

기계학습을 활용한 게임 어뷰징 검출

김정주

PyCon APAC 2016

발표자 소개

김정주 (hajeOl@naver.com)

전: 게임 개발

- NHN / NPLUTO
- 3D 엔진 / 게임 클라이언트 개발

현: 게임 데이터 수집 / 분석

- Webzen NPlay
- 로그 포워더, Pandas, Scikit-Learn,
PySpark



이 발표는

- 기계학습에 대한 기본 지식이 있는 분들을 대상
- 파이썬을 활용한 데이터 분석과 기계학습 사례를 공유
- 개발과 서비스에 기계학습을 도입하는 계기가 되었으면 합니다

시작 동기

- 게임 어뷰징 제재를
- 유저 신고 / GM 모니터링 / 패턴 찾기로는 한계
- 사람의 개입이 최소화된 어뷰징 탐지 시스템을 만들자

게임 어뷰징이란?

- “기획적으로 의도하지 않은 방식으로 게임 정보를 대량 획득하거나 도움을 주는 행위” 😈
- 사례
 - 서비스 구현상의 헛점을 이용한 플레이
 - 해킹 툴을 사용한 비정상 플레이
 - 전체 채팅창에 도배로 광고

통계와 탐색적 데이터 분석

우선, 통계

- 통계는 풍부하지 못한 데이터와 컴퓨팅 파워의 환경에서 발전
- 통계 학자들은 데이터/계산을 줄이는 방법을 연구
- 열악한 환경에서 만들어졌기에, 적은 데이터에서도 가치를 발견할 수 있음
- 기본적인 통계 지식은 개발, 기획, 서비스 등에 큰 도움이 됨

탐색적 데이터 분석

- 데이터에 숨어있는 정보를,
- 다양한 각도로 요약, 시각화 해보며 찾는 과정 
- 처음 접하는 데이터는 이 과정부터
- 자체 시스템(WzDat) 개발해 활용 
 - Jupyter + Utility + Dashboard
 - <https://github.com/haje01/wzdat>
 - <http://www.pycon.kr/2014/program/14>

사례1

간단한 통계적 아이디어로 스패머 검출

상황

- 신규 오픈한 게임의 채팅창이 게임 아이템 광고글로 가득 😢
- 해당 계정을 제재해도 바로 새 계정으로 광고 계속
- 빠른 제재가 필요하여, 기계학습을 진행하기에는 시간이 부족

채팅을 이용한 스팸 (Spam)

- 게임 내에서 전역 채팅으로 머니/아이템 판매 광고
- 어뷰저는 프로그램적 판별을 막기위해 메시지를 난독화

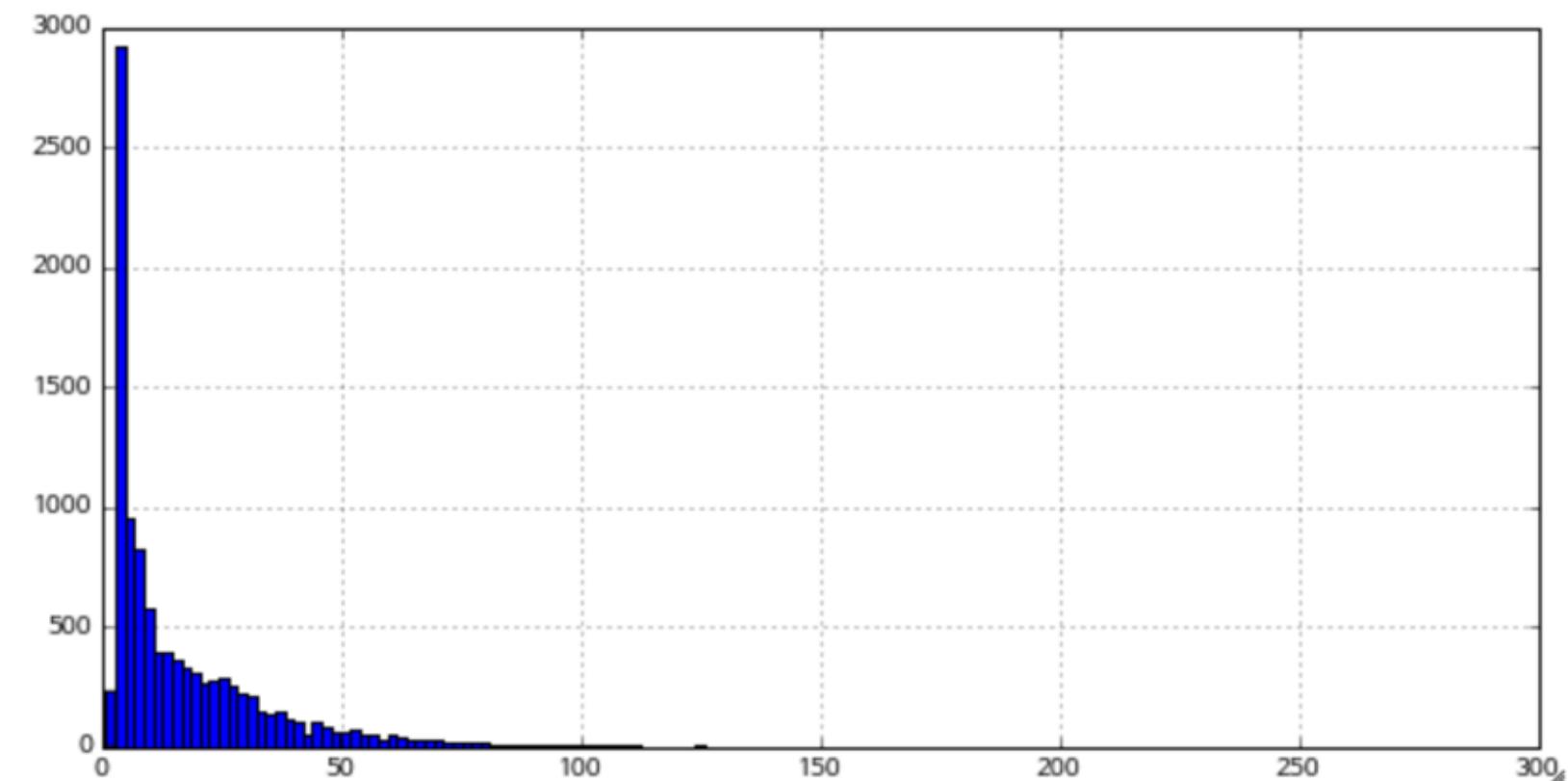
```
3 W.IGAME4S.C.0M,IGAME4S.C.0M,SELL gold,100gold only =2us d,enough stock,deliver after u pay
3 W.M M O U U.c o m Sell che@p Gold 24/7 Live Chat!100@=2.91@sd~~QQ97500581~skype:gamerassis
tant~~9
Get Disc0unted WC0INS RELOAD_CARD and the CHEA@PEST G0ld at ww®,SE@G®,C®M --960
3`W.MM OU U.c O m Sell c h e a p Gold 24/7 Live Chat!100G 4.6E Ⓜ R~~QQ97500581~skype:gamerass
istant6
SE@G®,C®M 10@=@SD0.22/RM1.01/SGD0.31/AUD0.32/RUB17.19/7.88Baht/3,097Rp/PHP11.1 --497
GOLDCE0.C^0^M-----GOLDCE0.C^0^M-----GOLDCE0.C^0^M seller gold 100G is 2 u_s_d-
3 W.M M O U U.c o m Sell che@p Gold 24/7 Live Chat!100@=2.91@sd~~QQ97500581~skype:gamerassis
tant~~1
3`W.MM OU U.c O m Sell c h e a p Gold 24/7 Live Chat!100G 4.6E Ⓜ R~~QQ97500581~skype:gamerass
istant0
3 W.M M O U U.c o m Sell che@p Gold 24/7 Live Chat!100@=2.91@sd~~QQ97500581~skype:gamerassis
tant~~0
3 W.M M O U U.c o m Sell che@p Gold 24/7 Live Chat!100@=2.91@sd~~QQ97500581~skype:gamerassis
tant~~4
3`W.MM OU U.c O m Sell c h e a p Gold 24/7 Live Chat!100G 4.6E Ⓜ R~~QQ97500581~skype:gamerass
istant5
3 W.M M O U U.c o m Sell che@p Gold 24/7 Live Chat!100@=2.91@sd~~QQ97500581~skype:gamerassis
```

스팸머 검출

- 다양한 방법이 가능하겠으나,
- 자연어 처리나 기계학습같은 고급 접근보다,
- 간단한 통계적 아이디어로 시도

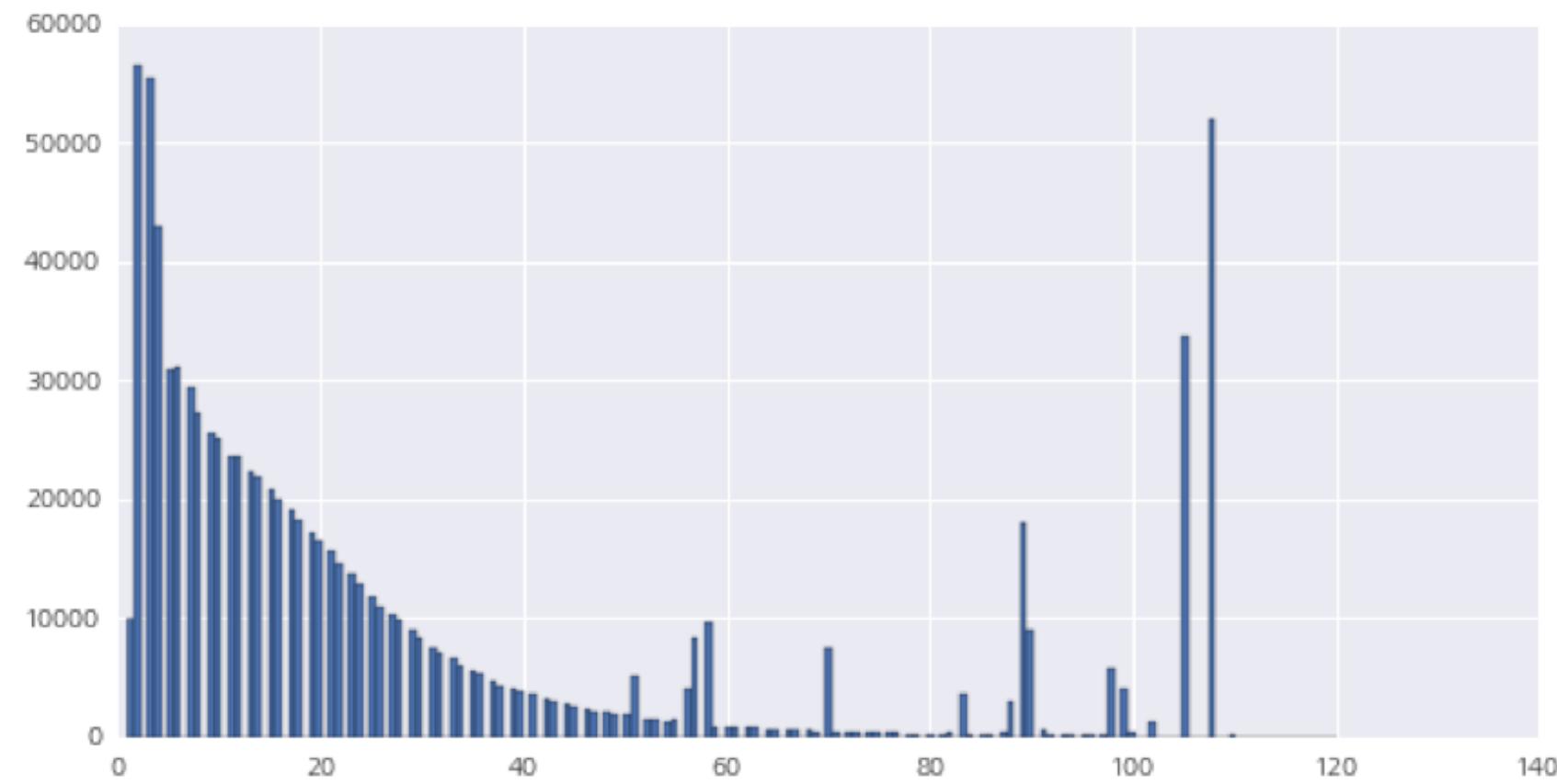
온라인 채팅 메시지 길이의 분포

- 일반적으로 로그 정규분포를 따른다고 알려져있다.
- 아래는 NPS Chat Corpus의 메시지 길이 분포



게임 내 채팅 메시지 길이 분포

- 온라인 채팅과 비슷하나 좀 더 두꺼운 경향
- 특정 길이 메시지가 됨(?) → 스팸으로 확인



아이디어

- 일반 유저: 메시지 길이가 다양하고, 빈도가 많지 않음
- 스파머: 메시지 길이가 다양하지 않고, 빈도는 높음
- 즉, 어떤 유저의 채팅 빈도가 높고 길이가 다양하지 않으면 스파머

간단한 검출 공식

$$spam_ratio = \frac{chat_cnt}{msg_size_cnt}$$

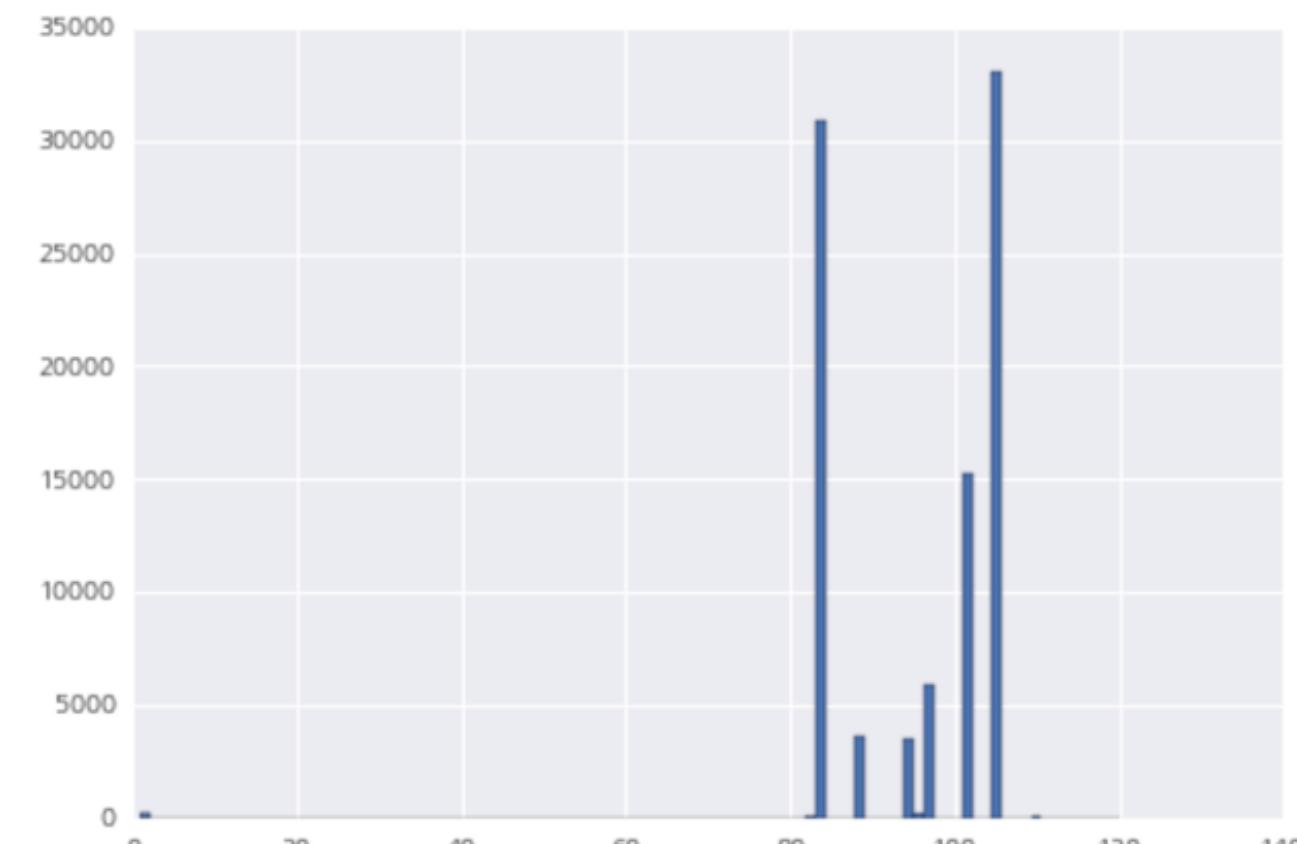
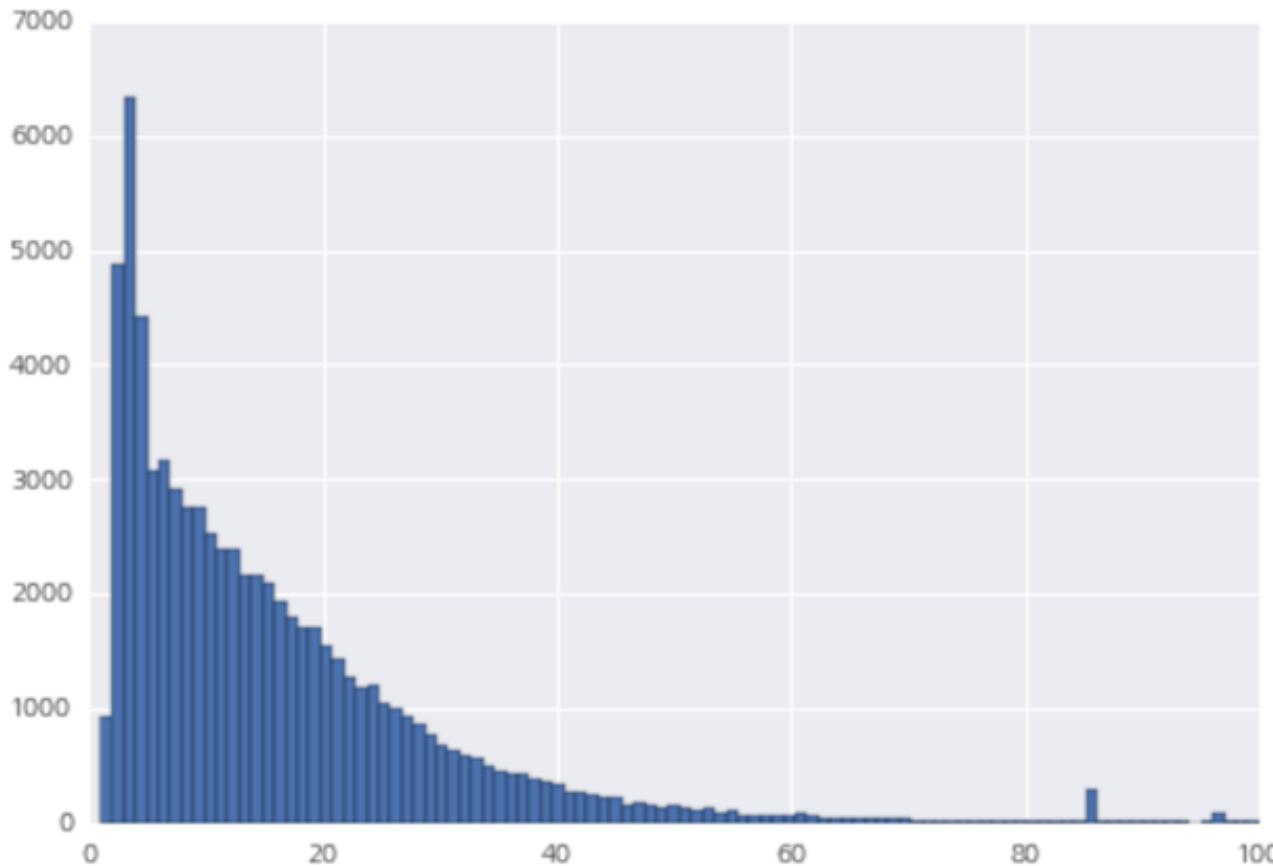
- 유저 별 채팅의 횟수 / 메시지 길이 종류 수
- 비슷한 길이의 채팅 메시지를 자주 보낼 수록 값이 커짐

분류

- `spam_ratio`가 기준 값 이상인 것을 스파머로 간주
- 기준 값 결정은 휴리스틱하게...
- 대충 설정 후, 분류된 캐릭터의 메시지 확인으로 값 조절

분류 후 메시지 길이 분포

- 빈도가 높은 특정 길이의 메시지(= 스팸)가 분리되었음



결과 적용

- 구현이 간단했지만, 오탐의 가능성 있음
 - 기준 값을 높게 잡아 신뢰도를 높임
- 이 결과를 가지고 제재

개선 방향

- 기준 값 결정을 좀 더 과학적인 방법으로
- 자연어 처리 기술(NLP) 도입
 - 단어별 빈도(Ziff's Law)와 중요도(TF-IDF) 고려
- 기계학습 알고리즘 적용

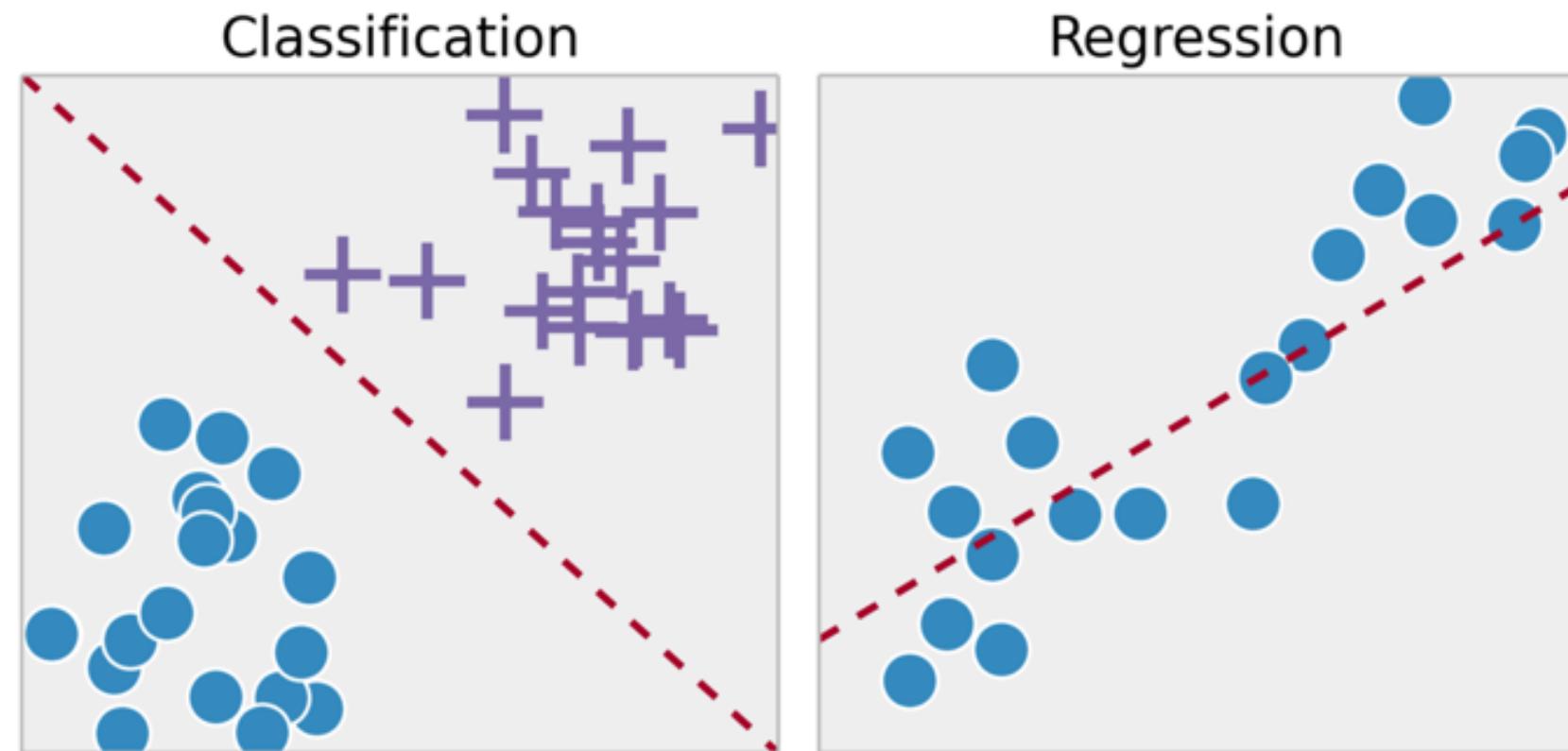
기계학습 소개

기계학습을 쓰는 이유

- 적은 노력으로 괜찮은 결과물
- 다양한 문제에 대한 일반적인 솔루션
- 다수의 특성(피쳐)을 동시에 고려할 수 있다
- 데이터 변동에 강함(강건성)

분류와 회귀

- 기계학습은 크게 분류
(Classification)과 회귀
(Regression)로 나뉨
- 분류 - 종류를 예측 하는 것
- 회귀 - 연속된 값을 예측 하는 것
- 어뷰징 검출은 분류에 속함



지도 학습과 자율 학습

- 지도 학습(*Supervised Learning*)
 - 기존 경험에 의해 분류된 샘플 데이터가 있을 때
- 자율 학습(*Unsupervised Learning*)
 - 분류된 샘플 데이터가 없을 때
 - 대부분의 데이터는 적절히 분류되어 있지 않다 → 풀어야할 문제

기계학습 알고리즘들

- 기본
 - 리니어/로지스틱 리그레션(Linear/Logistic Regression)
 - 결정 트리(Decision Tree)
- 고급
 - 랜덤 포레스트(Random Forest)
 - SVM(Support Vector Machine)
 - 인공 신경망(Neural Network)

알고리즘의 선택은?

- 일반적으로 고급 알고리즘은 더 복잡한 모델 학습 가능
- 그러나, 고급 알고리즘이 무조건 좋은 것은 아님
- 학습의 결과를 사람이 이해하기에는 기본 알고리즘이 좋다

예측에 대한 평가

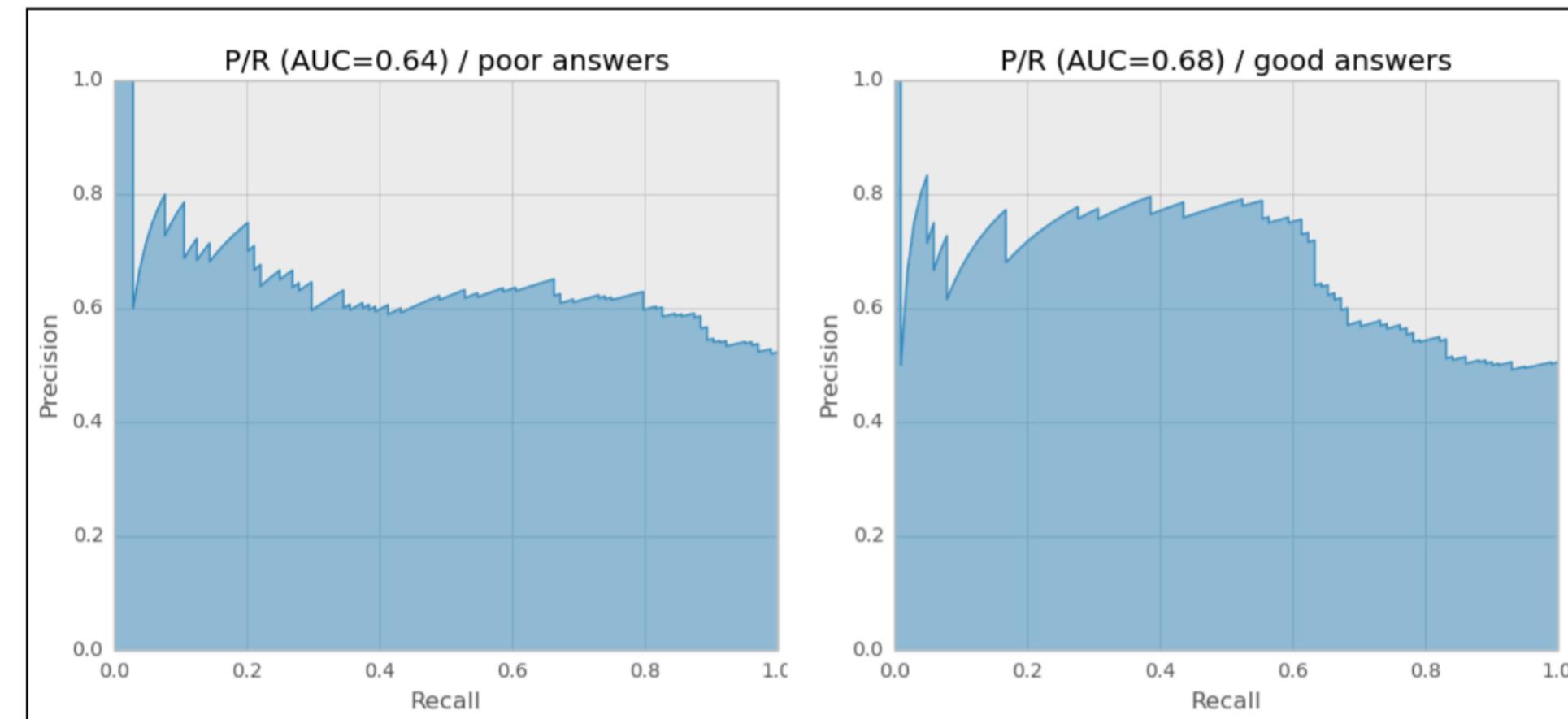
- 정확성에 대한 정의가 필요 
- Q: 유저 100명 중 2명 있는 어뷰저를 검출하려 한다. 실수로 모두 정상 유저로 판단했을 때 정확도는?
- A: 100명 중 2명이 틀렸으니... 98% !?#@

측정 단위

- 정밀도(Precision) 재현율(Recall)과 등 다양한 단위
 - 정밀도: 찾은 것 중 얼마나 진짜 어뷰저인가?
 - 재현율: 전체 어뷰저 중 얼마나 찾았는가?
- 데이터가 불균형(Imbalance)일때는 특히 정밀도와 재현율을 함께 고려해야
 - 앞의 경우는 재현율이 0

P/R Curve 와 AUC

좋은 분류기는?



사례2

기계학습으로 파밍 검출

상황

- 라이브 게임에서 각종 해킹 툴을 사용한 파밍 플레이가 활개 😈
- 파밍: 게임 내 재화를 비 정상적인 방법으로 습득
- 봇의 특성을 하나 둘로 특정하기 어려움 → 기계학습이 필요

학습 방식 선택

- 굳이 뉴럴넷/딥러닝으로 할 필요는 없는 듯...
 - 과거 로그가 저장되고 있었고,
 - 운영측에서 기존 어뷰저 캐릭터 리스트를 가지고 있었음 😊
- 기계학습, 특히 지도 학습이 가능!
- **Decision Tree** 방식의 지도 학습으로 결정

준비 과정

1. 로그 수집 상태 확인
2. 로그의 구조/의미 파악
3. 학습을 위한 피쳐(Feature) 추출

기계학습도 로그 수집부터

- 로그를 체계적으로 모으는 것도 쉽지 않음
- 분석/학습에 걸리는 시간은 10~20% 정도
 - 데이터를 모으고 가공하는데 대부분의 시간이 걸린다.
- 로그 형식은 가급적 그대로 사용 (스튜디오를 위해... 😅)
- 로그를 적절히 분류해 저장 (서버/로그 종류, 시점 별로)
 - 클라우드 스토리지(S3) 추천 ☁

윈도우 서버에서 로그 수집하기

- 게임 서버는 대부분 윈도우 기반
- 오픈 소스의 좋은 툴들(*fluentd*, *logstash* 등)을 쓰고 싶었으나
 - 윈도우 서버에 설치가 쉽지 않고, 일부 기능이 부족
- 자체 개발 
 - <https://github.com/haje01/wdfwd>
 - 서버에 남은 로그 파일을 **RSync**로 동기하거나
 - 게임 **DB**에 접속하여 **Dump** 후 전송

로그가 수집 되었으면 피쳐를 만들자

- 피쳐(Feature, 특성): 학습 대상의 특징을 설명해주는 값
- 예) 집 값을 예측하는 경우 
- 집의 크기, 방향, 환경, 교통, 편의시설 등이 피쳐

피쳐 개발(Feature Engineering)

- (비)정형 데이터에서 피쳐를 찾고 생성하는 작업
- 다른 피쳐들에 내재된 피쳐를 찾아내기도 함
 - 때로는 복잡한 코드가 필요(SQL로는 힘듦)
- 3개월 분량의 로그에서 하둡을 통해 피쳐 생성

하둡을 써야만 하나?

- 데이터가 **Big**하지 않으면 필요 없음
- 대신...
 - 배치 **Job**을 오랫동안 돌리거나
 - 주기적으로 **ETL**을 통해 **DB**에 넣어두는 과정이 필요할 수 있음
- 비정형/대용량 데이터에서 빈번한 피쳐 개발을 한다면 좋음

어떻게 써야하나?

- 직접 하둡 클러스터를 구축하여 사용할 수도 있으나,
셋팅과 운용의 어려움
- 클라우드 서비스에서 제공하는 하둡 서비스를 이용 😊
 - AWS의 EMR(Elastic Map Reduce)

AWS는 비싸지 않나?

- 최적화 하면 비싸지 않음 💰
- 필요할 때만 쓰는 단속적 클러스터(*Transient Cluster*)로 이용
 - Task 노드는 경매 방식의 *Spot Instance*로
 - m4.xlarge(4 vCPU, 16 GiB RAM): 시간 당 0.036\$
(서울 리전, 2016-08-09 기준)

AWS EMR 클러스터 시작 화면

The screenshot shows the AWS EMR Cluster Overview page. At the top, there is a filter bar with 'All clusters' selected and a dropdown for 'Filter clusters ...'. It displays '36 clusters (all loaded)'.

	Name	ID	Status	Creation time (UTC+9)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	<input checked="" type="checkbox"/> no_script.haje01.20151023.0 30648.230863	j-34AWLIR5JUH80	Bootstrapping	2015-10-23 12:06 (UTC+9)	4 minutes	0

Summary

Master ec2-52-68-15-190.ap-northeast-public DNS: 1.compute.amazonaws.com

Termination protection: Off [Change](#)

Tags: -- [View All / Edit](#)

Hardware

Master: Running 1 m3.xlarge

Core: Bootstrapping 1 m3.xlarge

Task: Provisioning 7 m3.xlarge (Spot: 0.042)

[View cluster details](#) [View monitoring details](#)

Steps

Name	Status	Start time (UTC+9)	Elapsed time
make_features.haje01.20151023.0.30648.230214: Step 1 of 1	Running	2015-10-23 12:11 (UTC+9)	1 second

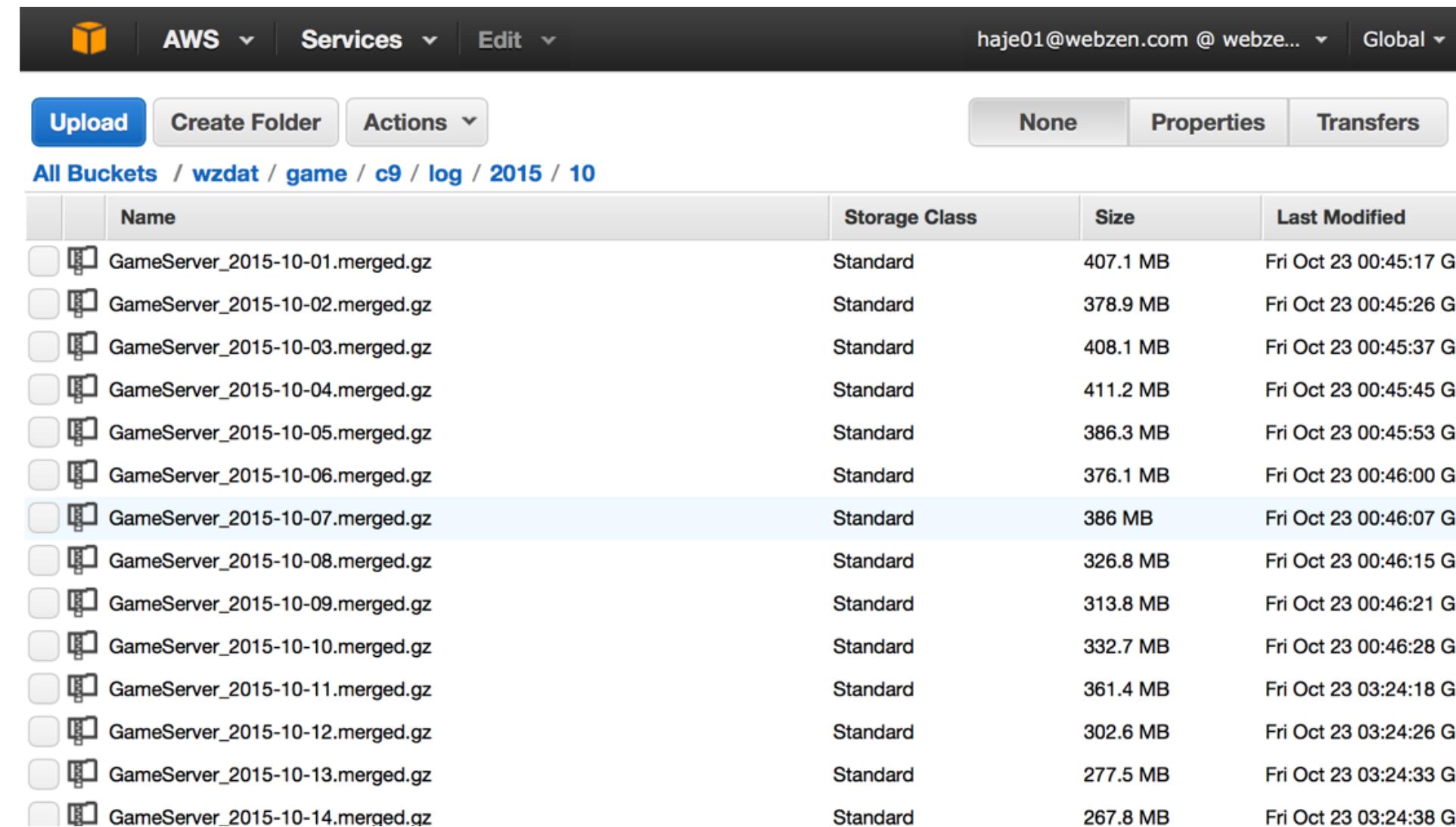
Bootstrap Actions

Name
master

하둡을 위한 로그 가공

- 하둡은 작은 파일(< 100MB)들이 많은 것에 취약
 - 작은 파일들은 병합, 소팅, 압축할 필요
- 마땅한 툴을 찾지 못해 개발 
- <https://github.com/haje01/mersoz>
- 바뀐 파일만 작업, 의존 관계를 고려한 병렬 처리

마지, 소팅 & 압축 후 S3에 저장된 로그



The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, 'AWS Services Edit', and user information 'haje01@webzen.com @ webze... Global'. Below the navigation bar, there are tabs for 'Upload', 'Create Folder', and 'Actions'. Under 'Actions', there are buttons for 'None', 'Properties', and 'Transfers'. The main area displays a table of objects in a bucket. The table has columns for 'Name', 'Storage Class', 'Size', and 'Last Modified'. The 'Name' column lists 14 files named 'GameServer_2015-10-[day].merged.gz' from 01 to 14. All files are in 'Standard' storage class, with sizes ranging from 267.8 MB to 411.2 MB, and were last modified on Friday, October 23, 2015.

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	GameServer_2015-10-01.merged.gz	Standard	407.1 MB	Fri Oct 23 00:45:17 GM
<input type="checkbox"/>	GameServer_2015-10-02.merged.gz	Standard	378.9 MB	Fri Oct 23 00:45:26 GM
<input type="checkbox"/>	GameServer_2015-10-03.merged.gz	Standard	408.1 MB	Fri Oct 23 00:45:37 GM
<input type="checkbox"/>	GameServer_2015-10-04.merged.gz	Standard	411.2 MB	Fri Oct 23 00:45:45 GM
<input type="checkbox"/>	GameServer_2015-10-05.merged.gz	Standard	386.3 MB	Fri Oct 23 00:45:53 GM
<input type="checkbox"/>	GameServer_2015-10-06.merged.gz	Standard	376.1 MB	Fri Oct 23 00:46:00 GM
<input type="checkbox"/>	GameServer_2015-10-07.merged.gz	Standard	386 MB	Fri Oct 23 00:46:07 GM
<input type="checkbox"/>	GameServer_2015-10-08.merged.gz	Standard	326.8 MB	Fri Oct 23 00:46:15 GM
<input type="checkbox"/>	GameServer_2015-10-09.merged.gz	Standard	313.8 MB	Fri Oct 23 00:46:21 GM
<input type="checkbox"/>	GameServer_2015-10-10.merged.gz	Standard	332.7 MB	Fri Oct 23 00:46:28 GM
<input type="checkbox"/>	GameServer_2015-10-11.merged.gz	Standard	361.4 MB	Fri Oct 23 03:24:18 GM
<input type="checkbox"/>	GameServer_2015-10-12.merged.gz	Standard	302.6 MB	Fri Oct 23 03:24:26 GM
<input type="checkbox"/>	GameServer_2015-10-13.merged.gz	Standard	277.5 MB	Fri Oct 23 03:24:33 GM
<input type="checkbox"/>	GameServer_2015-10-14.merged.gz	Standard	267.8 MB	Fri Oct 23 03:24:38 GM

하둡 MapReduce 코딩 - mrjob

- Yelp에서 만든 Python 패키지
- 하둡 스트림을 이용해 파이썬으로 MR 코딩
- 로컬에서 샘플 데이터로 개발한 후, EMR에 올림 
- 실행 속도는 Java 버전 보다 좀 느리지만 개발 속도가 빠름

```
from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"\w+")

class MRWordFreqCount(MRJob):

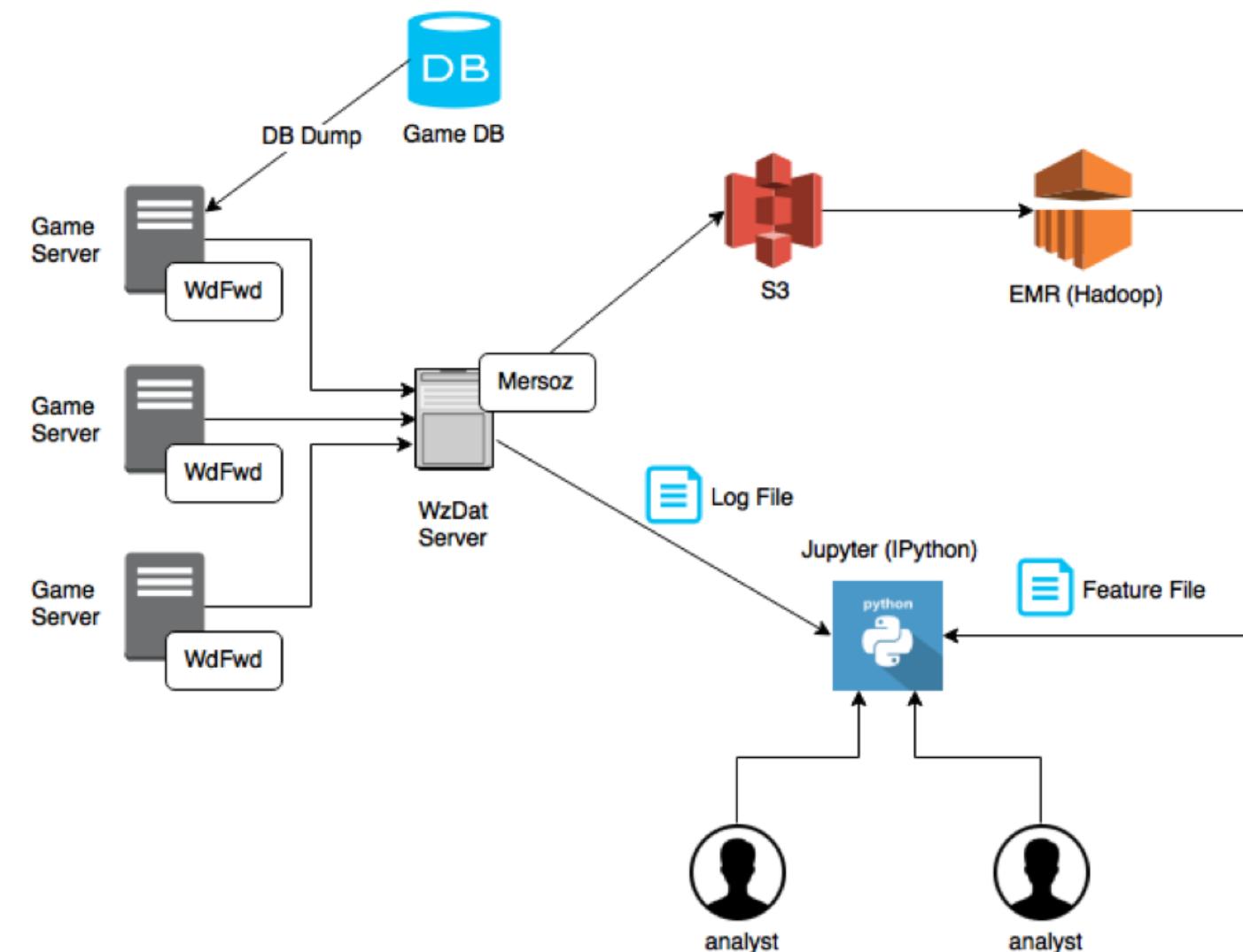
    def mapper(self, _, line):
        for word in WORD_RE.findall(line):
            yield word.lower(), 1
                # 로그 파일의 각 라인의
                # 모든 단어에 대해
                # '단어', 1 반환

    def combiner(self, word, counts):
        yield word, sum(counts)
                # 노드의 결과를 취합

    def reducer(self, word, counts):
        yield word, sum(counts)
                # 클러스터의 결과를 취합

if __name__ == '__main__':
    MRWordFreqCount.run()
```

시스템 구성도



현황 파악

- 기계학습을 위해
 - GM이 제작하는 근거(=피쳐)와
 - 제작된 캐릭터 리스트를 요청

피쳐 생성 팁

- 로그에서 캐릭터 기준으로 구함
- 정교한 피쳐보다는 다양한 피쳐를
 - 어차피 복합적으로 판단
- 초기에는 짧은 시간에 대해, 안정화되면 길게

초기에 뽑아본 피쳐들

- 로그인 수
- 플레이 시간
 - 로그 아웃이 불분명한 경우가 많음
 - 세션 아웃 도입: 5분 
- 아이템/머니 습득 수
- 퀘스트 종료 수
- NPC/PC 간 전투 수

피쳐의 타입은?

- 크게 실수 형, 카테고리 형, 불린(Boolean) 형으로 나눠짐
 - 가급적 실수 형으로 통일하는 것이 바람직
- Bool은 0, 1로
- 카테고리 타입은 OneHotEncoder를 사용해 실수형으로

만들어진 피쳐의 예

- 단순 텍스트 (.txt) 파일
 - 캐릭터명 + 피쳐 배열 형식

기계학습 진행

정작 기계학습은 가벼움

- 최종 피쳐 파일 크기가 작고, 기계학습 수행도 가벼운 편
 - 로컬 PC에서 수행
 - 추천 시스템처럼 모든 데이터를 봐야하는 학습은 무거울 것
- 모델을 선택하고 최적의 하이퍼 패러미터를 결정하는 것이 과제
 - 다양한 셋팅으로 여러번 실험해봐야
 - 분산 시스템을 활용하는 경우도...

어떤 알고리즘 모델을 선택할 것인가?

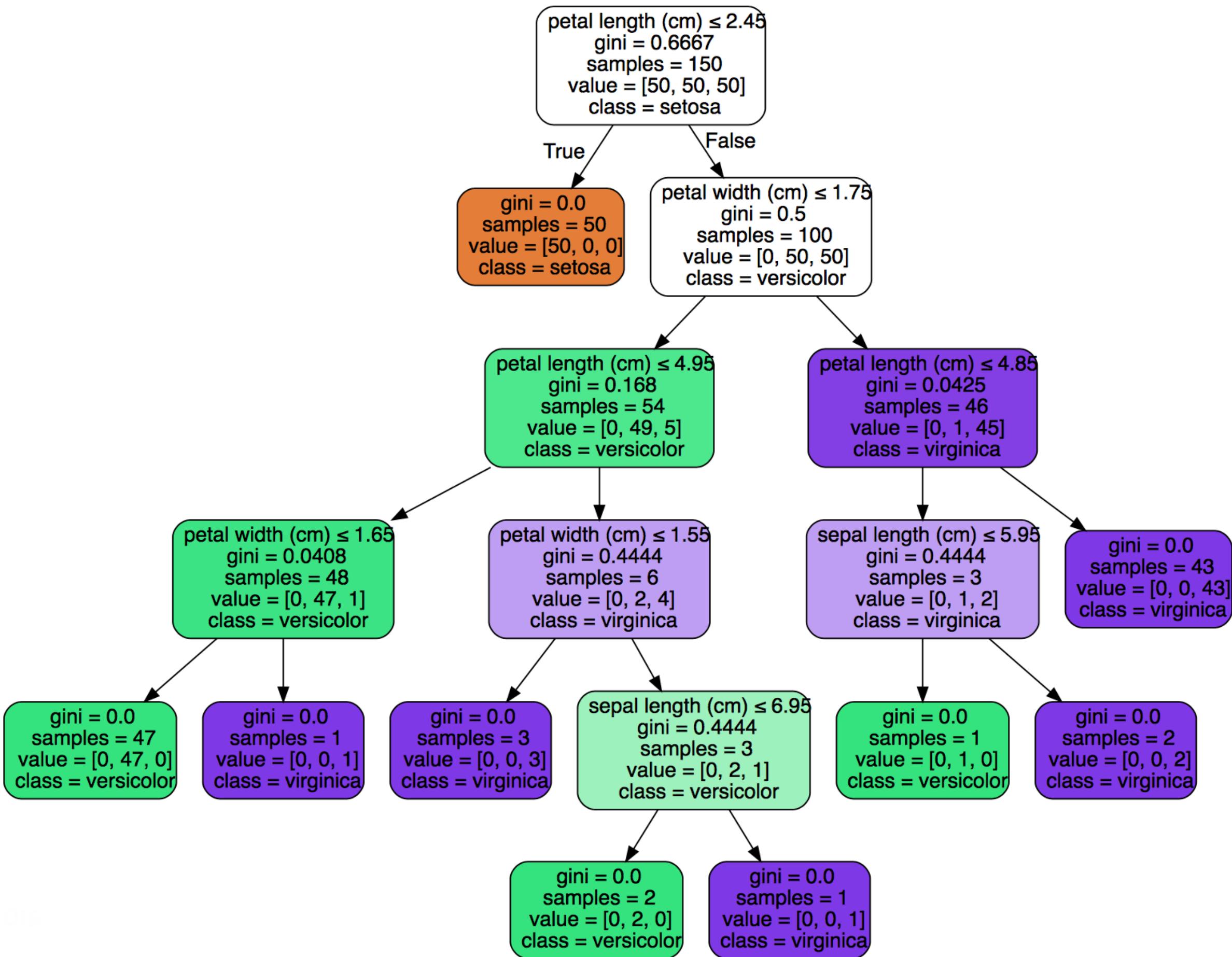
- 시작은 간단한 것으로
- 비슷한 사례의 선행 연구가 있으면 참고하자
- AUC나 ROC를 통한 모델 평가 및 선택

Decision Tree로 시작

- 복잡하지 않고 판단 과정의 이해가 용이
- 파이썬 **Scikit-Learn** 패키지의 것을 사용
 - 다양한 기계학습 알고리즘을 충실히 제공
 - 인터페이스가 통일되어 있어 모델 교체가 용이
- 피쳐(X)와 어뷰저 여부(y)를 넣고 학습
 - DT는 피쳐 정규화 필요 없어 편리

DT 사용 예 (붓꽃 분류)

```
from sklearn.datasets import load_iris  
from sklearn import tree  
  
iris = load_iris()  
clf = tree.DecisionTreeClassifier()  
clf = clf.fit(iris.data, iris.target)  
  
>>> clf.predict(iris.data[:1, :])  
array([0])
```



Decision Tree 학습 과정

1. 피쳐 파일에서 기존 어뷰저의 피쳐를 찾고
2. 동수의 정상 유저 피쳐 구함
→ Under Sampling
3. 데이터를 Train/Test 셋으로 나누고
4. 기본 패러미터로 학습 시작

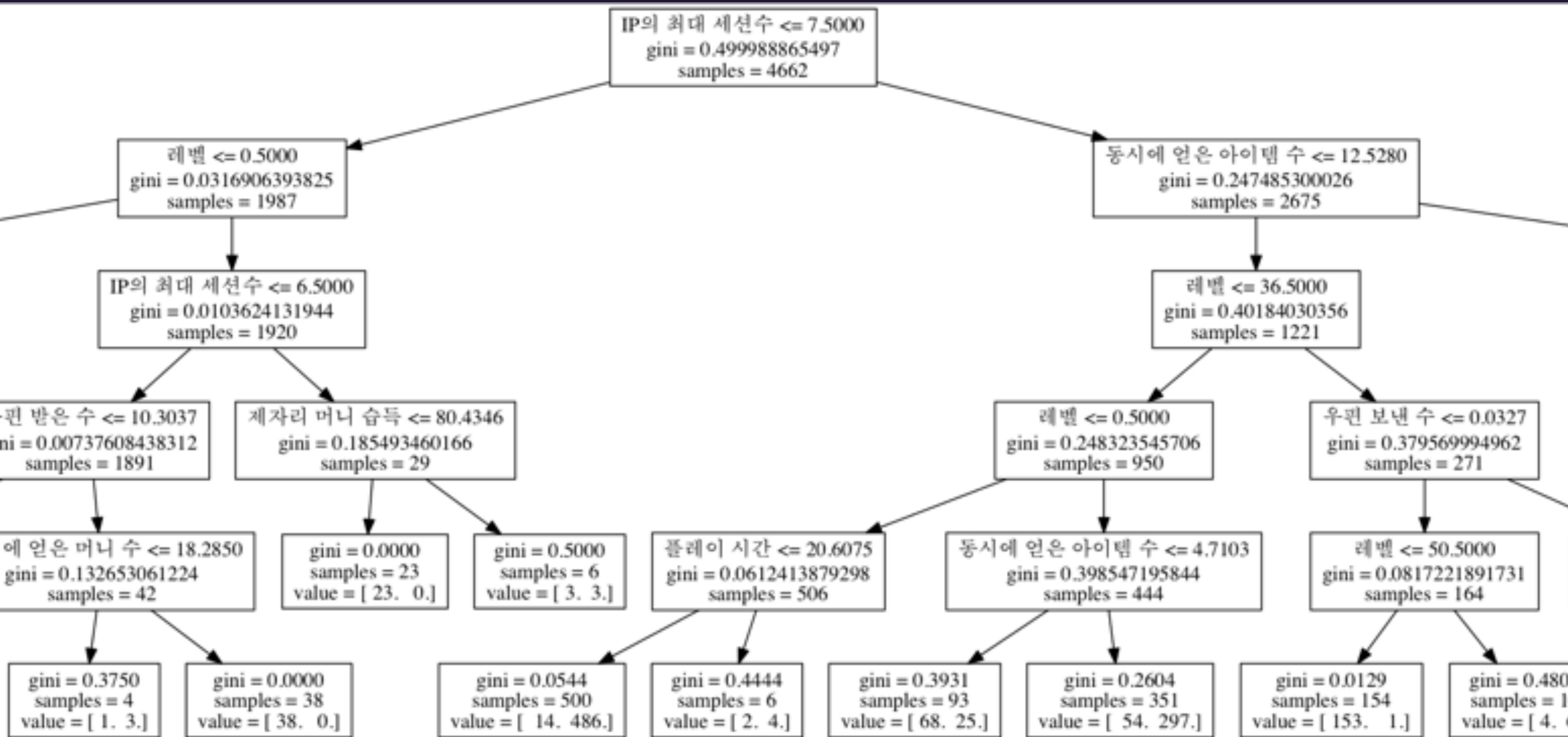
초기 결과

- 평균 정확도 80% 정도
- **Binary Class** 분류의 경우 점수가 잘 나오는 편
- 나쁘지 않은것 같지만,
 - 예측의 결과가 제재의 근거로 쓰인다는 점에서 많이 부족

정확도를 올리자

- 교차 검증(Cross Validation)을 위해 데이터 셋을 분리하고
- GridSearchCV를 통해 최적의 하이퍼 패러미터를 찾음
 - 평균 정확도 91%로 향상
- 어떤 기준으로 판단하는지 한 번 보고 싶다

tree.export_graphviz로 그려봄



결정 트리를 보니...

- 학습된 모델이 어떤 기준으로 판단하는지 알 수 있음
 - 다양한 직군의 사람들에 공유 가능 😊
- 하부로 내려갈 수록 복잡해지는 문제
- DT는 과적합(Overfitting)되기 쉽기에, Depth가 너무 깊지 않게 주의

여기서 더 이상 점수가 올라가지 않음

- GM님과 상의 후 새로운 피쳐들 추가
 - 동시에 얻은 아이템/머니 수
 - 맵 반복 횟수
 - 특정 클래스만 선택
 - 움직이지 않고 아이템을 얻은 수
- 난해해 보이는 것들도 피쳐로 만들 수 있는 것이 노하우
 - 예) '봇은 랜덤하게 생성된 이름을 가지고 있어요'

예) 캐릭터 이름의 랜덤성 판단 (자/모의 출현 패턴)

```
## 캐릭터 이름이 발음 가능한지 판단하는 슈도 코드
# 이름을 자모 심볼로 바꿈(1가 자음, 2가 모음)
# 예) anything -> '21211211'

symbols = get_cv_symbols(char_name)

# 다음과 같은 패턴이 있으면 발음 가능 (실제로는 더 다양)
if '2121' or '2112' or '1121' or '22122', ... in symbols:
    can_pron = False
else:
    can_pron = True
```

정확한 방법은 아니지만...

복합적으로 판단하기에 도움이 됨

추가 피쳐로 스코어가 향상, 그러나...

- 평균 정확도 96%로 향상. 점수는 높은 편이지만,
- 실제 적용해본 결과
 - GM님의 확인 과정에서 오탐이 꽤 나옴 😞
- DecisionTree의 고질적인 과적합 문제로 판단

Random Forest로 교체

- 많은 Decision Tree 를 조합한 앙상블 테크닉
- 다수의 DT를 분산 학습(=정규화 효과) 시키고 투표하는 방식
 - 점수가 낮아도 안정적인 결과
 - DecisionTree - 불안한 96%
RandomForest - 안정적인 95%

Random Forest 학습

- 기본적으로 Decision Tree와 비슷
- `max_depth`, `min_samples_leaf`
모델의 복잡도를 조절. 작게 시작해서 조금씩 키워본다
- `n_estimators`
 - 나무(DT)를 몇 그루 심을 것인지 결정 🌲
 - 너무 크면 학습시간이 길고, 너무 적으면 그냥 DT가 되어버림

RF 적용 후 결과

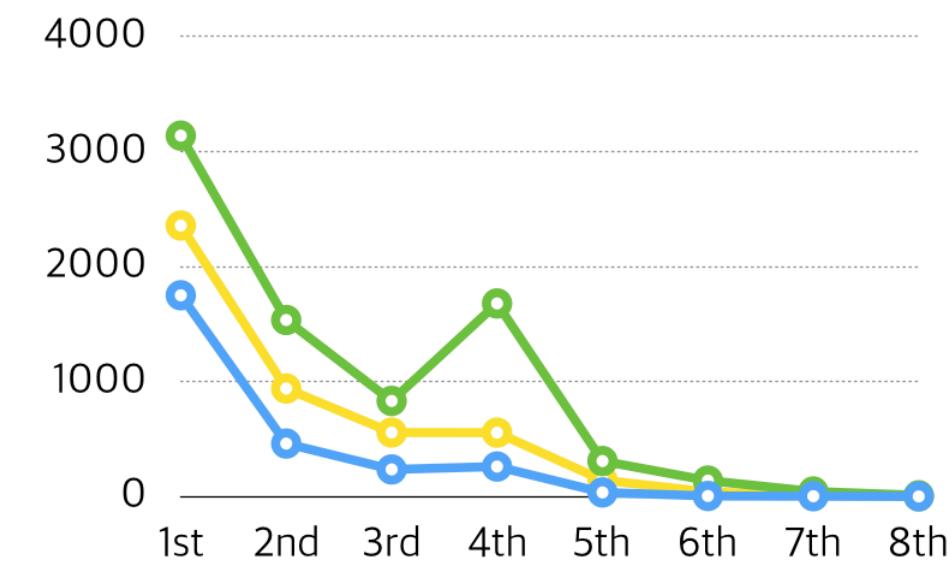
- 정확도는 95%
- 부당하게 징계 받는 사례가 없도록
 - `predict_proba`를 사용해 예측의 확률도 얻고
 - 확률이 높은(>70%) 예측 결과만 포함
 - 여기서 10~20%정도 재현율(Recall) 하락 추정
- 그러나, 정밀도(Precision)는...

100% 달성

GM님이 수작업으로 검토해 주신 결과... 😂

찾았으니 제재를...

	1st	2nd	3rd	4th	5th	6th	7th	8th
ASIA	1750	460	235	260	35	4	2	0
EU	3139	1535	831	1679	308	138	43	10
US	2357	939	556	555	142	39	10	0



- 2개월여에 걸쳐 제재
- 툴을 사용한 파밍이 대부분 사라짐! 😎
- 주기적/지속적으로 제재를 해야 효과가 있음

참고: 최종 피쳐의 중요도

플레이 시간: 0.03
죽은 수: 0.00
레벨업 수: 0.06
NPC킬 수: 0.01
PC킬 수: 0.00
동일 맵 비율: 0.01
복잡한 이름: 0.13
아이템 획득 수: 0.03
동시에 얻은 아이템 수: 0.04
유료 아이템 구매 수: 0.00
아이템 버린 수: 0.02
아이템 강화 수: 0.01
머니 변동 수: 0.03

아이템 거래 수: 0.00
머니 거래 수: 0.00
머니 거래 총액: 0.00
퀘스트 종료 수: 0.12
우편 보낸 수: 0.01
우편 받은 수: 0.01
제자리 아이템 습득: 0.03
제자리 머니 습득: 0.08
PvP 횟수: 0.00
슬레이어 계열: 0.00
레벨: 0.11
IP의 최대 세션수: 0.23
동시에 얻은 머니 수: 0.04

개선 방향

- 검출된 결과를 이용해 학습 모델 개선
- 봇 계정에 대한 PII를 수집해두면 신규 봇 학습에 용이할 것
- 제재 후 변종 봇 모니터링 필요

후기

느낀 점

- 데이터 수집부터 가공, 분석까지의 모든 과정을 파이썬으로 
- Jupyter 노트북을 통한 탐색적 데이터 분석 
- 더 다양한 분야에 기계 학습을 활용할 듯

기계학습 심화

- 깊이 있는 활용을 위해 기본 이론을 더 공부하자 
- 좋은 **Hypothesis**를 만들 수 있게 된다
- 최적화를 할 수 있게 된다
- 하나 이상의 알고리즘을 사용해 보자
- **SVM, Neural Net** 등 다양한 분류기
- **Super Learner** 방식으로 양상을

세월은 흘러... 새로운 로그 수집/분석 환경

- **RSync** 방식 -> **Fluentd/Kinesis** 실시간 로그 수집
- **gzip**된 **CSV** -> **Parquet** 포맷으로 **S3** 저장
 - **Columnar** 바이너리 포맷, 30x 속도 향상
- **MRJob** -> **PySpark**
 - 강력한 분산 처리 / **Cache** 기능(반복 학습에 강점)
 - 단속적 **Spark** 클러스터(20 VMs = 80코어, 320GB 램)로 이용 중
(시간 당 3000원 정도)

조언

- 기계학습이 내가 하려는 일에 적합한지 판단 
- 어뷰징의 특성이 단순하면 전통적인 방법으로 가능
- 탐색적 데이터 분석을 통해 특성을 먼저 파악하자
- 다양한 모델/피쳐를 테스트해보자
- 학습 모델에 따라 피쳐 정규화/직교화가 필요할 수 있으니 체크
- 클래스간 **Imbalance** 문제에 주의

딥러닝? 기계학습?

- 딥러닝
 - 정교한 피쳐 엔지니어링이 필요 없음
 - 많은 패러미터 = 많은 데이터가 필요
- 기계학습
 - 피쳐 작업이 중요하지만
 - 적은 패러미터 = 적은 데이터로도 효과

Pixels vs. Vector



잡상

- 데이터 엔지니어링의 어려움
 - 데이터의 확보가 가장 중요
- 스포트라이트를 받는 분야는 오히려 전망이 어두움
 - 탑 연구자가 아니라면 굴뚝산업/틈새 데이터야말로 블루오션
 - 모든 회사에 데이터 분석가가 필요한 시대
- 컴퓨터가 모든 모델/변수 조합을 테스트 할 수 있다면? 😊

끝으로... 의사 연관(Spurious Correlations)



- 실제로는 연관이 없지만, 있는 것처럼 보이는 경우
- 데이터에만 집착하지 말고, 도메인을 이해하자!

감사합니다.

참고 링크

- <http://www.aladin.co.kr/shop/wproduct.aspx?ItemId=28946323>
- <http://www.tylervigen.com/spurious-correlations>
- <http://scikit-learn.org/stable/modules/tree.html>
- <http://www.cimerr.net/conference/board/data/conference/I33I626266/PI5.pdf>
- <http://stackoverflow.com/questions/20463281/- how-do-i-solve-overfitting-in-random-forest--of-python-sklearn>
- <http://stats.stackexchange.com/questions/131255/class-imbalance-in-supervised-machine-learning>
- <https://www.quora.com/Is-Scala-a-better-choice-than-Python-for-Apache-Spark>
- <http://statklee.github.io/data-science/data-handling-pipeline.html>
- <https://databricks.com/blog/2016/01/25/deep-- learning-with-spark-and-tensorflow.html->