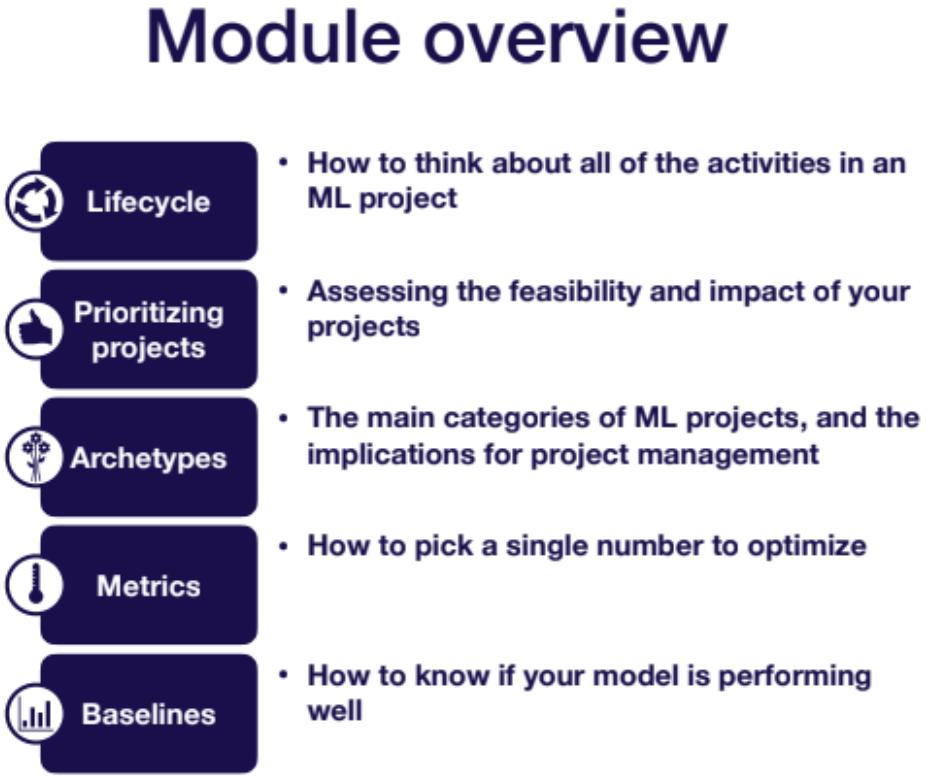
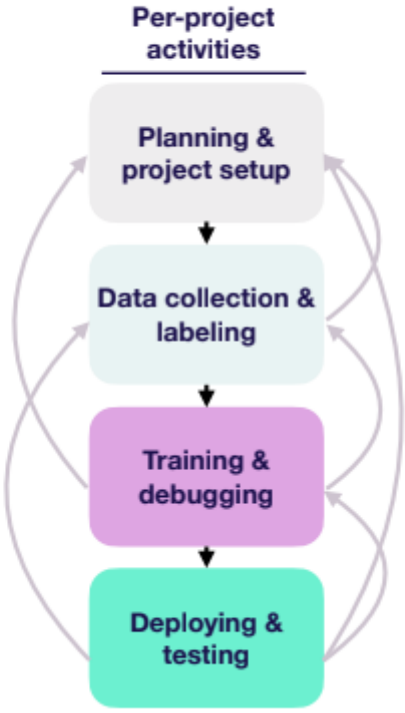
ML Projects

- 85% 의 AI 프로젝트는 실패
  - 왜 이렇게 많은 프로젝트가 실패하는가?
    - 여전히 ML은 연구중인 영역
    - 실패할 운명이다.
      - 기술적으로 불가능, 범위가 노답
      - 제품으로 도약된적없음
      - 깔끔한 성공 기준이 없음
      - 절망적인 팀 관리
- Module Overview
  - 

  - 생태싸이클
  - 프로젝트 우선순위정하기
  - 아키타입
  - 메트릭
  - 기준선
- Lifecycle

Lifecycle of a ML project



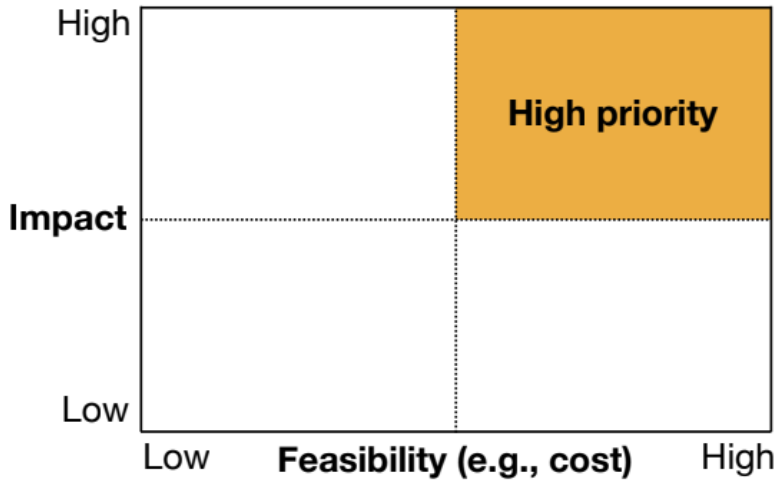
1. 계획, 프로젝트 Setup
2. 데이터 수집, 라벨링
3. 학습, 디버깅
4. 운영, 테스트

- 2 to 1
  - 데이터 얻기가 어렵거나, 다른 task의 데이터 라벨링이 더 낫거나
- 3 to 2
  - 데이터가 아직 부족하거나, 라벨링을 믿을 수 없거나
- 3 to 1
  - Task가 너무 힘들어요.. 요구사항이 중요한 것이 아니라서 다른 것과 trade off되었어요..
- 4 to 3
  - Lab에서 작동을 안해요... 정확도를 향상시켜야 겠어
- 4 to 2
  - 훈련데이터와 배치된 후 데이터가 달라 고쳐야할 때, 데이터가 더필요할 때
- 4 to 1
  - 선택한 메트릭이 맞지않아 다시 선택. 혹인 실제 세계에서 퍼포먼스가 구림

- 프로젝트 단위의 Activities는 위와 같으나 Cross-project Infrastructure는 다름
  - 팀단위, 고용
  - 인프라, tooling
- 무얼알아야 하는가?
  - 당신의 도메인
    - 무엇이 가능한지 알기
    - 다음에 시도해야 할 게 무엇인지 알기

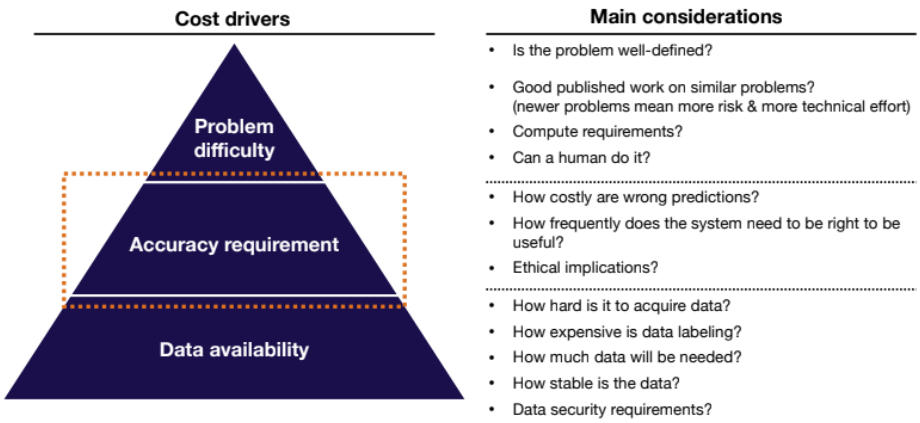
- Priortizing Projects

- A (general) framework for prioritizing projects



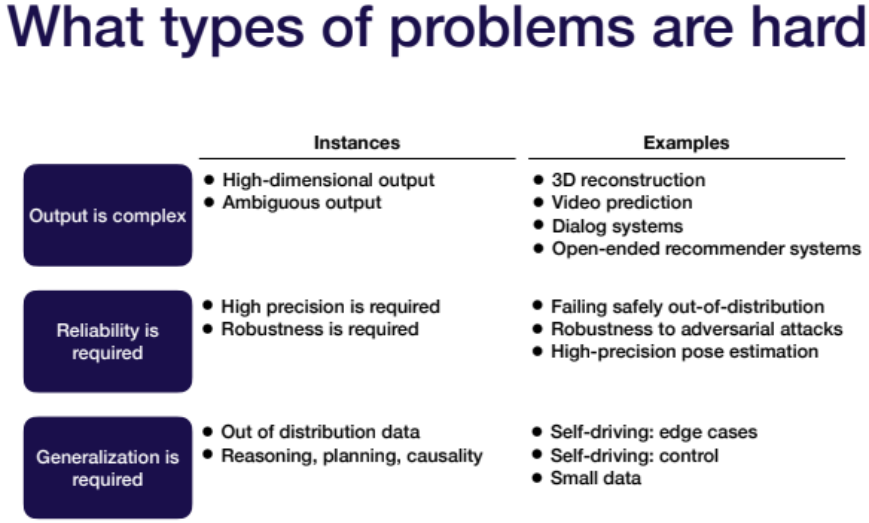
    - Impact가 높고 Feasibility도 높은 것이 우선순위가 높다.
- 높은 impact의 ML프로젝트를 위한 멘탈 모델
    - 어디서 cheap 예측의 이점을 가져올 수 있는가?
    - 프로젝트에서 마찰은 어디있는가?
    - 복잡한 process를 어디서 자동화할 수 있는가?
    - 다른 사람들은 무얼하는가?
- ML 프로젝트를 경제적으로 실현가능하게 만드는것?
    - AI는 예측 cost를 줄인다
    - 예측은 의사결정의 중심이다.
    - Cheap한 예측이란
        - 어디서든 예측이 가능
        - 이전엔 너무 비쌌던 문제
- 프로젝트에게 필요한 것? 40p
    - Software 2.0
        - 1.0은 전통적인 프로그래밍
        - 2.0은 목표를 지정해주면 알고리즘이 찾아나가는 방식
        - 2.0프로그래머는 데이터셋으로 일한다.
    - 다른사람들은 무얼하는가?
        - 구글, 페이스북, 넷플릭스에서 나오는 논문들
        - 다양한 회사들의 블로그 포스트들
    - ML 프로젝트의 타당성 평가
        -

Assessing feasibility of ML projects



- 문제의 어려움
  - 문제가 잘 정의되었는가?
  - 비슷한 문제에 잘 퍼블리싱된 일이 있는가?
  - 요구사항 계산
  - 인간이 할 수 있는가?
- **정확도 요구**
  - 잘못된 예측에 드는 비용?
  - 시스템에 유용하기 위해 얼마나 자주 옳아야 하는가?
  - 도덕적인 시사점
- 데이터 가능성
  - 데이터 얻기 어려운가
  - 라벨링에 소요되는 cost
  - 얼마나 많이?
  - 얼마나 stable한가?

- 정확도가 높아질수록, 높을수록 cost가 기하급수적으로 엄청나게 발생함
- ML에서 여전히 hard한것은?
  - 유머, 빈정거림 이해하기
  - In-hand Robotic 조작
  - 새로운 시나리오 생성하기
  - 비지도학습
  - 강화학습
    - 위의 둘은 제한된 도메인에서, 거업나 많은 데이터와 계산을 통해 가능함
- 지도학습에서 어려운 것?
  - 문제에 대답하기
  - 텍스트 요약
  - 비디오 예측
  - 3D 모델 구축
  - 리얼월드 음성 인식
- 다양한 어려운 문제점들
  -



        - 출력물이 복잡함
            - 고차원
        - 신뢰성이 요구됨
            - Robust하든가
            - 정확도가 높든가
        - 일반화가 요구됨
            - Distribution 데이터 부족
    - 왜 Full stack 로봇공학은 Pose estimation에 집중하는가?
        - 전통적 방식은
            - 느리고, 불안정함
        - Feasibility
            - 데이터가능성 : 데이터를 얻기 쉽고, 라벨링 데이터는 센서를 통해 생성가능
            - 정확도 요구수준 : 0.5cm미만의 높은 정확도가 요구됨
            - 정확도 요구수준 : 실패했을 때 cost는 작음.
                - % 성공률이 아니고 시간당 몇번 집는지가 중요
            - 문제 난이도 : 비슷하게 Published된 결과가 있지만 우리의 물건과 로봇에 적용해야함.
    - 어떻게 ML 타당성평가를 하는가?
        - ML이 정말 필요한가?
        - 성공기준을 정의하기 위해 모든 이해관계자와 사전작업진행
        - ML을 사용하는데에 있어 윤리적 부분 고려
        - Literature 리뷰를 해라
        - 라벨링된 벤치마크 데이터셋을 빠르게 만들기를 시도해라
        - 최소한의 싱행가능한 product를 만들어라(manual rule 같은 것.)
        - 이래도 ML이 정말 필요한가!?

- Archetypes 60p
    -

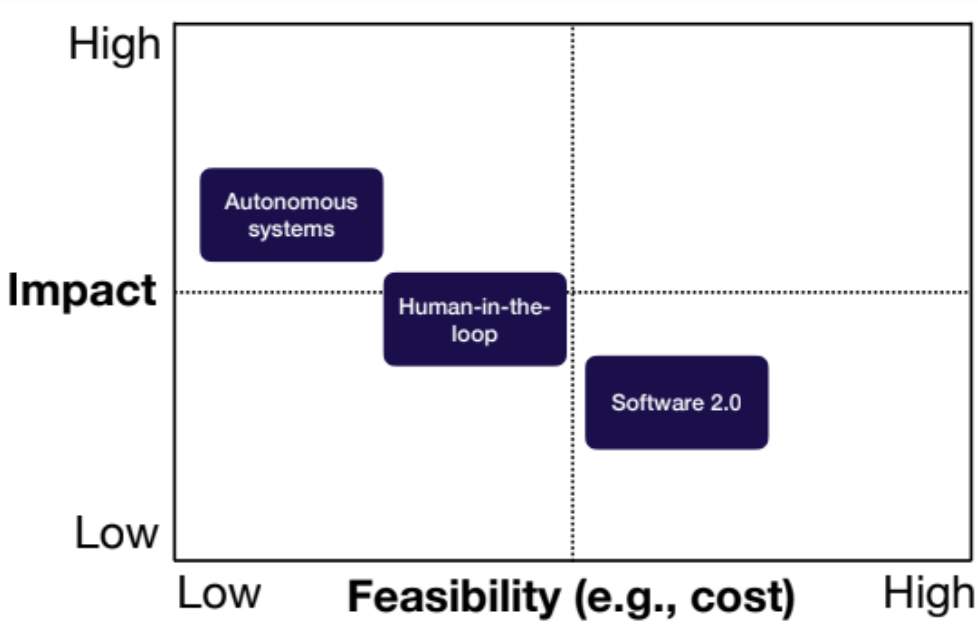
Machine learning product archetypes

	Examples
Software 2.0	• Improve code completion in an IDE • Build a customized recommendation system • Build a better video game AI
Human-in-the-loop	• Turn sketches into slides • Email auto-completion • Help a radiologist do their job faster
Autonomous systems	• Full self-driving • Automated customer support • Automated website design

- # Machine learning product archetypes

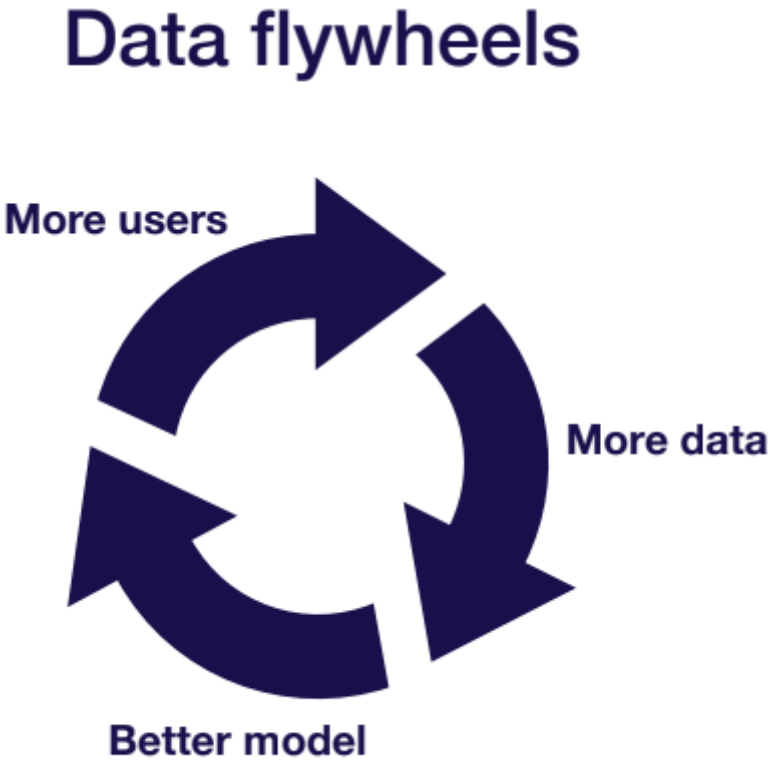
	Key questions
Software 2.0	• Do your models truly improve performance? • Does performance improvement generate business value? • Do performance improvements lead to a data flywheel?
Human-in-the-loop	• How good does the system need to be to be useful? • How can you collect enough data to make it that good?
Autonomous systems	• What is an acceptable failure rate for the system? • How can you guarantee that it won't exceed that failure rate? • How inexpensively can you label data from the system?

- Software 2.0 : high feasibility, low Impact
- Human-in-the-loop
- Autonomous systems : High Impact, low feasivility
-

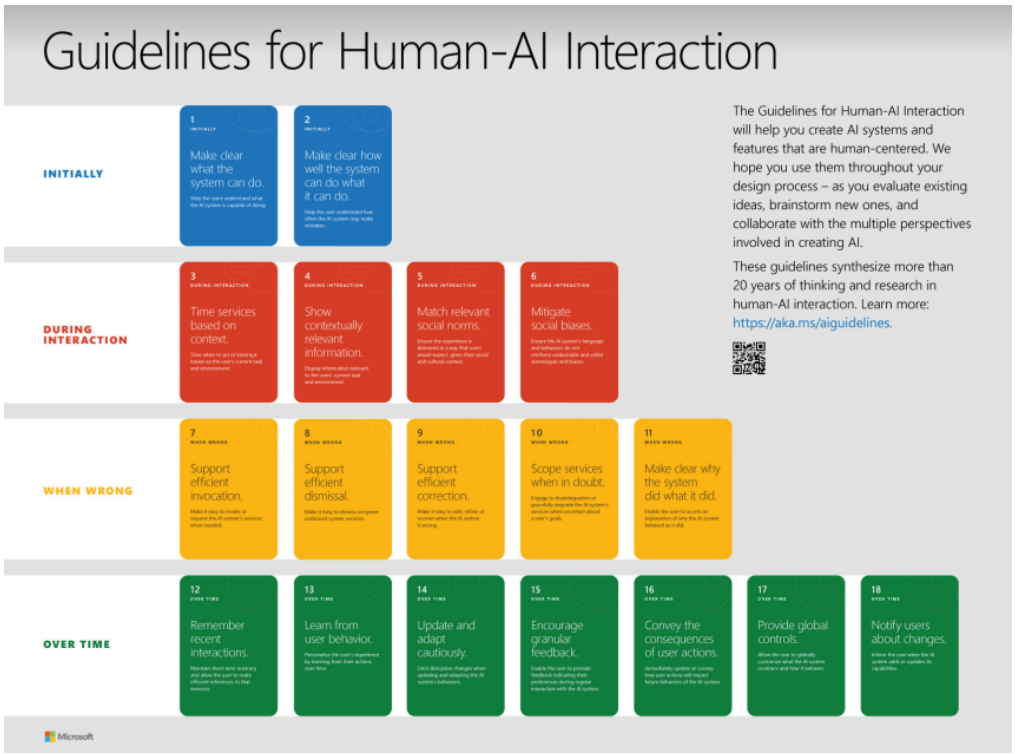


- 서로 Trade off 관계에 있다.

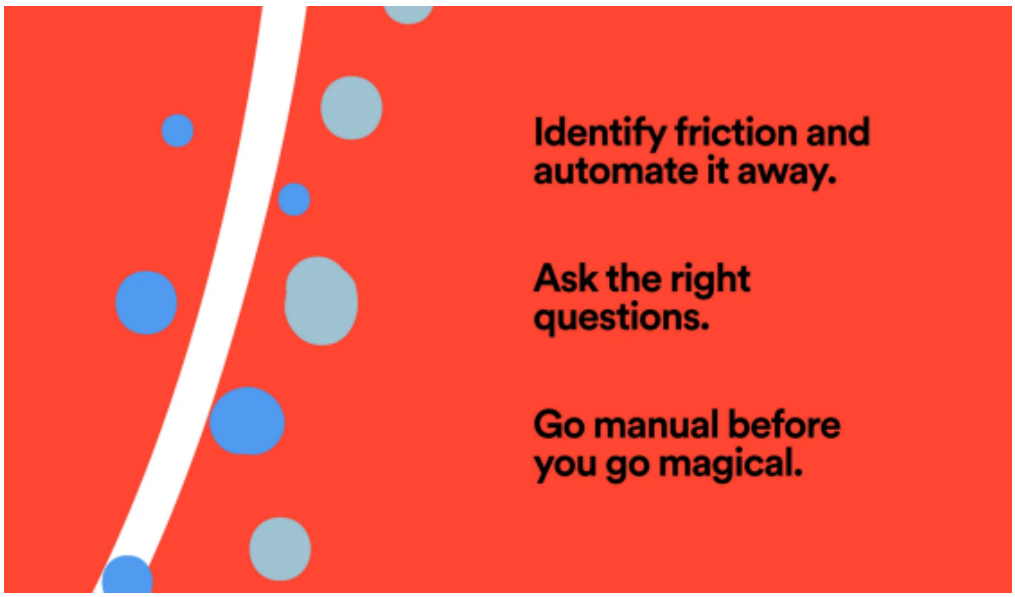
- Data flywheels



    - 더 많은 유저, 더 많은 데이터, 더 나은 모델
    - 반복!
- 제품 설계는 정확성에 대한 필요를 줄일 수 있다.
- Apple의 ML제품 디자인 가이드라인
    - 앱에서의 ML의 역할은?
    - 유저로 부터 무엇을 배울수있는가?
    - 어떻게 실수를 핸들링할 것인가?
- MS사의 가이드라인

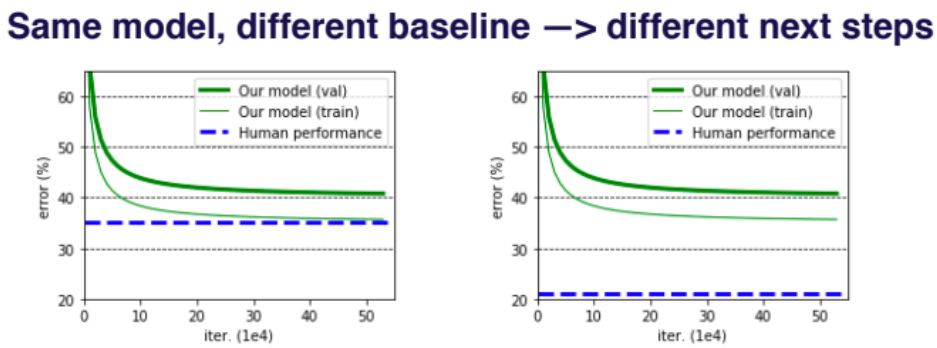


- Spotify의 가이드라인

- 

- Metrics
  - 메트릭을 고르는 Key
    - 현실세계는 복잡하므로 여러가지 메트릭을 고려해야함
    - 하지만 ML은 single number에 최적화했을때 베스트로 일함
    - 그러므로 공식을 골라 여러가지 메트릭을 combining한다.
    - 그 formula는 계속 바뀔것이다.
  - Accuracy, Precision, recall
    - Confusion Matrix
      - Precision과 Recall만 가지고 어느 모델이 좋은지 정할 수 있는가?
  - Combine하기
    - 산술평균? 가중평균?
    - n-1번째 까지 Threshold 하고 n번째 평가하기
    - 어떤 메트릭을 Thresholding 할까?
      - 도메인에서 결정하기
      - 어떤 메트릭이 모델선택에 가장 덜 민감한가?
      - 어떤 메트릭이 바람직한 값과 가장 가까운가?
    - Thresholing 값 고르기
      - 도메인에서 결정
      - 기준선 모델이 얼마나 잘 작동하는가?
      - 메트릭이 지금 얼마나 중요한가?
    - 더 복잡한, 도메인 최적화된 Formula
      - mean Average Precision = mAP
        - Recall에 따른 Precision값을 나타낸 곡선 PR곡선
        - 여기서의 Precision값의 평균이 AP(PR 곡선의 면적)
        - 단일 객체에 대한 성능이므로 이것들을 모아서 평균내면 mean AP
  - FSR로 돌아와서...
    - 우선 요구사항을 항목화하기
    - 현재 퍼포먼스를 평가하기
      - 몇개의 모델을 학습하기
    - 요구사항과 현재의 퍼포먼스 비교하기

        - (각도 에러 우선, 실행시간은 일단 나중에. 등등)
    - 숫자가 오르면 metric을 다시 확인하기

- Baselines
    - 기준선을 정하는 Key
        - 기준선은 기대되는 모델 퍼포먼스의 하한선을 제시한다.
        - 하한선에 타이트해질수록 더 useful한 기준선이다.
        - 

            - 휴먼에러에 가까워지니 좋은모델 VS 턱도없이 오류가 많은 모델
    - 외부적인 기준선
        - 비즈니스, 공학적 요구사항
        - Published된 결과물
    - 내부적인 기준선
        - Scripted된 기준선
        - 단순한 ML 기준선
        - 휴먼 퍼포먼스
    - 좋은 인간 기준선 만드는 방법
        - 

        - Amazon Turk : 데이터 라벨링 크라우드소싱 서비스
- 결론
    - Lifecycle : ML프로젝트는 반복적이다.
    - Prioritizing Project : Impact있게, 잘못된 예측에 대한 cost는 낮게
    - Archetypes : 자동화된 데이터 Flywheels를 만드는 것이 중요
    - Metrics : 실세계에선 많은 것을 신경써야 하지만, ML에 있어선 하나만 신경써라.
    - Baselines : 좋은 기준선은 당신의 노력을 올바른 방법으로 투자할 수 있도록 도와준다.