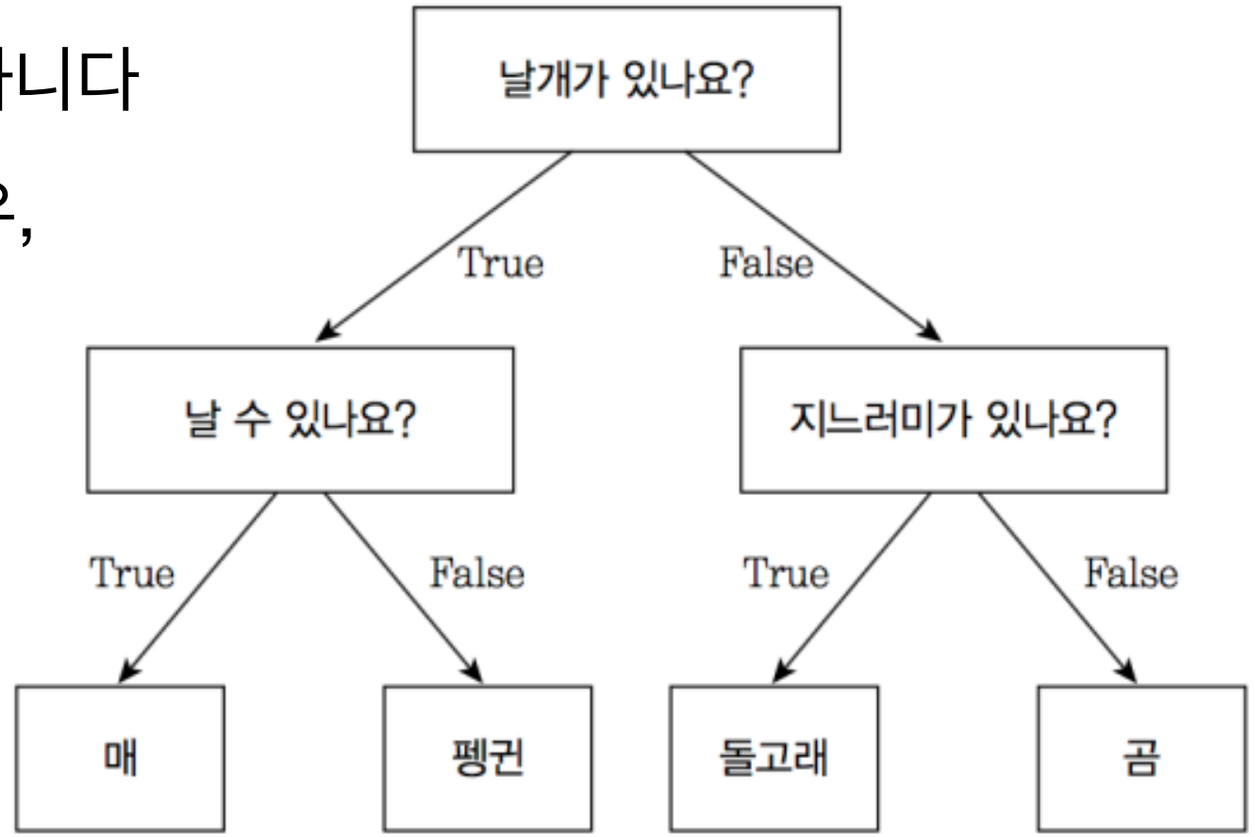
Python을 활용한 데이터 분석 강의

Supervised Learning

HYUNDAI MOBIS

Decision Trees

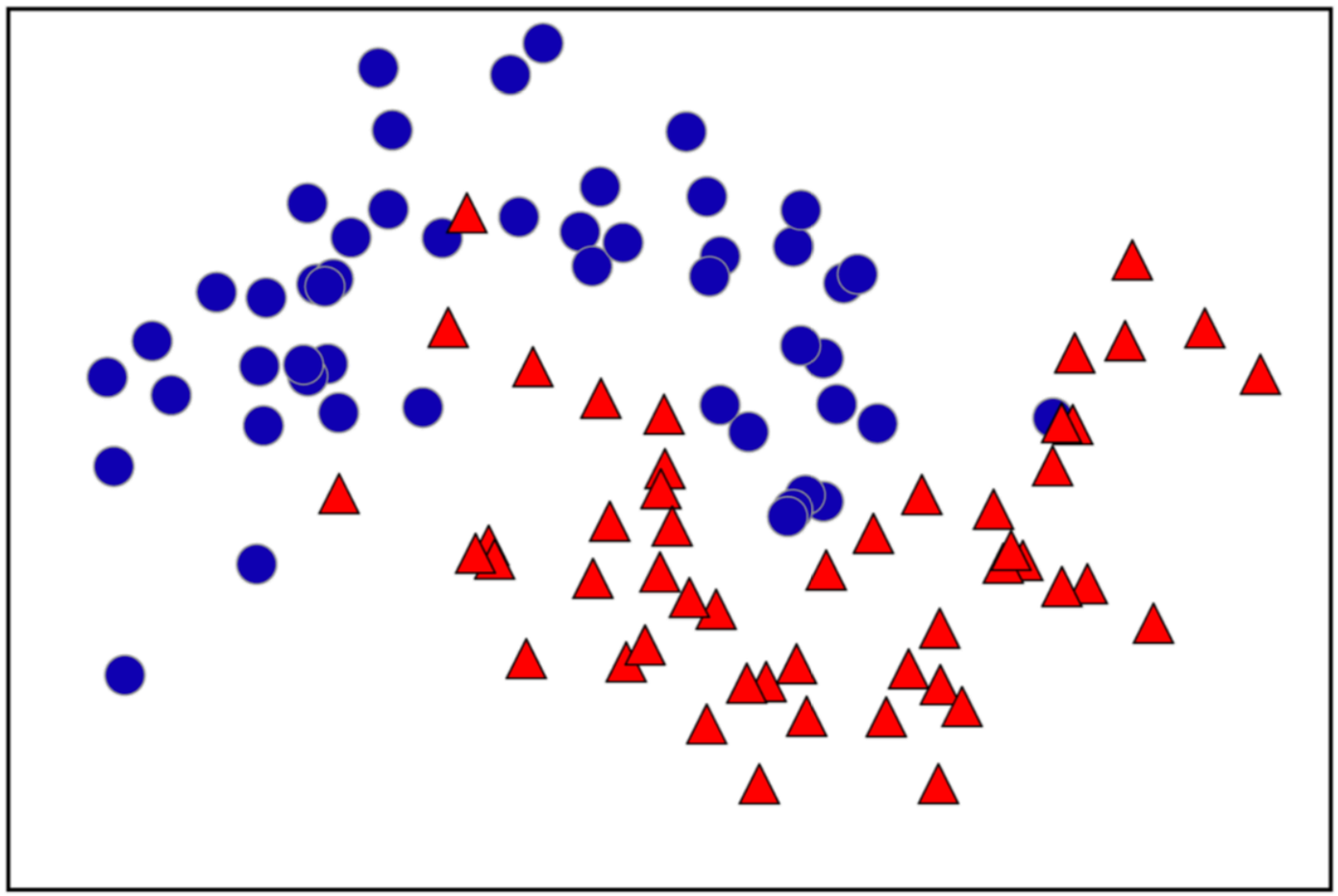
- if/else의 계층구조를 쌓아서 결론에 다다르도록
  - 스무고개랑 비슷!
- 항상 yes/no question은 아니다
- continuous feature일 경우,
  - ex. 나이가 35세 이상인가?



four classes of animals
using three features “has feathers”, “can fly”, and “has fins”

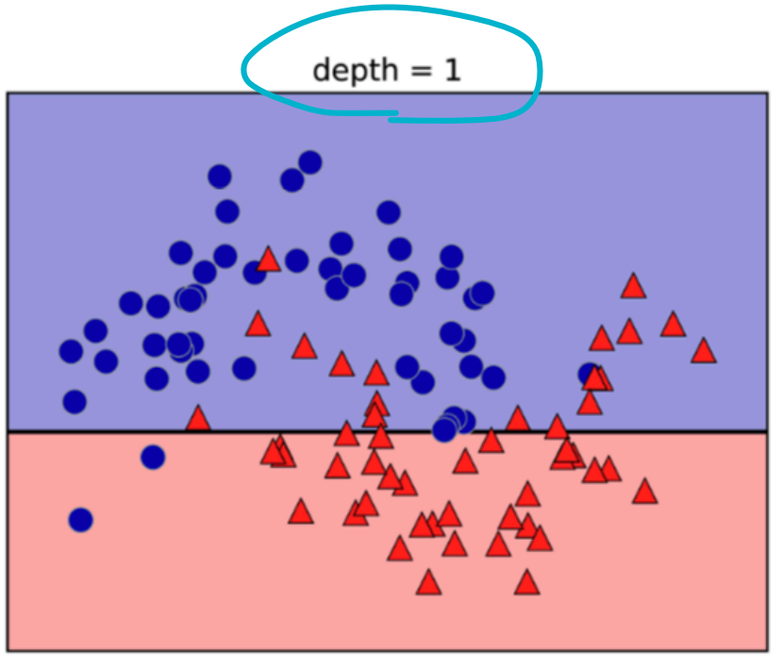
Decision Trees

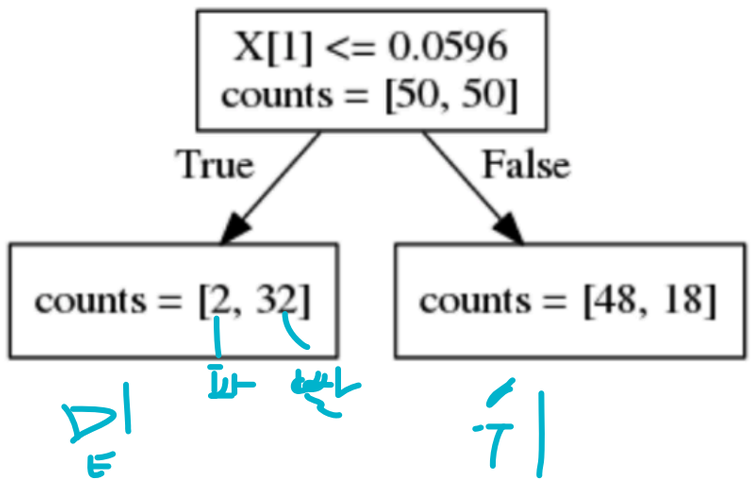
- Two_moons
  - two half-moon shapes(75 data points)



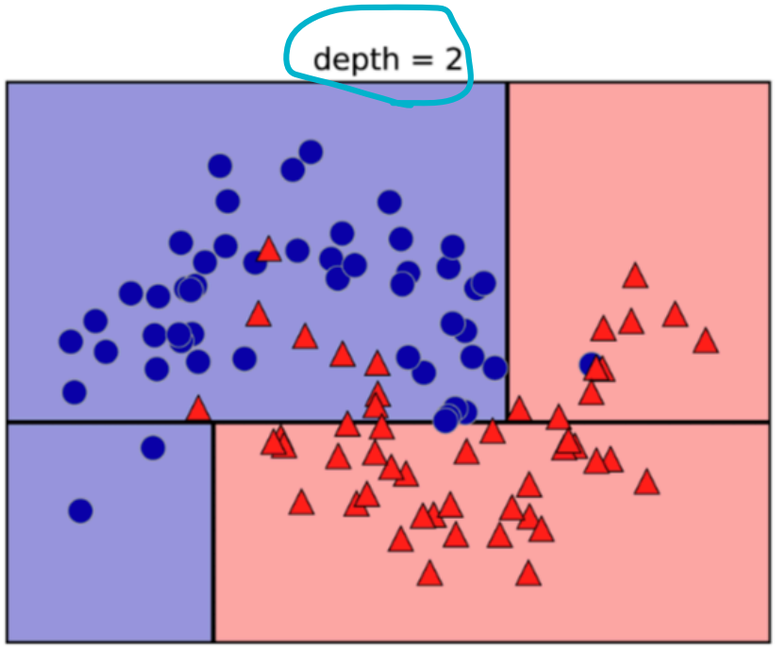
Decision Trees

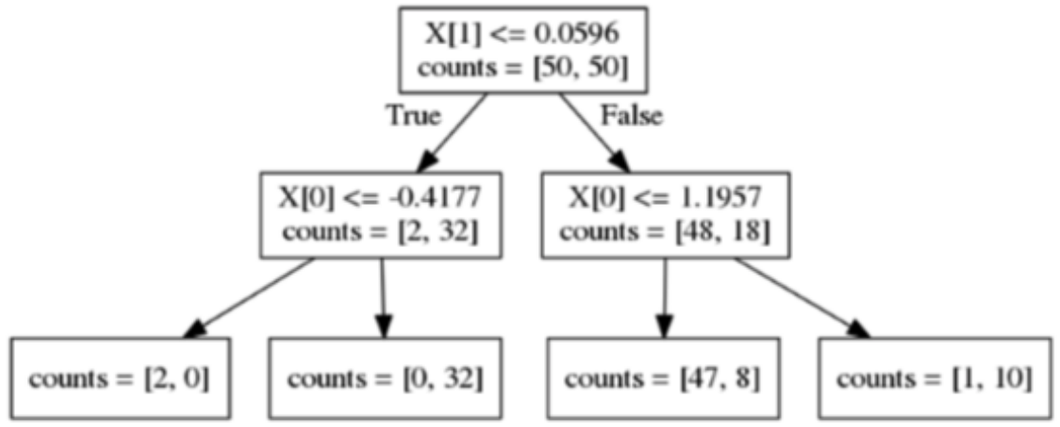
- 첫번째 test
  - X[1] <= 0.0596
  - class를 가장 잘 나누는 첫번째 test를 생성

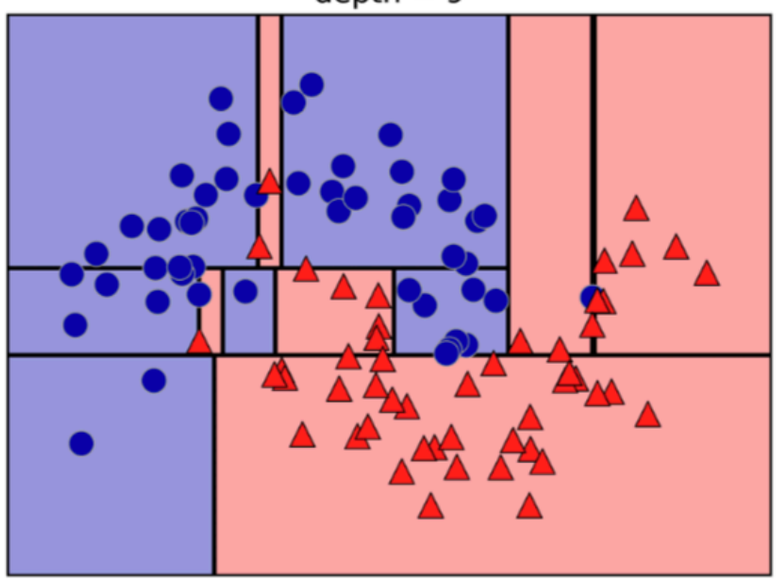


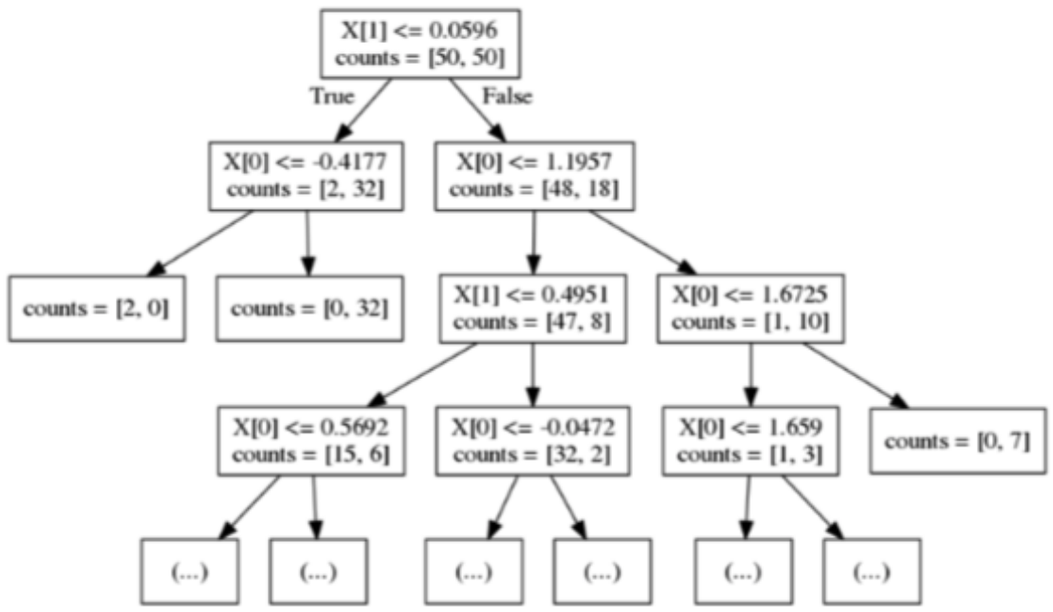


Decision Trees





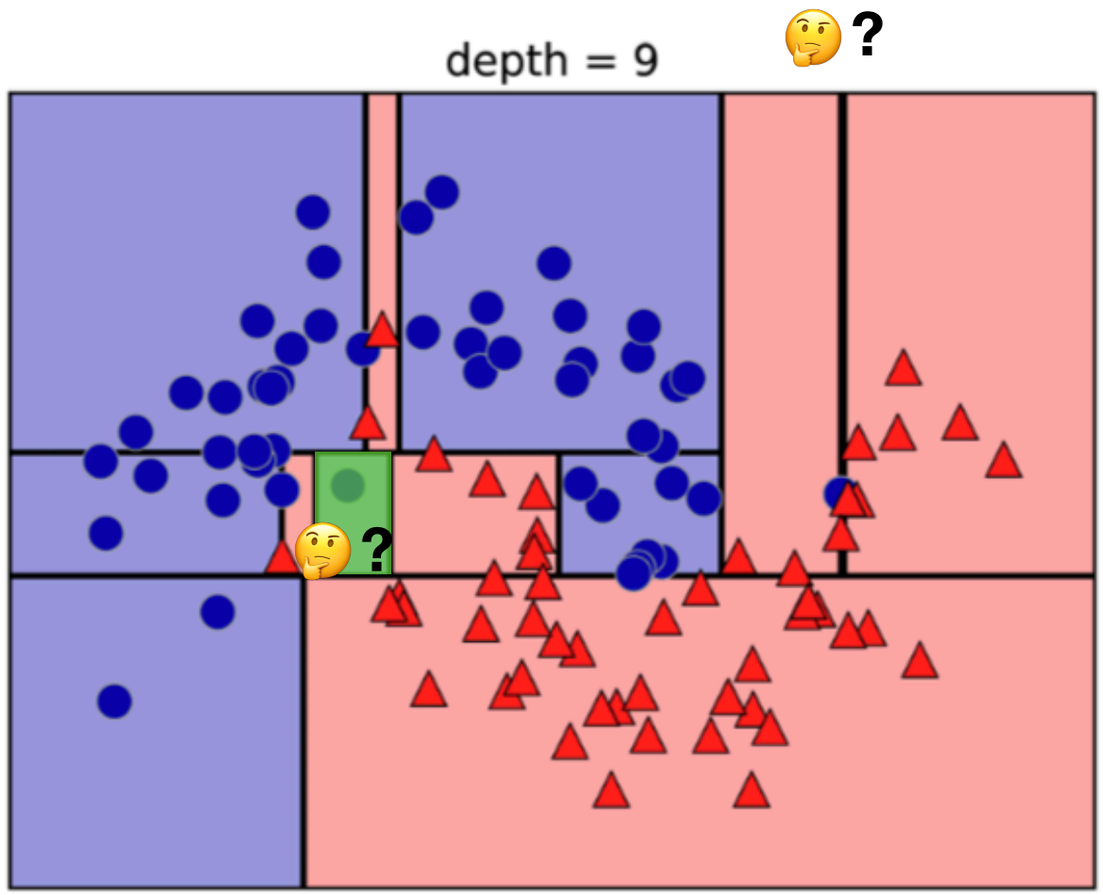




HYUNDAI MOBIS

Decision Trees

- decision tree를 정교하게 짜다보면 모델이 매우 복잡해지고 이로 인해 overfitting 문제가 발생



Decision Trees

- overfitting을 예방하기 위해 나무의 가지를 친다!

  - pre-pruning
    - 나무의 최대 깊이(max_depth)를 설정
    - 잎의 최대 숫자(max_leaf_nodes)를 설정
    - 계속 split하기 위한 결과 데이터의 최소 숫자(min_samples_leaf)를 설정

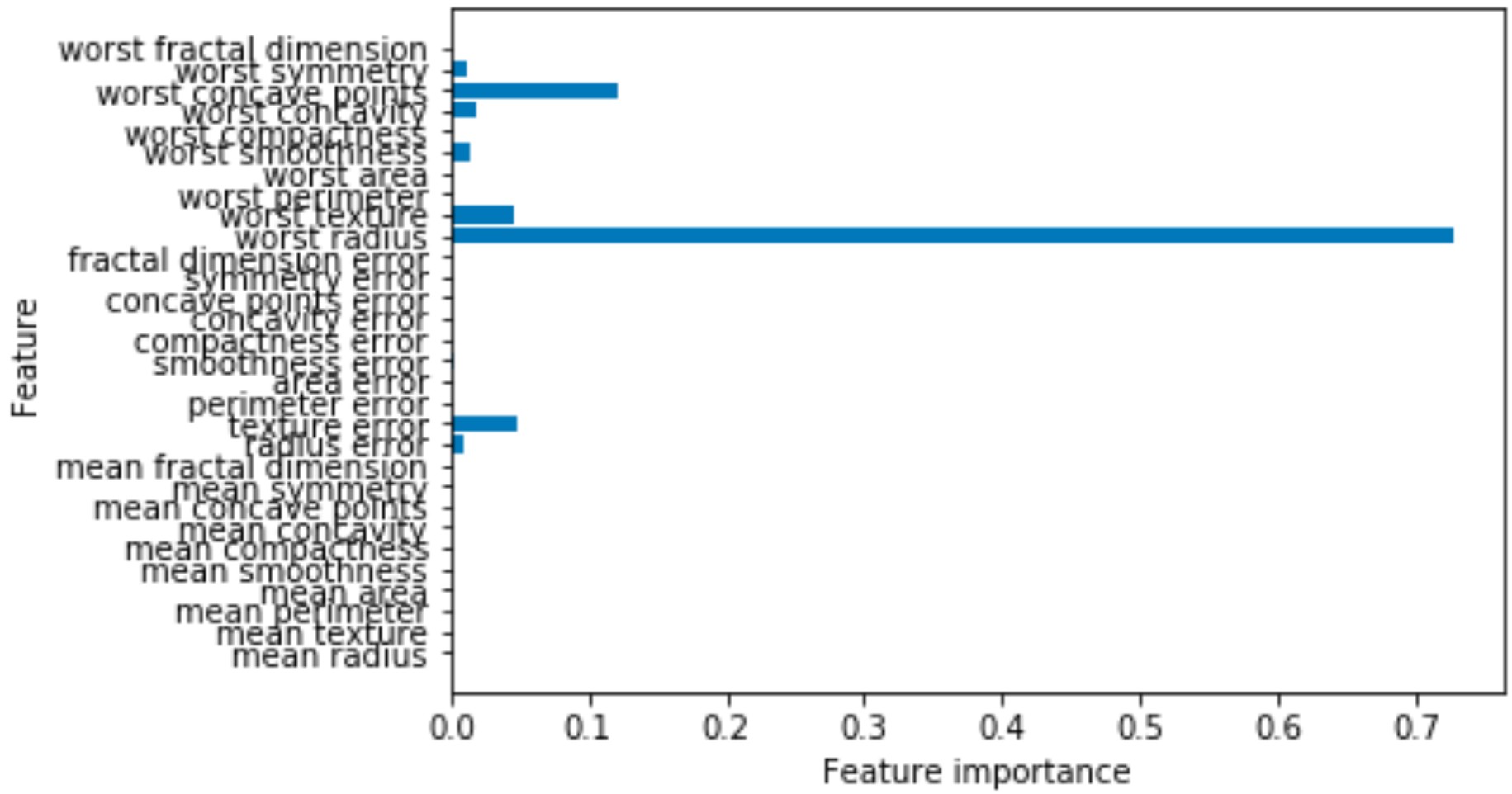
  - post-pruning(pruning)

Decision Trees

- Feature importance
  - tree를 요약해서 보여 줌
  - decision을 만들기 위해서 각 feature가 얼마나 중요한지를 도출
  - 0과 1 사이의 값을 가짐
    - 0: “not used at all”
    - 1: “perfectly predicts the target”
  - feature importance가 낮다고 해서 그 feature가 uninformative하다는 의미가 아니고, 그저 tree에 의해 선택되지 않았다는 의미
  - coefficient랑 달리 “worst radius”가 악성인지 양성인지를 알려줄 수는 없음

HYUNDAI MOBIS

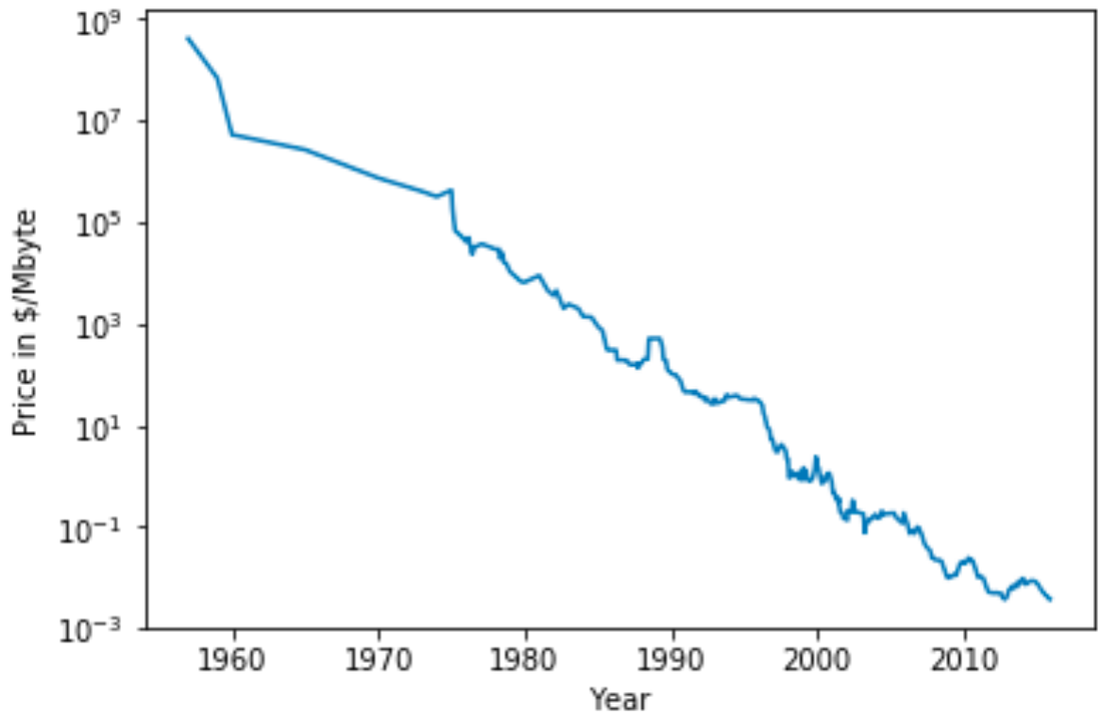
Decision Trees



HYUNDAI MOBIS

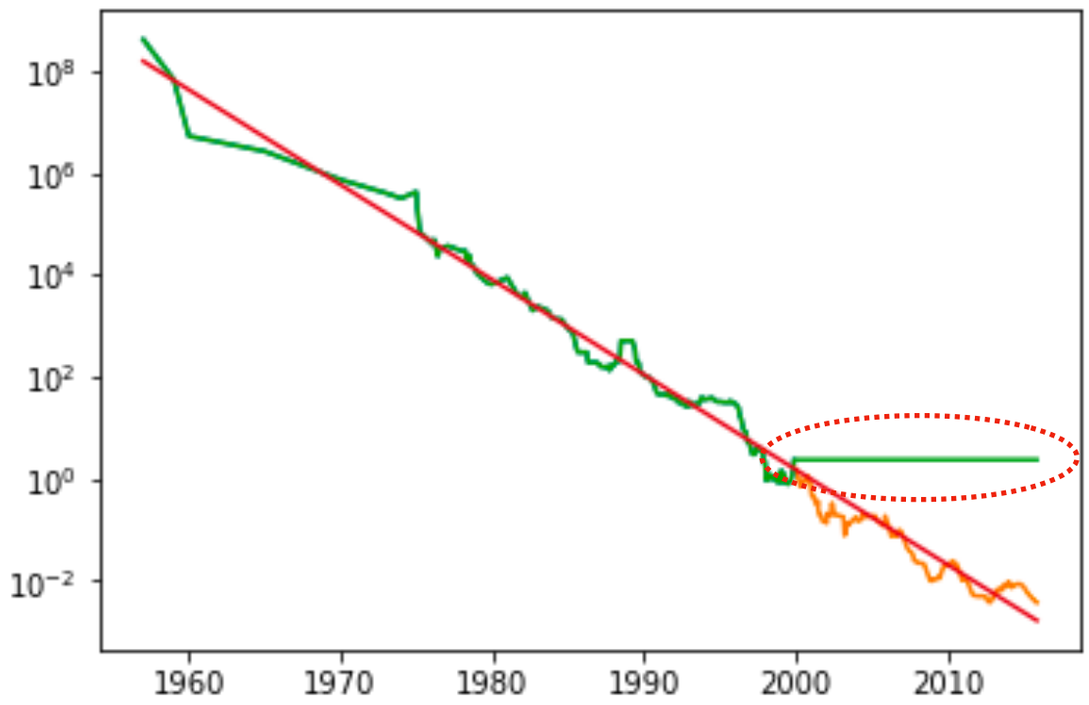
Decision Trees

- DecisionTreeRegressor : tree-based models for regression
  - training 데이터의 범위 밖에서는 예측할 수 없음
  - ‘RAM prices’ 예시
    - DecisionTreeRegressor VS. LinearRegression



Decision Trees

- DecisionTreeRegressor : tree-based models for regression
  - **training 데이터의 범위 밖에서는 예측할 수 없음**
  - ‘RAM prices’ 예시
    - DecisionTreeRegressor VS. LinearRegression



HYUNDAI MOBIS

Decision Trees

- Pros
  - visualization이 쉬움
  - 모델의 구조를 직관적으로 이해할수 있음
  - data의 규모와 상관없이 같은 알고리즘으로 구현

- Cons
  - overfitting 되기가 쉬움 = generalization이 어려움
  - 그래서 ensemble!

Ensembles of Decision Trees

- Ensembles : 여러 개의 ML 모델들을 결합시키는 방법

  - random forests

  - gradient boosted decision trees

HYUNDAI MOBIS

Random Forests

- 조금씩 다른 decision tree들을 결합
  - 각각의 tree들은 overfit 되어 있을 수 있지만 각각 다른 방향으로 overfit 되어 있을 것
  - 이것들을 평균내면 overfitting을 줄일 수 있음
  - 서로 다른 tree들을 만들기 위해서는?
    - selecting data points
    - selecting the features

Random Forests

- Building RF
  - tree의 개수(n_estimators)
  - bootstrap sample : 샘플을 복원추출
  - select subset of features(max_features)
    - high max_features? low max_features?

Random Forests

- Pros
  - decision tree의 장점을 그대로
  - 보고할 때 좋음 😀

- Cons
  - text data 같은 경우에는 잘 작동하지 않음(linear model is better)
  - memory를 많이 필요로 하고, 상대적으로 training이 느림
    - tree의 개수는 많을 수록 좋지만, memory와 시간의 문제

Gradient Boosted Regression Trees

- 여러 개의 decision tree를 결합
- RF와는 달리 이전 tree의 결함을 수정해 가면서 새로운 tree를 만듦
  - no randomization
  - pre-pruning을 강하게
- 보통 얕은 depth를 가짐(1 ~ 5)
- RF보다 parameter setting에 좀 더 민감한 경향이 있음
  - 얼마나 결함을 강하게 수정할 것인가(모델을 더 복잡하게 할 것인가)
    - learning_rate

Gradient Boosted Regression Trees

- Pros
  - 강력한 supervised learning 모델 중 하나
- Cons
  - 파라미터 튜닝에 공을 많이 들여야 함

- learning_rate가 낮다는 것은 비슷한 수준의 tree를 많이 만든다는 의미
- RF와 달리, tree의 개수를 높이면 overfitting 문제가 발생
- 보통 tree 개수를 memory, 시간에 맞추고, learnig_rate을 조정