터를 수집 · 축적 · 분석하여 감정을 이기는 무기로 사용함으로써 우리는 감정이 아닌 데이터에 기반하여 합리적인 의사결정을 할 수 있게 됩니다.

인공지능의 발전이 인간의 일자리를 위협할 것이라는 예측이 나오고 있습니다. 하지만, 인공지능의 발전이 인간과 인공지능 간의 경쟁을 야기한다고 보기보다는, 인공지능을 활용하는 자와 인공지능을 활용할 줄 모르는 자의 경쟁을 야기한다고 보는 것이 적절한 예측일 것입니다. 즉 일자리를 위협받는 사람은 모든 사람이 아니고 인공지능을 활용할 줄 모르는 사람입니다. 데이터 애널리틱스 분야에서도 마찬가지 예측을 할 수 있습니다. 앞으로는 데이터를 활용하는 자와 데이터를 활용할 줄 모르는 자의 경쟁이 야기될 것이고, 데이터를 활용하는 자가 절대적인 우위를 점거하게 될 것입니다. 그러므로 이제는 데이터 애널리틱스가 자신의 전공분야가 아니더 라도 기본개념 정도는 반드시 알고 있어야 하는 시대가 되었습니다.

모델학습 데이터의 구성

예시) 현 시점으로부터 과거 3년 동안의 분기별 이탈 고객의 데이터를 가지고 다음 분기 예측

2022년 1분기	2022년 2분기	2022년 3분기	2022년 4분기
2021년 1분기	2021년 2분기	2021년 3분기	2021년 4분기
2020년 1분기	2020년 2분기	2020년 3분기	2020년 4분기
2019년 1분기	2019년 2분기	2019년 3분기	2019년 4분기

모델학습 데이터 예측 분기

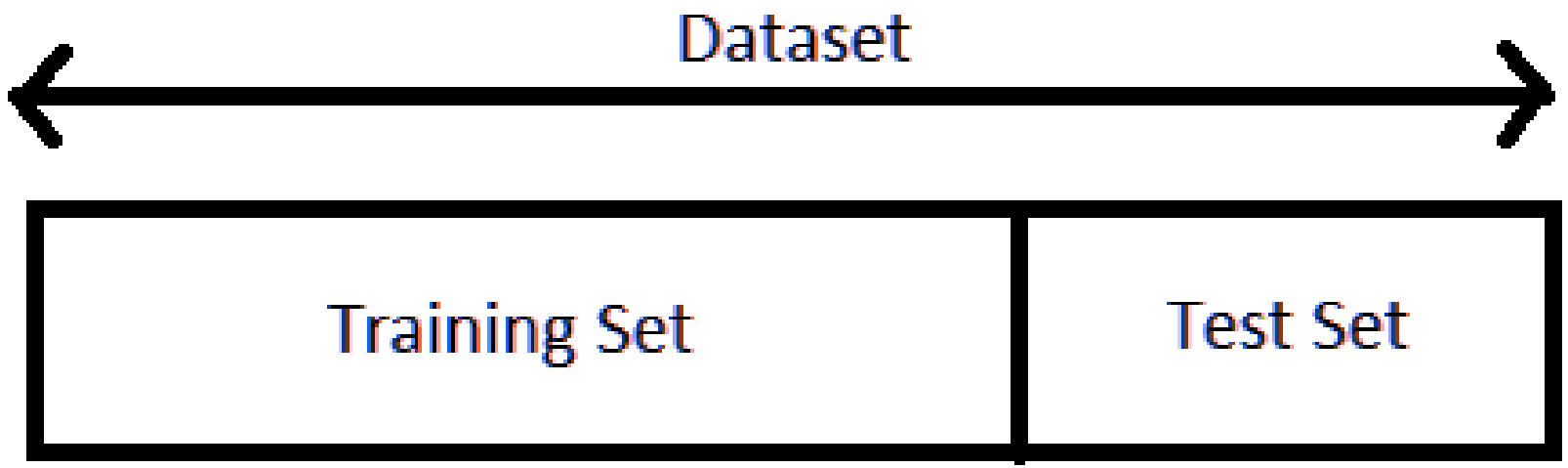
학습 모델의 평가

예시) 현 시점으로부터 과거 3년 동안의 분기별 이탈 고객의 데이터를 가지고 다음 분기 예측

2022년 1분기	2022년 2분기	2022년 3분기	2022년 4분기
2021년 1분기	2021년 2분기	2021년 3분기	2021년 4분기
2020년 1분기	2020년 2분기	2020년 3분기	2020년 4분기
2019년 1분기	2019년 2분기	2019년 3분기	2019년 4분기

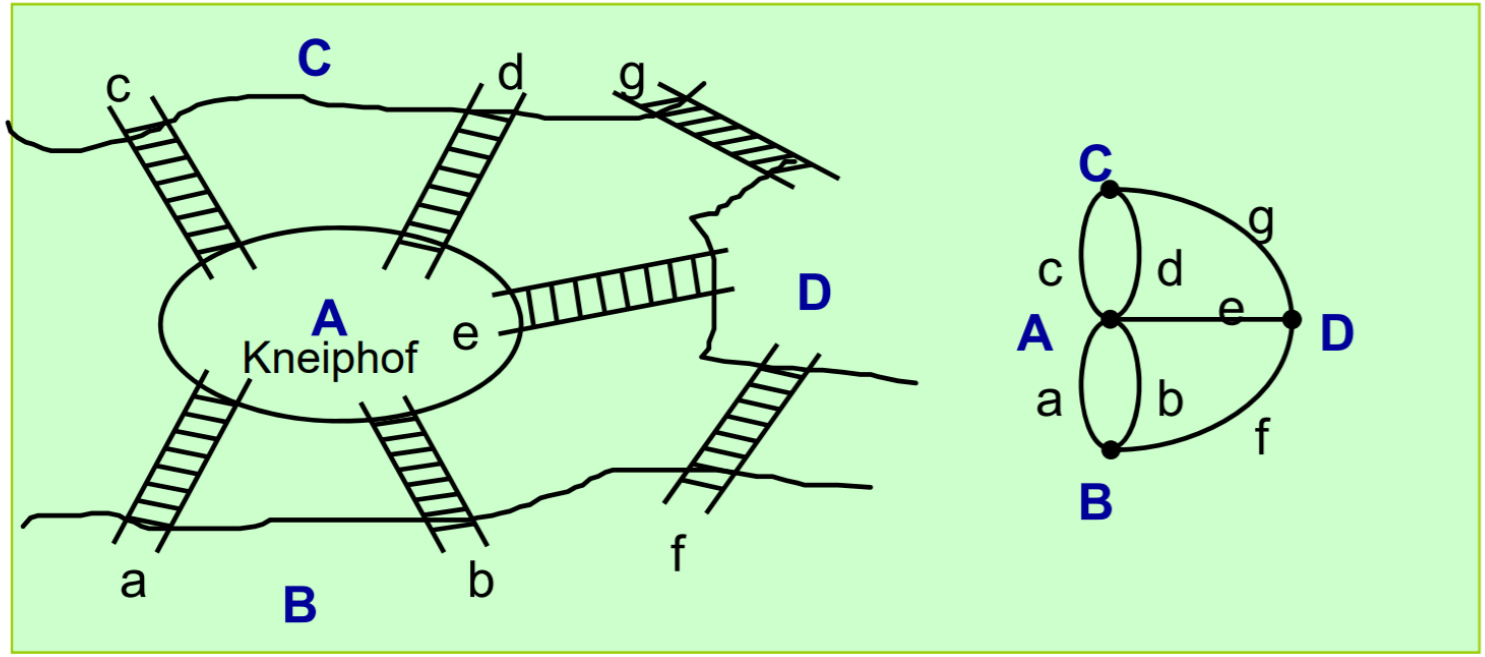
모델학습 데이터 | 모델 평가 (테스트)분기 | 예측 분기

모델 학습 데이터의 준비



Graph: 관계를 위한 자료구조

오래된 그래프 문제로 다음과 Köenigsberg 다리 문제가 있다. 이 문제는 다음과 같은 지형이 있을 때 임의의 한 곳(A,B,C,D)에서 출발하여 a부터 f까지 "모든 다리를 한번씩 건널 수 있는가?" 하는 문제이다. 자료구조의 그래프는 이러한 문제를 컴퓨터에 표현하고 알고리즘을 개발하는 분야이다.



Königsberg 다리 문제

Graph: 관계를 위한 자료구조

- 그래프의 수학적 정의

그래프 : G = (V,E) 이고, V,E는 다음과 같다.

V(G) : 정점(set of vertices)

E(G) : 간선(set of edges), 정점을 연결하는 선, V X V의 부분집합

무방향 그래프(undirected graph) – 예를 들면 쌍방통행이 가능한 도로의 지도이다.

– 정점을 연결하는 선에 방향이 없다(undirected, unordered).

즉 $(v_i,v_j) = (v_j,v_i)$이다.

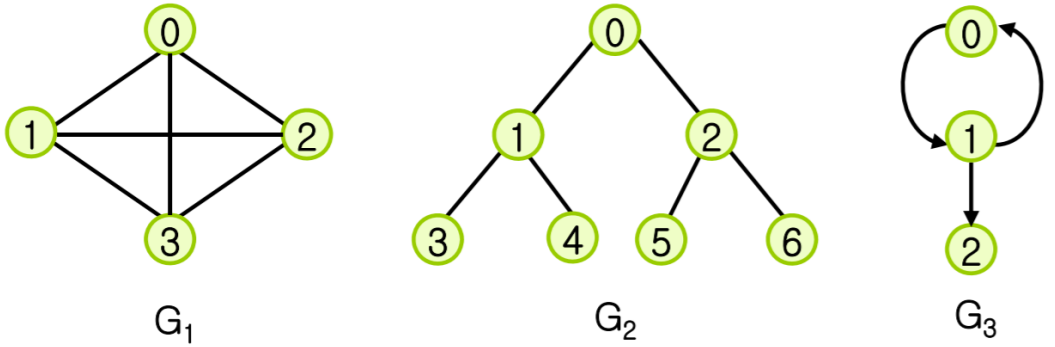
방향 그래프(directed graph) – 예를 들면 일방통행만 있는 도로의 지도이다.

– 정점을 연결하는 선에 방향이 있다(directed, ordered).

즉 $\langle v_i,v_j\rangle \neq \langle v_j,v_i\rangle$

Graph: 수학적인 표현

• **그래프의 예와 수학적인 표현** : 그래프를 표현하는 방법은 여러 가지이다. 아래 방법은 그림으로 그리는 그래프의 모습과 수학적인 기호로 표현하는 방법이다.



G_1 G_2 G_3

예 1) 그래프의 수학적인 표현

그래프 G_1

$V(G_1) = \{0,1,2,3\}$

$E(G_1) = \{(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)\}$

그래프 G_2

$V(G_2) = \{0,1,2,3,4,5,6\}$

$E(G_2) = \{(0,1),(0,2),(1,3),(1,4),(2,5),(2,6)\}$

그래프 G_3

$V(G_3) = \{0,1,2\}$

$E(G_3) = \{<0,1>,<1,0>,<1,2>\}$

Graph: 수학적인 표현 (Adjacency matrix)

- 그래프 G = (V,E), |V| = n(≥1)일 때 그래프를 이차원 행렬에 다음과 같이 저장하는 방법이다.
  adj_mat[i][j] =

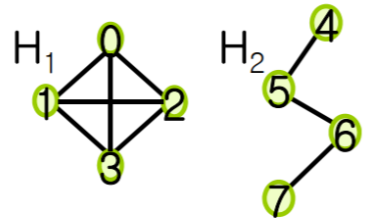
  1 : if (v_i, v_j)가 인접할 때(adjacent)

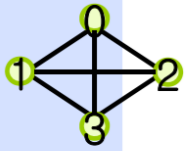
  0 : 인접하지 않을 경우

-필요한 기억장소의 크기 = 공간복잡도(space complexity) : $S(n) = n^2$

- 무방향 그래프에서는 행렬이 대각선을 중심으로 대칭(symmetric)이다.

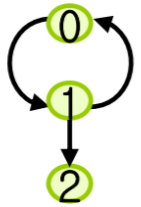
G4





G1

	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0



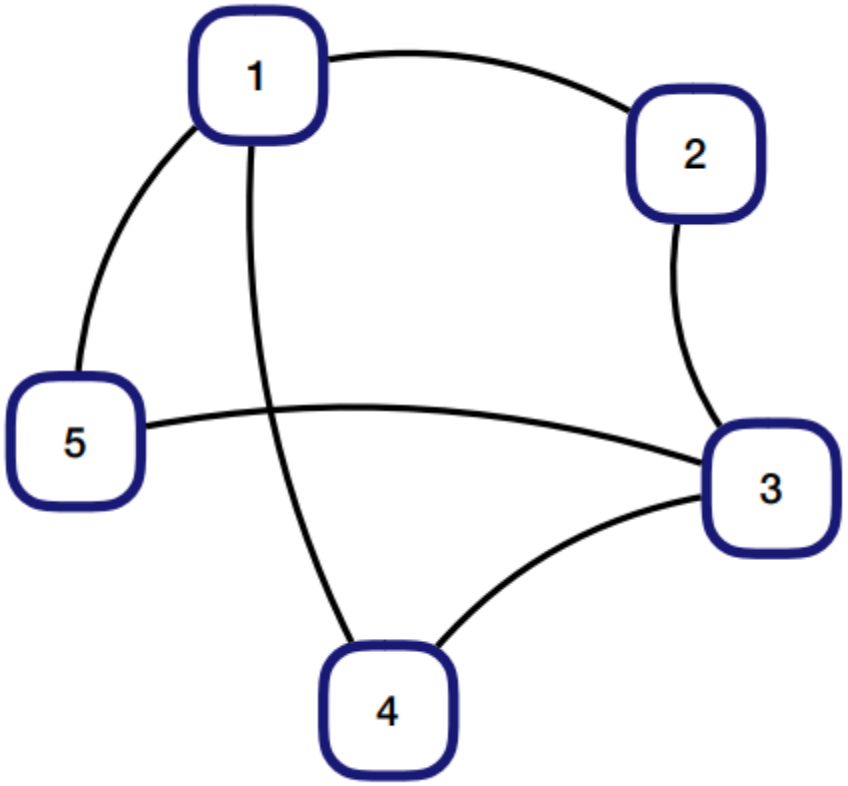
G3

	0	1	2
0	0	1	0
1	1	0	1
2	0	0	0

	0	1	2	3	4	5	6	7
0	0	1	1	1	0	0	0	0
1	1	0	1	1	0	0	0	0
2	1	1	0	1	0	0	0	0
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	1	0	0
5	0	0	0	0	1	0	1	0
6	0	0	0	0	0	1	0	1
7	0	0	0	0	0	0	1	0

인접 행렬로 표현된 그래프 G1, G3, and G4

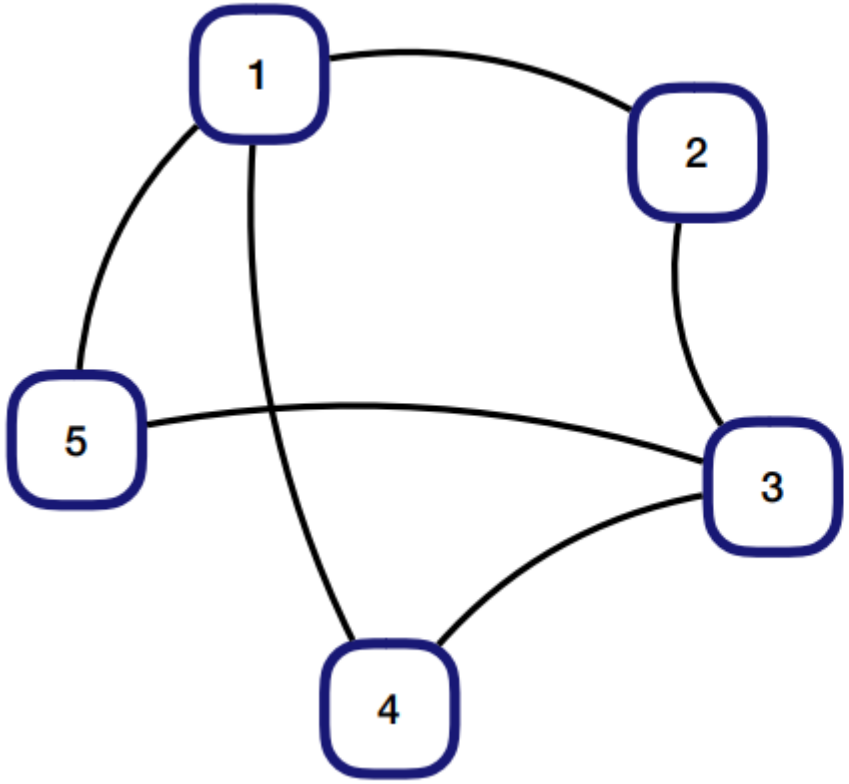
Graph: 관계를 위한 자료구조



노드(Node)와 간선(Edge)으로 구성
너와 나의 연결고리

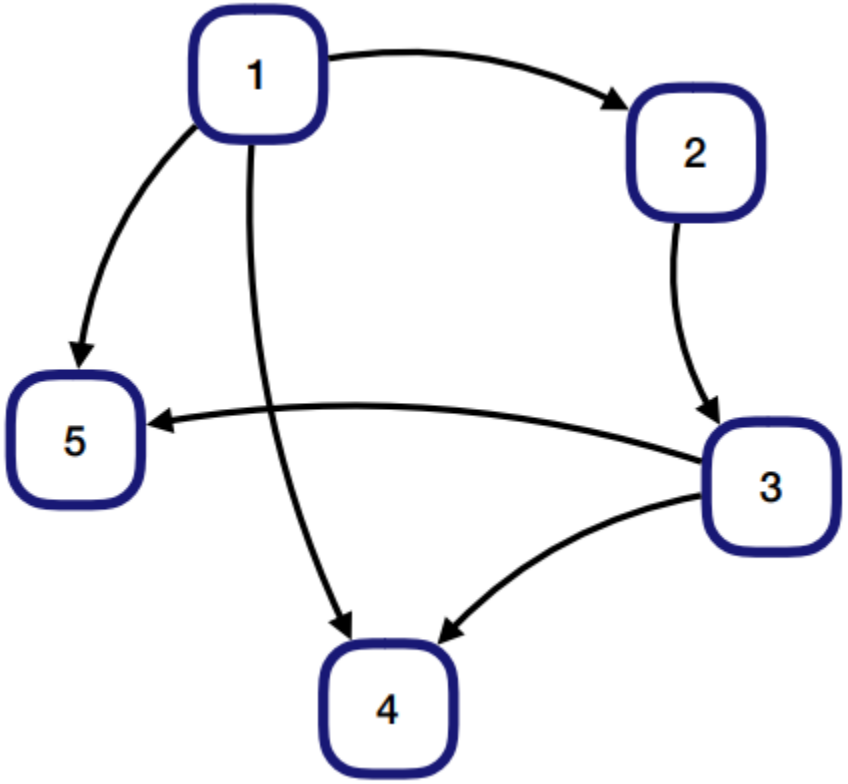
들어오는 간선 수를 **indegree**
나가는 간선 수를 **outdegree**

Graph: 저장하는 방법 / 인접 행렬



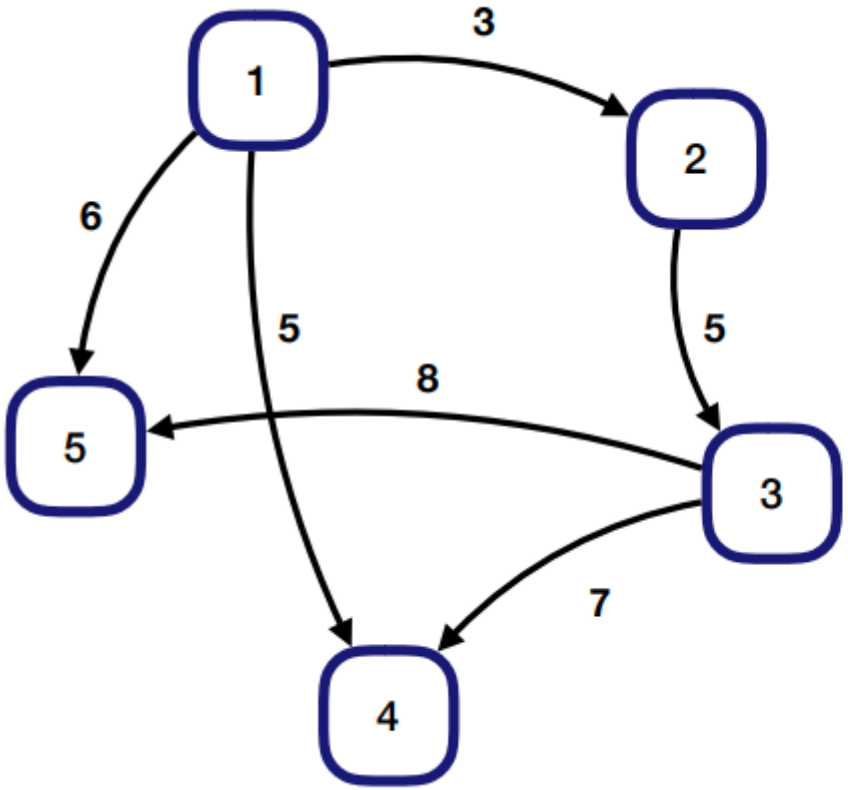
	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	0	0
3	0	1	0	1	1
4	1	0	1	0	0
5	1	0	1	0	0

Graph: 방향을 추가하면?



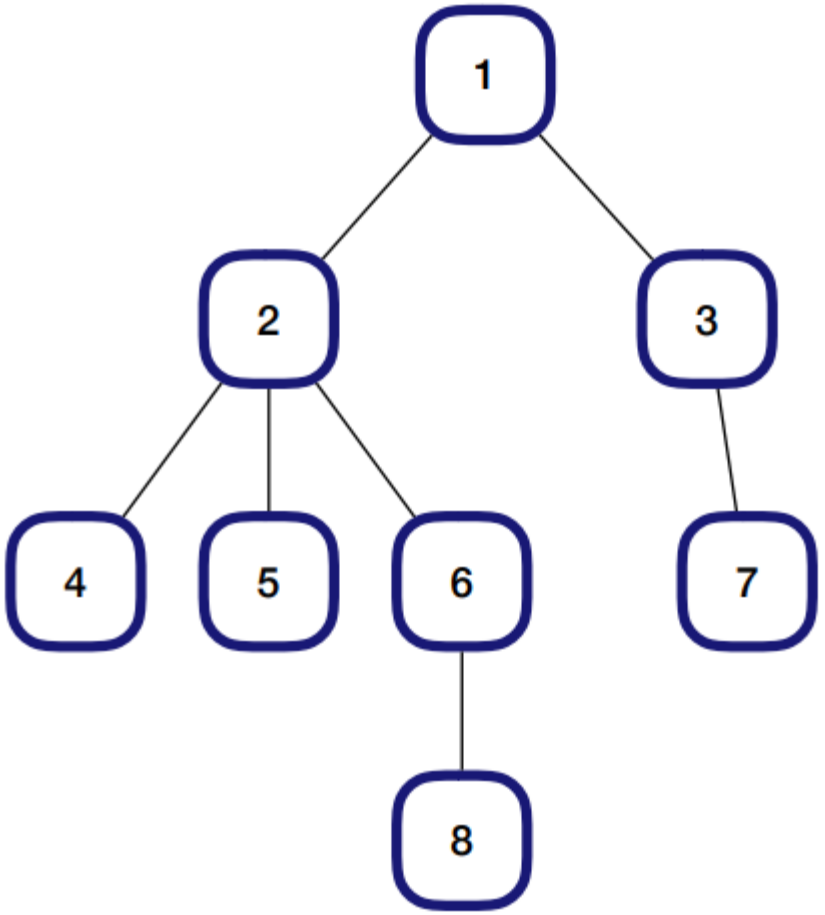
	1	2	3	4	5
1	0	1	0	1	1
2	0	0	1	0	0
3	0	0	0	1	1
4	0	0	0	0	0
5	0	0	0	0	0

Graph: 가중치를 추가하면



	1	2	3	4	5
1	0	3	0	5	6
2	0	0	5	0	0
3	0	0	0	7	8
4	0	0	0	0	0
5	0	0	0	0	0

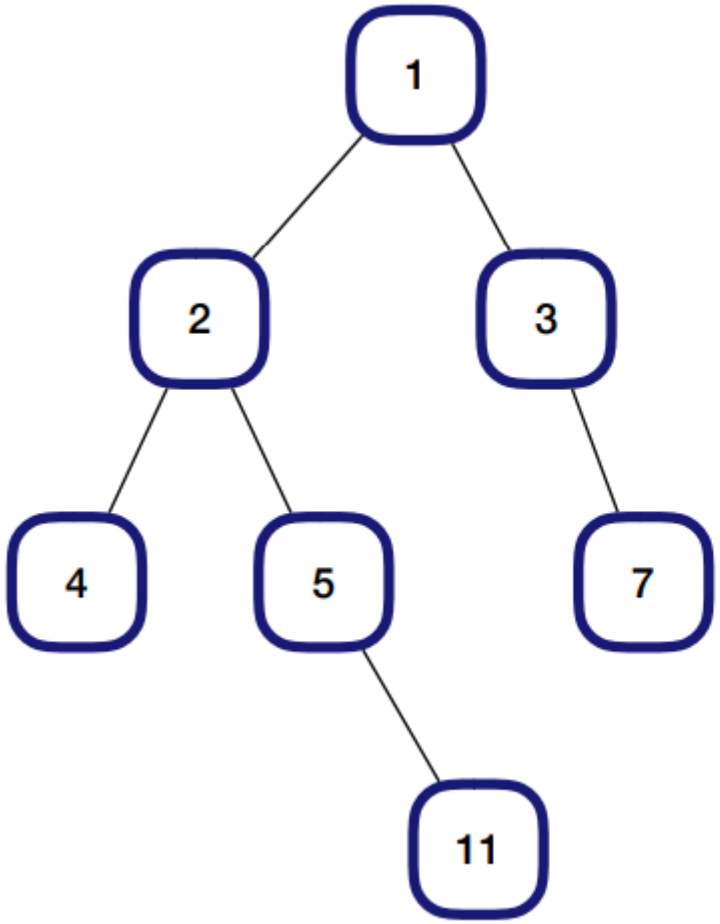
Tree: 특수한 구조를 가진 그래프



그래프이기에 똑같이 **노드(Node)와 간선(Edge)**으로 구성

상위/하위 관계를 나눌 수 있음

Binary Tree: 이진 트리



부모 노드가 X이면, 왼쪽 노드가 2X 오른쪽 노드가 2X+1
Heap과 BST에 대해 알아봅시다.

시간 복잡도: Time Complexity

연산에 따라 속도는 모두 같을까? + 연산과 * 연산은 같을까?

모든 연산을 Counting 할 수 있을까? break 등의 생략은?

최악의 경우와 최선의 경우? 수열의 정렬에서 최악과 최선?

Counting에 따라 실행시간은? 서버와 컴퓨터의 속도 차?

컴퓨터 과학(Computer Science) 에서 알고리즘은 어떠한 문제를 해결하기 위한 방법이고,
어떠한 문제를 해결 하기 위한 방법은 다양하기 때문에 방법(알고리즘) 간에 효율성을 비교하기 위해 빅오(big-O) 표기법을 보통 가장 많이 사용한다.

빅오 표기법은 보통 알고리즘의 시간 복잡도와 공간 복잡도를 나타내는데 주로 사용 된다.

(시간 복잡도는 알고리즘의 시간 효율성을 의미하고, 공간 복잡도는 알고리즘의 공간(메모리) 효율성을 의미한다.)

그런데 시간과 공간 복잡도를 나타내는 방법으로는 점근 표기법이라고 해서

빅오(Big-O), 빅오메가(big-Ω),빅세타(big-Θ) 표기법이 있다.

빅오 표기법은 알고리즘 효율성을 상한선 기준으로 표기하기 때문이다.
(알고리즘 효율성은 값이 클수록 즉, 그래프가 위로 향할수록 비효율적임을 의미한다.)

시간 복잡도: Big-O Notation

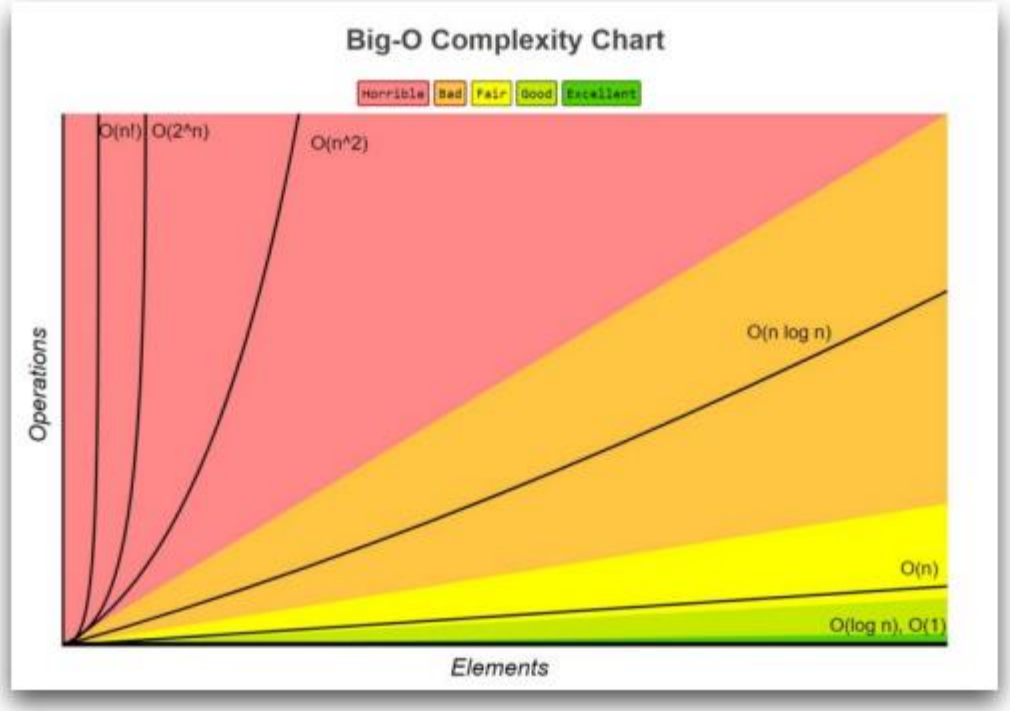
대충 계산하자 => 점근적 표기법

입력이 N일때, 연산 횟수가 최악이 2N^2 + 4N 이라면??

N이 무한대로 커질 때, 증가에 미치는 영향은 가장 큰 항만 필요하다!

O(N^2)

시간 복잡도: Big-O Notation



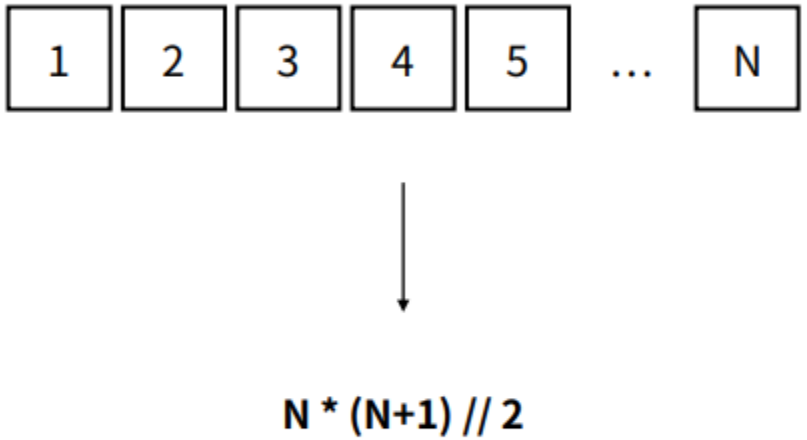
Slow $O(n!) > O(2^N) > O(N^2) > O(NlogN) > O(N) > O(\sqrt{N}) > O(logN) > O(1)$ Fast

(지수함수 > 다항함수 > 선형함수 > 로그함수 > 상수함수)

시간 복잡도: O(1) 예시

Q. 1부터 N까지 합을 구하시오.

```
def sum_N(N):
    return N*(N+1)//2
```

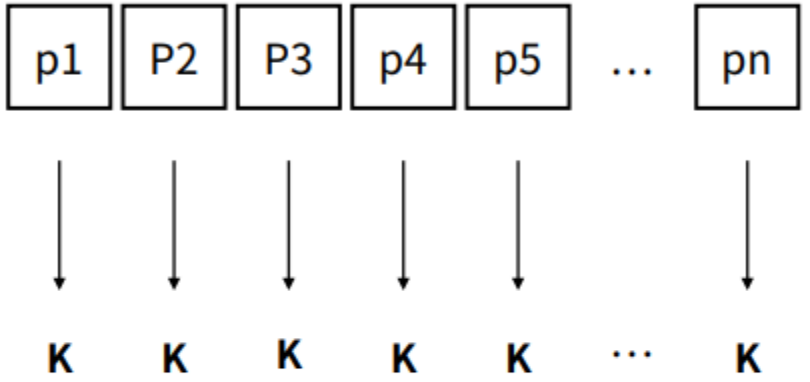


단순한 수식으로 표현 가능

시간 복잡도: O(N) 예시

Q. 길이 N 수열에서 수 K 찾기 (sum, min, max도 유사)

```
def search(lst, N, K):
    for i in lst:
        if i == lst : return True
    return False
```



최악의 경우 N번을 돌려봐야함
확률적으로는 N/2번

시간 복잡도: O(logN) 예시

계속 길이가 절반으로 줄어듬 $N/2/\cdots/2 \simeq 1 \Rightarrow N \simeq 2^k \Rightarrow k = log_2 N$

2라는 상수를 때면… $O(log_2 N) = O(logN)$

```
def binary_search(lst, N, K):
    lo, hi = 0, N-1
    while lo <= hi :
        mid = (lo + hi) // 2
        if lst[mid] == K: return True
        if lst[mid] > K: hi = mid-1
        else : lo = mid+1
    return False
```

Big O 예제

1. O(1) : 스택에서 Push, Pop
2. O(log n) : 이진트리
3. O(n) : for 문
4. O(n log n) : 퀵 정렬(quick sort), 병합정렬(merge sort), 힙 정렬(heap Sort)
5. O(n**2): 이중 for 문, 삽입정렬(insertion sort), 거품정렬(bubble sort), 선택정렬(selection sort)
6. O(2**n) : 피보나치 수열

유클리드호제법: 최대공약수와 최소공배수

최대공약수란 공통적인 약수 중 최댓값
최대공약수가 1이면 **서로소**

최소공배수는 다음 공식으로 구할 수 있음
공통된 배수 중 최솟값

$$LCM(A, B) = A \times B / GCD(A, B)$$

우리는 GCD만 잘 구하면 LCM은 O(1)에 구할 수 있다

유클리드호제법: 최대공약수와 최소공배수

혹시 유클리드 호제법 아시는 분??

$$GCD(A, B) = GCD(B, A \,\%\, B)$$

```
def gcd(a, b):
    return b if a%b==0 else gcd(b, a%b)
```

강사 소개

정 준 수 / Ph.D (heinem@naver.com)

- 前) 삼성전자 연구원
- 前) 삼성의료원 (삼성생명과학연구소)
- 前) 삼성SDS (정보기술연구소)
- 現) (사)한국인공지능협회, AI, 머신러닝 강의
- 現) 한국소프트웨어산업협회, AI, 머신러닝 강의
- 現) 서울디지털재단, AI 자문위원
- 現) 한성대학교 교수(겸)
- 전문분야: 시각 모델링, 머신러닝(ML), RPA
- https://github.com/JSJeong-me/