Valohai

실리콘밸리의 ML옵스

머신러닝 서비스 구축을 위한
실전 ML옵스 가이드

WITH CHAPTERS FROM

Superb AI SIGOPT tecton

Contents

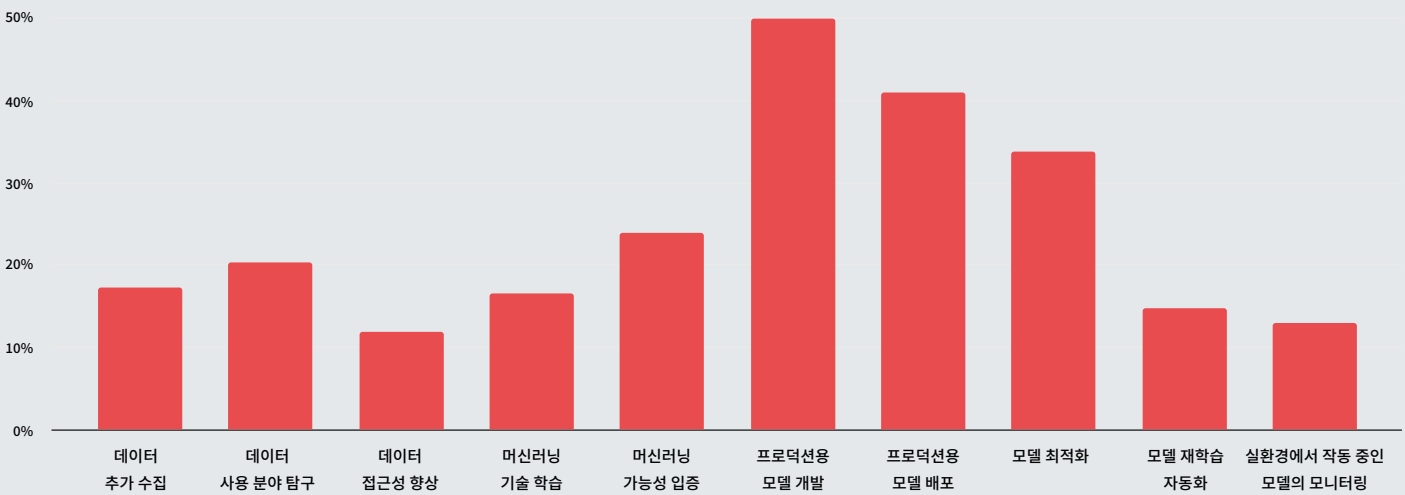
MLOps는 왜 중요한가? (서문)

지금까지, 머신러닝은 데이터 사이언티스트가 별도로 진행하는 개별 과학 실험의 관점에서 주로 진행됐다. 그러나 머신러닝 모델이 실제 문제 해결책의 일부가 되고 비즈니스에 중요해지면서, 우리는 과학 원칙을 평가절하하기보다 더 쉽게 접근하고 재현 가능하며 협력할 수 있도록 관점을 바꿔야 할 것이다.

2020년 5월, 다양한 분야의 기업에 종사하는 330명의 데이터과학자, 머신러닝 엔지니어 그리고 매니저를 대상으로 한 설문 조사에서, 그들에게 향후 3개월 동안 무엇에 중점을 둘 예정이며 지금까지 어떤 주요 장애물에 직면했는지 질문했다. 응답자 20%는 여전히 실험 및 학습 단계에 더 집중하고 있다고 답했지만, 응답자 절반은 상품화 목적의 모델 개발에 집중하고 있고 40% 이상은 상품화 목적의 모델을 배포할 거라고 답했다.

그림 1

출처: State of ML 2020, Valohai
응답자 330명



응답자 대다수는 모델 재학습 자동화와 모델 생산 과정 모니터링 관련 직무에 종사하고 있지 않으며, 이는 머신러닝이 상당 부분 비교적 실제 환경 적용 이전의 초기 단계에 머물러 있다는 걸 의미한다. 그러나 'ML옵스'라는 단어가 뉴스 기사에 더 자주 등장하고, 검색 엔진에서 검색량이 증가하고 있는 현상을 보면, 모델 재학습을 자동화하고 모델을 모니터링하는 일들은 점점 더 많아지고 있다. 실제 환경에서 동작하고 있는 모델은 데이터 사이언티스트뿐 아니라 엔지니어, 제품 관리자 그리고 규정 준수 담당자 등 다양한 이해관계자들을 포함하고 있는 팀에게 새로운 도전과제를 던지는데, 이는 공동으로 해결해야 한다.

대부분의 현실 적용 사례에서는 기초 데이터는 끊임없이 변하고, 이에 따라 피처 드리프트(feature drift) 문제 해결을 위해 모델 재학습이 필요하거나 전체 파이프라인을 재설계해야 할 수도 있다. 비즈니스와 규제 요구 사항도 빠르게 변경될 수 있기 때문에, 더 잦은 모델 릴리즈 주기가 요구될 수 있다. 이는 ML옵스가 운영 노하우와 머신러닝, 데이터 과학 지식이 결합되는 지점이라는 걸 의미한다.

머신러닝에서 2020년이 실제 환경에 적용할 모델의 해였다면 2021년은 ML옵스의 해가 될 거라고 예측한다. 본 자료에서 ML옵스에 대한 이해를 바탕으로 독자의 상황에 걸맞은 새롭고 참신한 아이디어를 얻어 갈 수 있기를 바란다.

머신러닝 프로젝트의 역할 체계

현실 환경에서 동작하는 머신러닝을 개발하는 프로젝트의 방대한 규모와 범위는 우리들 모두를 놀라게 했다. 데이터를 수집하고, 모델을 학습하고, 수익 창출을 위해 이를 응용하는 비교적 간단해 보이는 절차는 비즈니스-운영-IT 등 모든 분야의 리소스를 빨아들이는 거대한 블랙홀이 되곤 한다. 단일 프로젝트는 데이터 보관과 보안, 접근성 제어, 리소스 관리, 높은 가용성, 기존 사업과의 연동, 테스트, 재학습 등 수 많은 분야의 주제를 다룬다. 따라서 많은 머신러닝 프로젝트는 기업의 역사상 가장 범분야적이고 범조직적인 대규모 개발 프로젝트가 될 가능성이 크다.

여러 전문 분야에 걸친 머신러닝 프로젝트 특성을 이해하기 위해, 우리는 프로젝트에 연관된 다양한 역할들을 살펴보고자 한다. 이 역할 체계는 지난 4년간 스타트업부터 포춘(Fortune) 500대 기업까지 약 500여 개 다양한 분야의 조직을 연구한 결과에서 얻은 고유하고 포괄적인 시각이 반영됐다. 그 목록의 역할이 완벽하지 않을 수 있지만, 머신러닝 프로젝트와 연관된 인력의 전체 골자를 볼 수 있다.

주의할 점은 구성원 1명의 역할이 하나로 제한된다고 볼 수 없으며 1인이 다수의 역할을 수행할 수도 있다는 것이다. 그리고 작은 조직에선 구성원 1인이 종종 다수의 역할을 수행할 수밖에 없다는 점을 기억하자. 예를 들어, 많은 데이터 사이언티스트가 머신러닝 엔지니어링 업무를 함께 처리하고 때론 데브옵스(DevOps)와 IT 업무가 동일시되는 경우도 있다. 중요한 건 조직 규모가 커질수록, 개인 구성원의 역할 범위가 더 전문화되고 고립되기 때문에 조직 내 협업이 더욱 요구된다.

데이터 사이언티스트

데이터 사이언티스트는 데이터 분석에 기반해 다양한 비즈니스 문제에 솔루션을 찾는 업무를 수행한다. 데이터 사이언티스트는 수학, 통계, 컴퓨터 과학 또는 공학 분야에서 석사 이상의 학위를 수료한 고학력자인 경우가 많다. 이들은 데이터 랭글링(wrangling), 통계 분석, 데이터 시각화 및 머신러닝 응용 분야에 강점을 갖고 있다.

일반적으로 데이터 사이언티스트는 근본적인 비즈니스 문제를 해결하기 위해 기존 데이터를 분석하고 패턴을 찾는 업무에 먼저 투입된다. 예를 들어, 사용자 데이터를 분석하여 유의미한 사용자 세그먼트를 찾은 뒤 해당 사용자를 적합한 세그먼트로 분류하는 모델을 구축한다. 그리고 최종 사용자 경험을 차별화하고 사용자 활동을 증진시키는 일에 기여한다. 데이터 사이언스의 주요 목적은 데이터를 탐구하고 모델을 구축하는 일이지만, 데이터 사이언티스트는 작업 중 상당 시간을 이용 사례에 맞는 데이터 조작 업무에 사용한다.

데이터 엔지니어

데이터 엔지니어는 데이터가 정상적으로 저장돼 데이터 사이언티스트의 접근이 가능하도록 인프라를 책임진다. 이들은 다양한 경로를 통해 확보한 원시 데이터를 모아 중앙 데이터 레이크에 저장하는 시스템을 구축한다. 또한, 데이터 레이크에서 데이터를 가져오는 별도 데이터 창고(warehouse)를 구축할 수 있고, 이를 최종 사용자에게 더 편리한 형태로 변환해 제공한다. 예를 들어 이미지 데이터의 크기를 조절하거나 파일 형식을 변경하여 데이터 사이언티스트가 데이터 조작보다 데이터 분석에 더 집중할 수 있도록 돕는다.

머신러닝 엔지니어

머신러닝 엔지니어의 역할은 아직 초기 단계지만, 기업이 머신러닝으로 실질적 효과를 누릴 수 있도록 주도하면서 그 중요도가 커지고 있다. 데이터 사이언티스트는 이론적 개념을 탐구하고 증명하는 반면, 머신러닝 엔지니어는 이 개념들을 실 서비스에 배포 가능한 규모의 시스템으로 구축하는 업무에 집중한다. 제목에서 알 수 있듯이 머신러닝 엔지니어는 머신러닝의 이론적 이해를 바탕으로 지속적 통합(CI), 지속적 제공(CD)과 같은 개발 영역 패러다임에 더 밀접한 업무를 수행한다. 이들은 모델 배포 전의 학습 및 테스트 자동화와 스케일링, 그리고 실 환경에 적용할 모델의 배포 및 모니터링 업무를 맡는다. 이들은 종종 노트북을 스크립트로 변형하는 일처럼 데이터 사이언티스트가 작성한 코드를 재작성하고 재설계한다.

데브옵스 엔지니어

데브옵스 엔지니어는 소프트웨어 공학과 클라우드 인프라를 결합한 역할을 담당한다. 전통적인 소프트웨어 프로젝트에서 데브옵스는 테스트와 신규 코드 배포 관련 자동화 업무를 수행하며, 보통 테스트와 프로덕션 환경을 관리하기도 한다. 그러나 머신러닝 프로젝트에서 데브옵스는 머신러닝 모델과 최종 사용자를 위한 어플리케이션의 연결고리가 된다. 예로, 웹 어플리케이션에 신규 추천 엔진을 배포하는 상황이 있다. 이들은 주로 사용자용 어플리케이션의 확장성, 안정성 그리고 반응성을 최우선 순위로 고려하고 머신러닝 모델로 인해 이런 요소들이 저하되지 않는지 확인하는 업무를 맡는다.

IT

IT 운영자는 대부분의 대규모 조직에 존재하는데 이들은 일반적으로 자원 관리, 접근성 제어와 정보 보안 분야 업무를 맡는다. 종종 관료적으로 인식되기도 하지만, IT 운영과 그에 연관된 모든 절차는 컴퓨팅 자원 제공과 외부 업체 승인 요청 등 프로젝트가 차질 없이 진행되도록 지원하는 역할을 한다.

비즈니스 책임자

머신러닝 분야에서 비즈니스 책임자의 역할을 칭하는 구체적 명칭은 없으나, 비즈니스 책임자는 프로젝트 성공을 위해 꼭 필요한 존재다. 비즈니스 책임자는 구성원 중 최종 사용자를 가장 깊게 이해하고 머신러닝 시스템이 정확하게, 잘 만들어지도록 도울 뿐 아니라 실제 사용 가치를 갖도록 팀을 지도한다. 이들은 어떤 종류의 예측이 실용성을 갖는지 정의하고, 운영과 규제 측면에서 어떤 부분에 리스크가 존재하는지 이해할 수 있도록 돕는다.

관리자

머신러닝을 다루는 팀은 비교적 신생 팀인 경우가 많아 관리자 역할이 다소 불분명할 수 있다. 관리자 역할에서 중요하게 고려돼야 하는 부분은 머신러닝 프로젝트의 큰 복잡성과 다분야적 특성이다. 프로젝트 실행에 요구되는 자원이 단일팀에 속해 있지 않은 때가 많고, 이에 따라 관리자는 조직의 여러 다른 파트에서 자원 확보를 조율해야 한다. 팀의 업무 영역이 조직의 손익과 직접적으로 연관되지 않을 가능성이 높기 때문에, 관리자로서 팀 업무의 투자 대비 효과(ROI)를 이해하고 구성원과 소통하는 일이 어려울 수 있다.

머신러닝은 기존의 소프트웨어와 어떻게 구분되는가?

앞서, 머신러닝 개발이 다양한 분야에 걸쳐 있다는 특성으로 인해 인해 조직 관리에 새로운 형태의 질서가 요구된다는 내용을 살펴보았으나 이는 일부에 불과하다. 머신러닝을 기존 소프트웨어 영역과 구분짓는 또 하나의 측면은 바로 데이터다. 단순하고 자명하다고 인식될 수 있으나, 데이터(또는 일반적으로 빅데이터)의 등장은 기존 개발 프로세스에 완전히 새로운 도전과제를 가져온다.

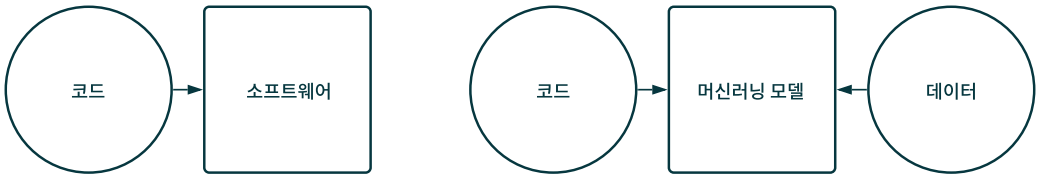


그림 2

소프트웨어는 일반적으로 코드 변경이 최종 결과물에 어떤 영향을 미치는지 거의 즉각적인 피드백을 받도록 로컬 환경에서 개발될 수 있다. 반면 머신러닝에선 코드 변경의 영향을 확인하려면 모델 재학습 과정이 필수다. 방대한 데이터셋으로 학습된 모델로 작업할 때, 이런 재학습 과정은 개발자에게 매우 오랜 시간을 대기하게 하거나, 원격 컴퓨팅을 사용해야 하는 등 인프라 측면에 큰 장애가 될 수 있다. 앱 개발자가 코드 변경 후 브라우저에서 새로 고침을 누르는 상황과 데이터 사이언티스트가 유사한 코드 변경 후 결과 확인을 위해 GPU 클러스터 가동, 코드 배포, 데이터 전송 그리고 모델 훈련 등 일련의 과정을 거치는 상황을 상상하면 우리는 그 차이를 쉽게 이해할 수 있다.

그러나, 데이터라는 요소로 인해 개발 과정에서 코드가 동작하는 방식이 변화하는 것 뿐만 아니라, 데이터 그 자체가 변화할 수 있는 주체라는 점도 고려해야 한다. 전통적 소프트웨어 개발 방식에선 하나의 코드 버전이 하나의 소프트웨어 버전을 생성했으나, 머신러닝에선 하나의 코드 버전과 하나의 데이터셋 버전이 결합해 하나의 머신러닝 모델 버전을 생성하기 때문이다.

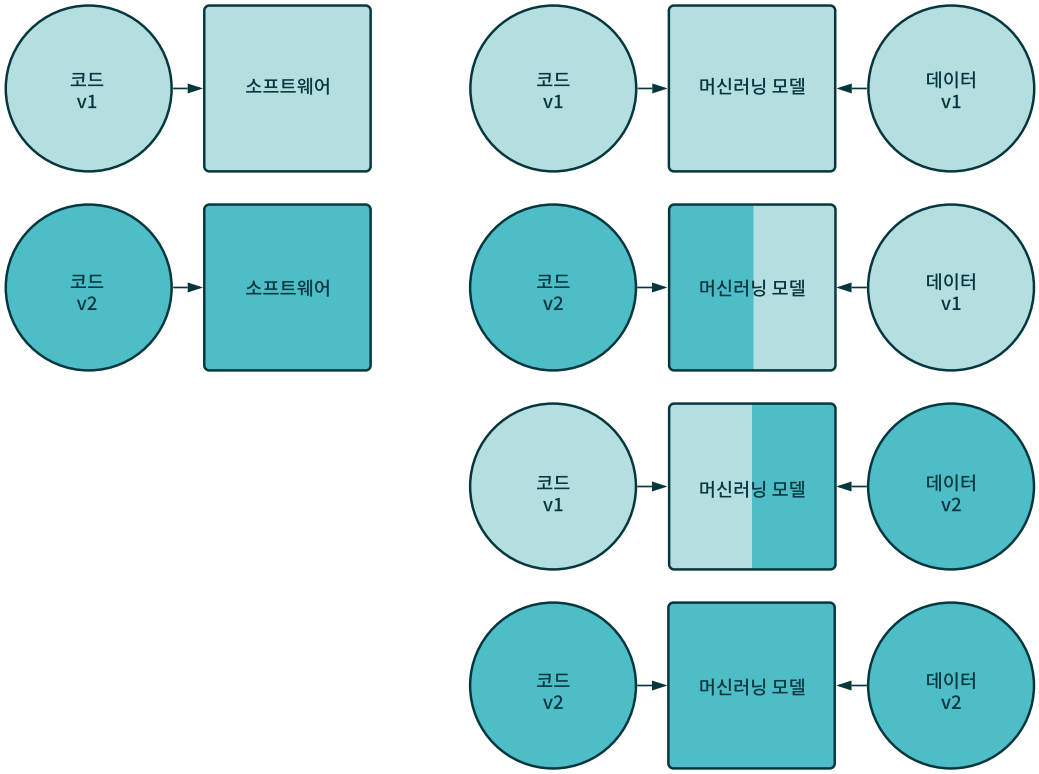


그림 3

데이터는 개발 프로세스에서 고려해야 할 새로운 변수로, 개발 결과를 재현하려면 사용하던 기존 데이터를 똑같이 재현해야 하므로 개발 복잡도를 상당히 높인다. 따라서 모델의 모든 가능한 변형을 만들지 않더라도, 재현을 위해선 모델 구성 요소에 대한 버전 관리가 필수다. 데이터 준비, 증강, 생성 등 여러 단계를 포함하는 전반적인 머신러닝 시스템을 고려하기 시작하면 코드와 데이터 두 가지 모두의 버전을 관리하는 복잡도는 더욱 증가한다.

ML옵스 워크플로우

ML옵스의 목적

이 글을 읽고 있다면 ML옵스의 필요성을 이미 직감할 것이다. 이 직감은 올바른 방식으로 업무를 진행하고자 하는 의지와 계속되는 일상적 업무에서 오는 기술적 어려움일 가능성이 크다.

하지만 일정 시간과 돈의 투자를 승인받기 위해 팀원과 이야기할 때, 직감에만 의존한 논쟁은 어려울 수 있다.

이번 장에선 ML옵스에 대한 개념을 구체화해서, 왜 이 분야에 집중할만한 가치가 있는지 이해를 돕고, 또 그만한 비용 투자를 어떻게 정당화 할 수 있는지를 다룰 것이다. 이번 장은 또한 우리가 ML옵스의 어떤 세부분야에 집중할지 평가에 도움이 될 것이고, 투자 대비 효과(ROI) 측정과 달성 방안은 다음장에 자세히 서술하겠다.

최신 자료에서 모델의 생애주기(lifecycle)을 설명할 때 다음과 같은 그림이 자주 등장한다.

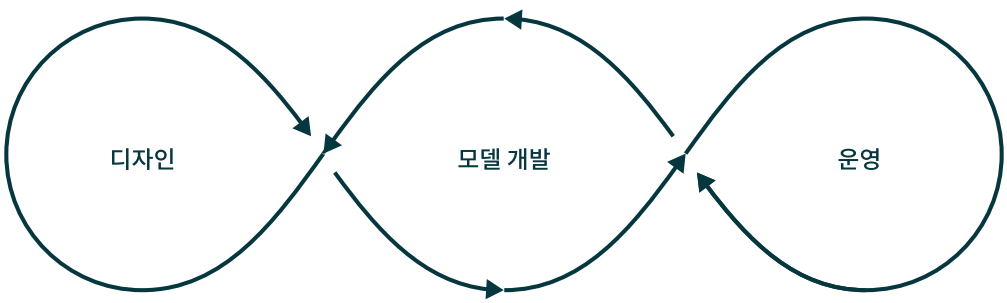


그림 4

그러나 실제로 많은 조직의 모델 개발과 운영 사이의 업무 연계가 복잡하고 비효율적인 사이클 안에서 운영된다.

연구환경에서 개발하는 것이 재미는 있겠지만, 현실에선 외부와 단절돼 독자적인 상태로 개발하는 게 아니며, 상용화된 모델만이 가치를 창출할 수 있다는 점은 자명하다. 즉, 모델의 ROI는 실제 환경에서 사용되기 전까지는 '0'이라고 볼 수 있다. 따라서 출시까지 소요되는 기간(Time to market)은 머신러닝 프로젝트에서 가장 중요하게 고려하고 최적화해야 할 지표다.

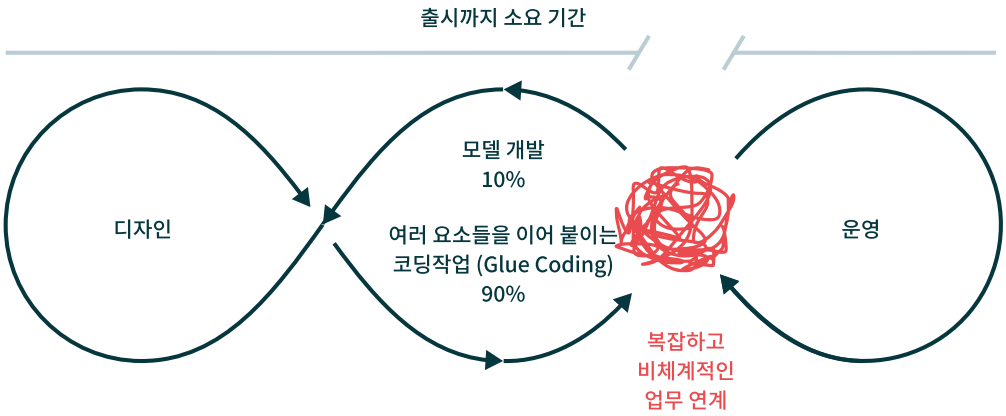


그림 5

> **ML옵스의 목표는 가장 빠른 시간내에, 가장 적은 위험을 부담하며 아이디어 단계부터 상용화 단계까지 ML 프로젝트를 진행할 수 있도록 기술적 마찰을 줄이는 것이다.**

위 문장을 자세히 살펴보면 출시까지 소요 시간 단축, 위험 부담 감소 두 부분으로 나눌 수 있다. 이 두 가지 목표는 데이터 사이언티스트와 비즈니스 관계자가 동일 선상의 입장에서 소통하고, 사업 관점에서 ML옵스를 필수 요소로 인식하도록 유도한다. ML옵스로 수작업을 자동화하고 코드 품질을 향상할 수는 있지만, ML옵스 자체가 조직 전체의 자발적 참여를 이끄는 근본적 목표가 되기는 어렵다.

다음으로 위험 요인과 출시 시기를 살펴보자. 위험요소를 평가하는 것은 개념상 복잡하지 않아 먼저 설명하겠다.

머신러닝의 위험 요인

ML옵스의 목표는 가장 빠른 시간내에, 가장 적은 위험을 부담하며 아이디어 단계부터 상용화 단계까지 ML 프로젝트를 진행할 수 있도록 기술적 마찰을 줄이는 것이다.

ML옵스 워크플로우에서 다루는 세 가지 주요 위험 유형을 확인했다.

1. 지식 상실
2. 상용화 실패
3. 규제와 윤리

위에서 나열한 각 주제는 산업별, 구현 방식별, 심지어 회사별로 다뤄야 할 세부 내용이 방대하나 머신러닝의 상용화에서 발생하는 가장 큰 위험요인들은 대부분 위 분류 중 하나에 속한다.

참고로, 데이터 보안 같은 머신러닝에 국한되지 않는 기본적인 IT 위험 요인은 기존의 많은 문헌에서 잘 알려져 있기에 별도로 다루지 않았다. 예를 들어 외부인이 데이터베이스에서 데이터를 탈취해선 안 된다는 건 자명한 사실이고, 머신러닝뿐 아니라 모든 데이터베이스에서 동일한 내용일 것이다.

지식 상실 위험

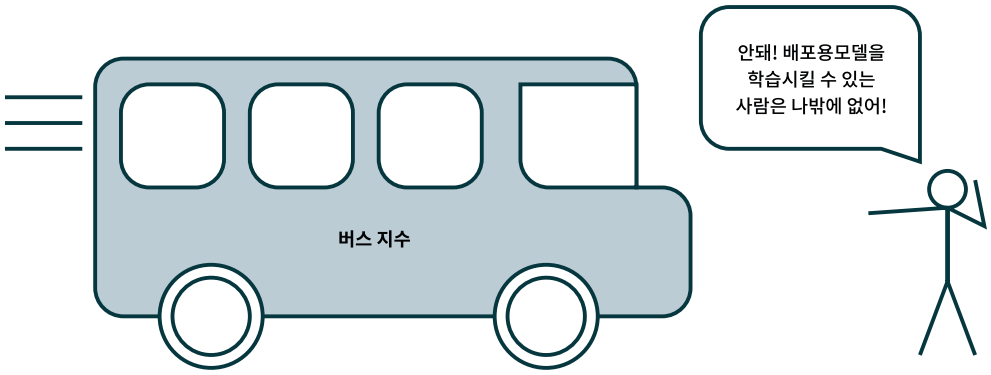


그림 6

버스지수: 팀원 간 공유되지 않는 정보와 기능으로 인한 위험을 버스에 부딪힐 경우로 비유해 측정한 것. 프로젝트가 중단될 위험을 고려한 팀 구성원의 최소 단위. (편집자)

버스 지수는 일반적으로 소프트웨어 공학에서 사용되는 용어로, 핵심 인력이 마치 버스 교통사고 같은 갑작스러운 원인으로 프로젝트에서 제외되면서 발생하는 위험 요인을 의미한다. 이는 기존 소프트웨어 개발 분야에서 많이 다뤄진 문제지만, 특히 머신러닝 분야에서 버스 지수는 극대화된다.

머신러닝에는 어느 정도 자체적으로 문서화가 가능한 코드 이외에 데이터라는 요소가 존재한다. 단일 데이터 세트에는 수집 경로와 추출된 피처(feature) 등 많은 양의 정보가 숨어 있을 수 있다. 단일 데이터 세트를 얻기 위해 다양한 버전의 노트북과 스크립트가 사용되었을 수 있다.

지식 상실은 여러 방식으로 일어날 수 있다. 제일 쉽게는, 데이터 사이언티스트가 회사를 그만두는 일인데 이는 어느 정도 규모를 갖춘 조직은 대다수 겪어본 일일 것이다. 그러나 이 문제는 인사 변경 없이도 발생할 수 있다. 여러 프로젝트에 오랜 기간 걸쳐서 업무를 진행하다 보면 본인이 담당했던 업무의 상세 내용을 잊어

버리는 경향이 있다. 반년 전 작업했던 프로젝트로 다시 돌아와 오류 없이 모델을 재학습(retrain)시키는 일엔 어려움이 따른다. 실제로 많은 이들이 기존에 작동하고 있는 모델을 혹여나 망가뜨릴까 봐 추가 작업을 꺼려하고 있을 것이다.

전통적인 소프트웨어 개발 방식에선 깃(Git)과 같은 검증된 버전 관리 도구로 재현 가능성을 보장한다. 반면 머신러닝에선 적절한 버전 관리와 재현 가능성을 달성하기 위해 기존 방식 외에 추가 조치가 필요하다.

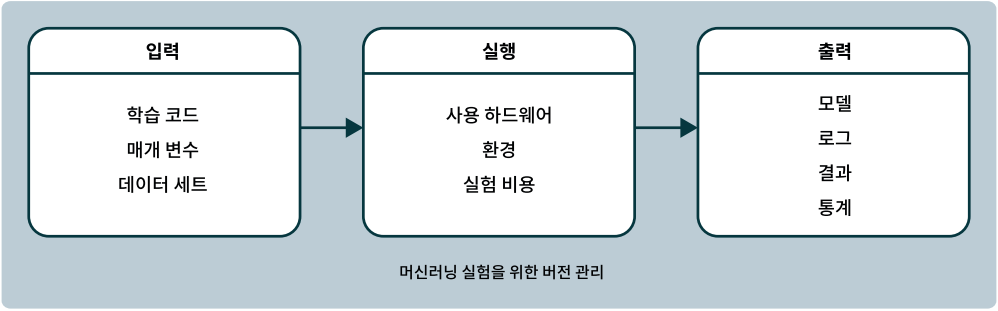


그림 7

머신러닝에서 모델이란 코드, 데이터, 매개 변수 그리고 학습 환경을 조합한 개념이다. 타인이 작업 도중 그만두었거나 오래전 본인이 작업했던 모델을 이해하고 개발하려면 깃 저장소만으론 턱없이 부족하다.

코드가 사용하는 데이터가 무엇인지 알아야 하고 어떻게 생성된 데이터인지 알아야 한다. 주로 여러 개의 스크립트로 서로 다른 데이터 소스에서 데이터를 불러와, 피처 결합(feature aggregation)하는 방식으로 필요한 데이터 세트를 얻는다. 다른 사람이 문서화하지 않은 채 작업한 여러 스크립트와 데이터 소스를 취합하고 이어가는 일은 처음부터 시작하는 것보다 더 복잡할 수 있다. 이 문제를 해결하려면 원시 데이터부터 코드, 환경, 구성, 매개변수 등 모델까지 포함한 전체 파이프라인을 추적할 수 있는 버전 관리 시스템 도입을 권장한다.

프로덕션 실패 위험

다음은 프로덕션 실패 위험이다. 소프트웨어 개발자라면 결함이 있는 코드를 프로덕션 환경에 배포하는 것이 시스템 오류의 주요 원인이 된다는 것을 잘 알고 있다. 따라서 여러 관계 조직들이 막대한 시간을 들여 지속적 통합과 지속적 전달(CI/CD) 파이프라인을 구축하는 일은 개발자가 최신 업데이트 배포 후, 그 업데이트가 제품을 망가뜨리지 않을 거라는 확신을 갖도록 돕는다.

이런 상황은 머신러닝에도 동일하게 적용된다. 배포 후 실패 요인은 가장 간단한 문법 오류부터 데이터 스트림 변경, 예상치 못한 모델의 성능 변경까지 다양하다.

이상적인 환경에서 ML옵스라는 도구는 파이프라인별 코드와 데이터에 대한 다양한 테스트 방식을 지원할 뿐 아니라 머신러닝 프로젝트의 설계 단계부터 이 테스트를 코드화하는 방법을 채택한다.

머신러닝 워크플로우의 여러 세부 부분과 관련 테스트 방법은 다른 장에서 다루겠지만, 아래는 참고할만한 몇 가지 주요 사항들이다.

1. 모델 학습용 데이터가 의도한 형태와 같은지 확인해야 한다. 데이터 수집과 저장 같은 이전 단계에서 변경됐을 수 있기 때문이다.
2. 학습 환경뿐 아니라 실제 환경에서도 모델의 정상 동작 여부를 확인해야 한다. 과적합(overfitting)은 실질적 위험 요인으로 파이프라인과 프로세스가 이를 고려하도록 설계되어야 한다.
3. 인프라가 일관되게 작동하도록 확인해야 한다. 머신러닝 파이프라인은 입력값이 같다면 항상 동일한 결과를 내야 하며, 원하면 언제든지 롤백(roll back)할 수 있어야 한다.

규제, 윤리적 위험

머신러닝 알고리즘은 사람이 명확하게 정의하지 않기 때문에 엄격한 규제와 도덕적 기준이 적용되기도 한다. 이런 위험이 현실화하면 금전적 손실뿐 아니라 더 치명적인 결과로 신뢰와 명성을 잃게 되기도 한다.

따라서 데이터 사이언티스트는 잦은 관리 감사로 많은 사람을 불안하게 만들 수 있다. ML옵스의 모범사례(best practices)를 적용하면 재현 가능성이 최우선시되며 머신러닝 모델의 모든 구성요소에 버전 관리 체계를 도입할 수 있다.

이는 마치 회계 업무에서 장부만 잘 정리돼 있으면 별다른 걱정이 없는 것과 비슷하다. 가장 중요한 점은 누군가 프로덕션 모델 예측에 의문을 제기하면, 모델이 학습된 과정으로 언제든 역추적할 수 있다는 점이다.

ML옵스 도구 대부분은 버전 관리를 워크플로우에 도입할 수 있도록 지원하지만, 이는 규제 위험을 최소화하는데 절반에 불과하다. 나머지 절반은 프로덕션 모델이 편향되지 않도록 하는 일인데, ML옵스 도구는 프레임워크만 제공하기 때문에 실제 올바른 규칙을 적용하려면 관련 분야 전문 지식이 요구돼 훨씬 까다롭다.

편향된 모델이 기술적 관점에선 실패한 게 아닐 수 있지만, 규제 관점에선 실패를 초래할 가능성이 높다. 따라서 머신러닝 파이프라인에서 편향성 및 기타 윤리 문제 방지를 위한 테스트를 기술 테스트에 포함하는 일을 고려해야 한다. 이런 윤리적 문제는 애플리케이션에 따라 크게 달라진다. 예를 들어, 헬스케어 관련 애플리케이션과 금융 관련 애플리케이션은 확연히 다른 윤리적 문제가 존재할 것이다. ML옵스로 모델 편향성 문제 자체를 해결할 수는 없지만, ML옵스는 모델 편향성을 검증하기 위한 테스트를 머신러닝 파이프라인에 도입하는 방식으로 이 문제를 해결하고자 하는 개념이 내재돼 있다. 학습용 데이터 형태와 기대한 예측 결과에 대한 검사와 안전장치를 마련한다면 관리 감사가 훨씬 덜 위협적으로 다가올

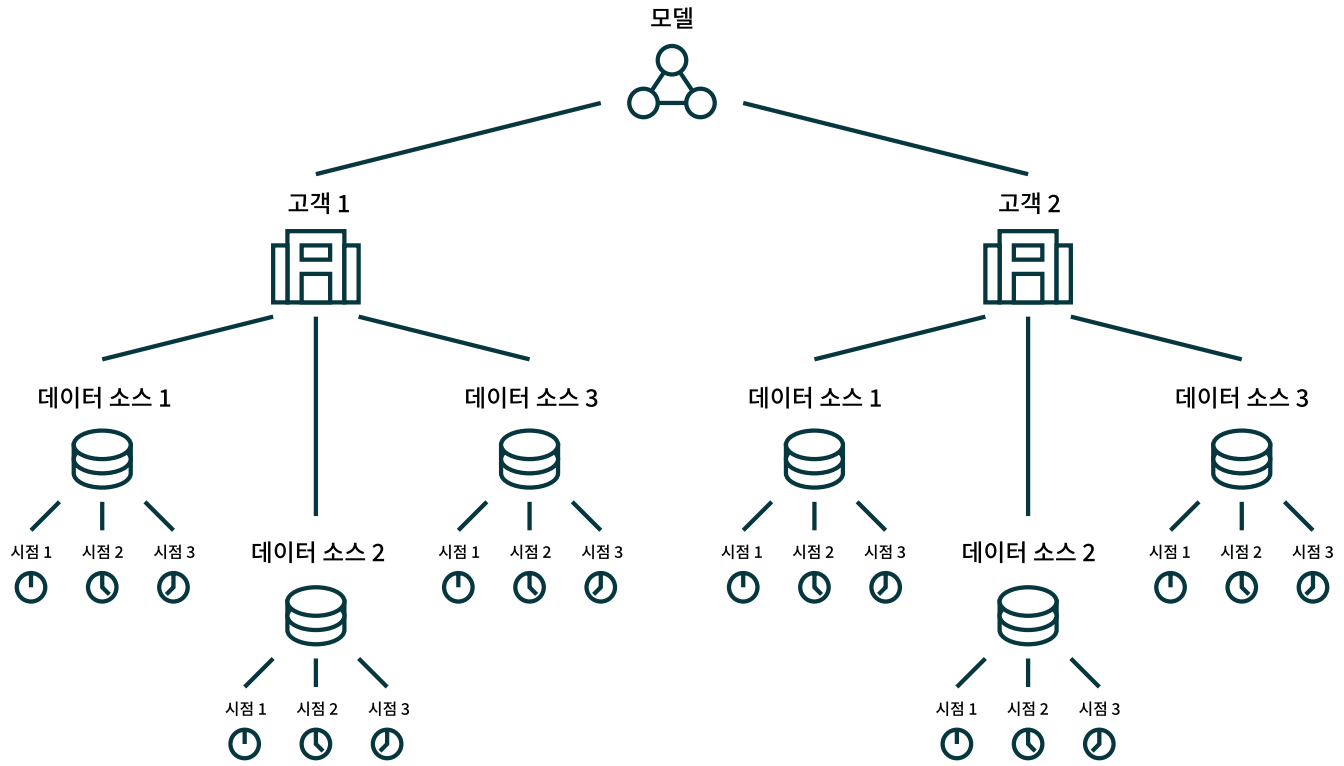
것이다. 모든 프로덕션 모델은 안정적이고 신뢰성 있는 규칙을 따라 제작될 것이기 때문이다.

출시 기간

머신러닝 모델을 처음 개발한다면 해당 프로젝트 만을 위한 임시방편적인 워크플로우(ad-hoc workflow)로 작업하는 게 출시 기간을 단축하는 가장 빠른 방법일 것이다. 많은 데이터 사이언티스트와 엔지니어가 업무 프로세스에 엄격한 규범 도입을 꺼리는 일도 이런 이유 때문이다. 하지만 프로덕션용 머신러닝에서 단일 모델의 단일 버전이 프로덕션에 배포된 뒤 끝나는 일은 매우 드물고, 장기적인 관점에서 지속 발전할 수 있는 머신러닝 역량을 구축하는 일이 더 중요하다.

많은 경우에 우리는 모델 구축 단계를 다양한 방면에서 고려해야 한다. 무엇보다도 한 버전의 모델이 장기간 동안 작동하는 경우는 드물며, 보통 어느 정도 정기적으로 재학습이 필요하다.

두 번째로, 모델 구축 이외에도 고민해야 하는 여러 다른 방면이 존재한다. 예를 들어, 고객의 특정 데이터 또는 지리적 정보 등을 기반으로 모델을 재학습해야 하는 때가 종종 발생한다. 여러개의 동일한 아키텍쳐를 가진 모델이 각각 다른 버전과 다른 종류의 데이터 세트로 주기적으로 재학습 되어 동시에 프로덕션에 실행되고 있을 가능성도 있다. 결과적으로 기하급수적인 재학습이 빠르게 진행되고 임시방편적인 워크플로우로 인한 혼란과 고객 불만족, 부진한 결과가 발생한다. 머신러닝을 기반으로 한 확실한 확장 가능한 사업 모델을 발굴하려면 제품을 단일 개체의 학습 모델로 바라보기보다 파이프라인 전체를 바라보는 넓은 시야가 필요하다. 아래 그림은 현재 머신러닝 분야에서 가장 성공적인 기업들의 체계이다.



18개의 학습 된 모델

그림 8

일반적으로 ML옵스는 머신러닝 출시 기간을 단축하는데 두 가지 방법을 제공하다.

1. 다양한 이해 관계자들 간의 공용어 역할을 한다.
2. 수작업을 자동화한다.

공통 언어

지난 몇 년간 소프트웨어 개발 속도와 효율이 급격히 증가했다. 기업은 최신 도구와 업무 공유 방법론(CI/CD, 버전 관리, 마이크로서비스)으로 소프트웨어 개발 생산성을 기하급수적으로 향상했다.

공통 언어는 개발 가속화가 필요한 상황에서 신규 인력의 빠른 실무 투입을 가능하게 한다. 이와 유사하게 머신러닝 분야에서 공통 언어는 개발 생산성을 기하급수적으로 향상하는 주요 조건 중 하나로 볼 수 있다.

두 번째, 위에서 언급한 대로 ML옵스는 모델 제작 과정을 일종의 머신러닝 파이프라인 형태로 바꾸는 과정을 중요하게 다룬다. 이를 달성하려면 관련 조직은 업무를 시스템화하고 구성 부품처럼 나눠야 한다. 단일 노트북과 일부 수작업에 의존하는 것과 비교한다면 반드시 추가적인 노력이 필요하지만, 업무를 구성요소별로 세분화하는 일은 업무 확장성에 차원이 다른 결과를 가져온다. 이는 대규모 개발 프로젝트에서 마이크로 서비스 방식이 달성한 결과와 비슷하다.

한 명의 데이터 사이언티스트에게 의존해 전체 모델을 제작할 필요는 없으며, 팀 단위에서 데이터 전처리, 모델 학습, 테스트 등 세부 업무별로 분담 작업이 가능하다. 모델 제작의 복잡도가 커질수록 파이프라인에서 공동 작업하는 방식의 이점도 함께 증가한다.

또한, 다양한 코드 조각을 포함한 파이프라인은 일정 수준 자체 문서화하는 효과가 있어 신규 담당자에게 업무 인계 시 수작업 대비 편리한 측면이 있다.

자동화

데브옵스는 소프트웨어 개발자 업무 사이클을 월 단위 또는 분기 단위에서 일 단위 또는 주 단위 사이클로 단축했다. 앞서 언급한 대로 머신러닝은 코드 외에도 고려할 다양한 변수가 존재하는데, 이런 복잡성에도 불구하고 지속적으로 진행 상황을 최적화해야 한다.

머신러닝에서 지속적 통합과 지속적 전달(CI/CD) 파이프라인을 구축하는 건 전통적인 소프트웨어 분야보다 어렵지만 같은 수준의 이점이 적용된다. 데이터 수집, 학습, 모델 평가 등 업무의 자동화는 가장 부족한 리소스, 즉 데이터 사이언티스트가 추가 개발에만 집중할 수 있도록 돕는다. 게다가 이런 파이프라인은 일회용이 아니라 다른 용도로 신규 모델 개발 시 부분별 재활용 가능한 장점이 있다.

다시 한번 강조할 점은, 짧은 출시 사이클의 장점은 사용 사례마다 다르기 때문에 중요도 역시 애플리케이션에 따라 다르다는 점이다. 그러나 가장 극단적인 경우를 보면, 모델의 기반이 되는 데이터가 변화하는 속도에 맞추어 모델이 상시 업데이트 되어야 할 필요할 수 있다. 이것이 Facebook, Uber 같은 선구적인 회사가 ML옵스라는 용어가 만들어지기 전에부터 자체 ML옵스 인프라를 구축한 이유 중 하나다.

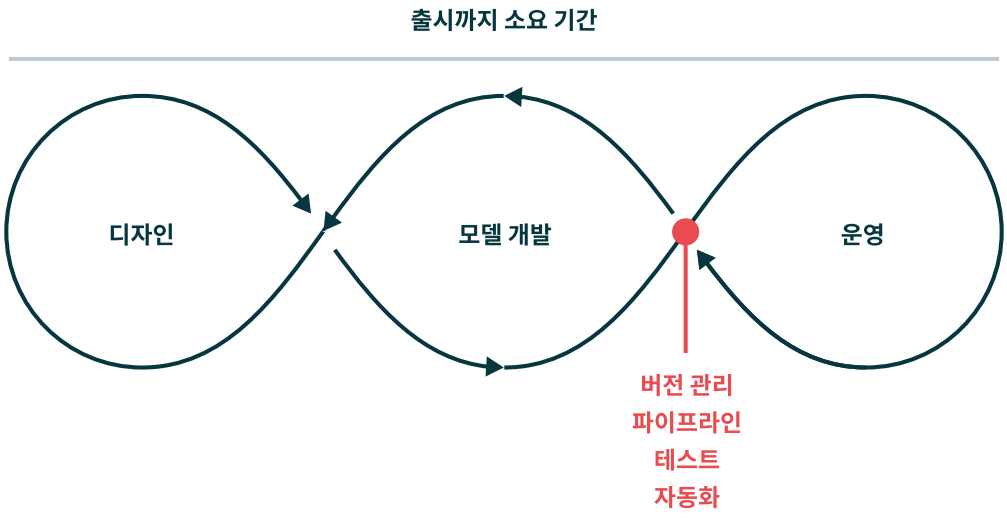


그림 9

ML옵스 워크플로우 모범 사례

ML옵스는 일련의 우수 방법론을 이용해 모델 개발과 운영 부분을 지속적인 하나의 과정으로 통합한다. 이런 모범 방법론은 다양한 방법으로 시행될 수 있으며, 그 중 가장 빈번한 방식은 공통된 하나의 플랫폼을 사용하는 것이다. 우리는 가능한 한 위험을 최소화하면서 가능한 한 짧은 시간 안에, 아이디어에서 프로덕션으로 모델을 가져오는데 드는 기술적 마찰을 줄이는 일에 도움이 되는 네 가지 모범 사례를 확인했다.

- 모델, 코드, 데이터, 매개 변수, 환경 등 모든 요소의 버전을 관리하자.
  누구나 모델 제작 과정을 추적할 수 있게 하자.
- 모델 제작 과정을 단계별로 부품화해 하나의 파이프라인을 구축하자.
  단일 노트북은 파이프라인이라고 할 수 없다.
- 테스트 과정을 시스템화하자.
  체크 포인트와 안전장치로 모델이 준수해야 하는 표준을 세우자.
- 업무를 자동화해 향후 개발 업무에 사용할 시간을 확보하자.

ML옵스 프로젝트 성과 측정 방법

성과 측정이란 비단 비즈니스 성과에만 국한되지 않는다. ML옵스의 현 발전 단계에서, 성과 측정은 여러 기능의 협업과 학습 내용 문서화 그리고 절차상 성과까지 모두 고려돼야 한다.

> "프로덕션에서 실행되는 모델만이 가치를 제공합니다."

단순히 실험하는 수준의 머신러닝 프로젝트를 넘어 프로덕션 시스템에 실제 비즈니스 가치 제공을 목표로 하는 프로젝트로 변하면서, 프로젝트 난이도는 그만큼 높아진다. 이와 함께 프로젝트 성공 불확실성도 커진다. 이에 대처하는 방법으로 아래 세 가지를 살펴보자.

1. 모든 이해관계자와 함께 목적을 정의하며 명확하고 구체적이고 측정 가능한 평가 지표를 설정한다.
    - 모든 이가 프로젝트의 실행 목적과 기대 효과를 이해하도록 돕고, 추후 머신러닝 프로젝트 평가의 토대가 될 기본 전제를 숙지할 수 있도록 공유한다. 예를들어, 모델 정확성 측정 방식과 과적합(over-fitting, 오버피팅), 과소적합(under-fitting, 언더피팅) 개념에 대한 기본 지식을 갖춰야 한다.
    - 조직의 현재 업무 수행 방식과, 이를 개선하기 위해서는 어떤 것이 필요한지 문서로 만들어 공유해야 한다.
    - 특정 문제의 다양한 선입견을 발견하고 이를 해결하기 위한 세션을 주최해 다양한 조직 구성원 사이에 존재하는 여러 가정을 파헤친다. 예를 들어, 데이터 소스 선별, 데이터 처리 방식, 모델 출시 방식과 관련한 다양한 가정이 있을 수 있다. 이런 가정을 사실로 받아들이기 전에, 소규모 실험을 통해 가설을 검증해야 한다.

2. 최종 결과뿐 아니라 프로젝트의 각 진행 단계별 성과를 측정한다.

- 우리가 일정을 계획할 때, ML옵스 프로젝트 전체 일정을 한꺼번에 산정하는 일은 어렵다. 성공 여부가 불확실한데도 6~12개월씩 소요될 프로젝트에 과감히 비용 투자를 정당화하기란 쉽지 않을 것이다.
- 프로젝트를 여러 단계로 세분화해 가설을 검증하면서 진행 상황을 추적하고, 피드백을 수집하는 일이 필요하다. “자주 출시하고 피드백을 기반으로 개선을 반복하라”라는 유명한 방법론이 동일하게 적용된다. 단, 업무를 장기적으로 바라보지 말라는 건 결코 아니며, 진행 상황을 주기적으로 (예를 들면 매 6주 마다) 가시화할 수 있어야 한다.
- 프로젝트를 단계별로 진행하면서 최초에 의도한 방향으로 문제를 해결하고 있는지 그리고 진행 상황을 회고한다. 이때 문제를 재정의하는 일을 두려워해선 안 된다. 배운 내용을 문서화해 다른 조직이 참조해 배울 수 있도록 내부 위키 공간에 정리한다. 내부 데이터 접근 정책, 최초에 식별하지 못한 내부 프로세스 그리고 자체 인프라 한계와 관련한 내용이 교훈이 될 수 있다.

3. 역할과 책임이 명확한 종합 팀을 구축한다.

- 성공 평가 지표 중 한 가지로, 조직의 협력 수준을 측정해야 한다. ML옵스 프로젝트에서 가상의 종합 팀을 구축하면 조직 내 다양한 영역에서 발생하는 예상치 못한 장애 요소를 효과적으로 해결할 수 있다.
- 모든 구성원이 프로젝트에서 서로의 역할과 책임을 확실히 이해하도록 한다. 명확한 이해를 바탕으로 해야, 더 원활한 협력과 업무 추진이 가능하고 불필요한 절차를 줄일 수 있다.
- IT 조직을 조기에 참여시킨다. 이들은 오랜 기간 축적된 경험을 바탕으로 신뢰성 높은 성숙한 인프라를 제공할 수 있다.

- IT 조직은 모델이 자동화되길 바랄 것이다. (예: 전처리, 학습, 빌드, 배포의 자동화)
- IT 조직은 대개 데이터 규정과 접근 관련 내용에 밝은 편이다.

공통 언어로 소통하라

데이터 사이언티스트의 워크플로우는 내부 데이터베이스에서 데이터를 추출하고, 데이터를 탐색하며 모델을 개발해 다른 구성원이 활용할 수 있도록 클라우드 공간에 저장하는 업무를 포괄한다. 모델 성능 기준은 대체로 비즈니스 책임자의 최초 요구에서 시작하고, 정확도(Accuracy), 정밀도(Precision), 재현율(Recall) 같은 모델 평가 지표로 구성될 가능성이 크다.

하지만 이런 과정을 프로덕션 환경에 적용하는 과정은 단순히 모델 파일을 프로덕션 서버에 복사해 붙여넣는 일 같은 간단한 과정이 아니다. 대부분의 조직에서 프로덕션 배포 과정은, 성숙한 인프라와 업무 수행 방식을 갖춘 경험 많은 IT (또는 운영) 부서가 관리한다. 이들은 신규 데이터를 사용한 모델 재학습부터 기존 모델과의 성능비교, 프로덕션 환경에 배포 등 머신러닝 파이프라인에서 대부분 단계를 자동화할 수 있다고 가정할 것이다.

모든 프로젝트를 구성원에게 머신러닝 교육을 시킬 수 있는 기회로 삼아라.

머신러닝 프로젝트를 진행하며 배운 점을 전사 회의, 타운홀, 가벼운 티타임 등 선호하는 방식으로 조직 전체에 꼭 공유해야 한다. ML옵스 프로젝트 진행 시, 조직의 효과적인 협업을 위해 추상적인 주제부터 세부 디테일까지 포괄해 공유한다.

ML옵스 프로젝트에선 조직 내 다양한 전문가의 참여가 지속적으로 요구된다. 고로, 직원에게 ML옵스 프로세스를 교육하는 과정을 마지막 단계로 미뤄선 안 된다.

공동 목표와 평가 지표를 명확히 하라.

조직의 모든 숨겨진 가정들을 알아냈다면, 다음은 프로젝트 목표와 성과 측정용 개별 평가 지표를 문서로 만들어 가상 팀과 공유한다.

가정
해당 프로젝트는 왜 존재하는가? 이 프로젝트가 머신러닝 프로젝트로 시작된 계기는 무엇인가? 사업적 기대는 무엇인가?
현재 어떤 방식으로 성과를 측정하는가? 어떤 기준을 대비해 벤치마킹하고 있는가? 현행 프로세스를 개선하는 게 목표인가? 혹은 수작업 과정을 보완하거나 자동화하는 일이 목표인가?
배포한 모델은 어떤 형태로 사용되는가? HTTP 엔드포인트(endpoint)를 이용해 공개적으로 모델에 접근할 수 있는가? 또는 접근이 내부 서비스로만 제한되어 있는가?
데이터는 누가 소유하는가? 데이터는 어디에 저장되고 어떻게 접근하는가? 데이터는 얼마나 자주 업데이트 되는가?

성공 기준

개별 조직과 프로젝트에 적합한 맞춤형 평가 지표를 설정한다.

기존 프로세스를 개선하거나, 수작업 과정을 보완하거나 완전 자동화하는 등 각자의 집중 방향이 다를 수 있다.

담당자는 단순히 프로젝트의 산출물이 아니라 추적이 필요한 성과와 달성해야 하는 수치를 방향성에 맞게 정의하고 나열해야 한다.

성공 기준을 위한 평가 지표 정의 시, 아래 항목들을 고려하라.

- 모델의 프로덕션 배포 가능 여부를 결정하는 핵심 평가 지표는 무엇인가?
- 모델은 프로덕션에 어떻게 배포될 것인가?
- 모델은 얼마나 자주 업데이트할 것인가?
- 모델 평가 지표 (예: 정확도, 정밀도, 재현율 등)
- 모델 유지 보수 그리고 업데이트 비용 (파이프라인 관련 비용 포함)
- 윤리적 고려 사항

최소 기능 제품 (Minimum Viable Product)이란? (*MVP: 완전 제품 출시 전, 최소 실행 가능한 형태로 출시해 고객들의 반응을 살펴보기 위한 제품. 최소 실행 가능 제품. —편집자)

모델 테스트를 위한 최소한의 요건을 정의한다. 이로써 프로젝트를 작은 단위로 세분화할 수 있고 더 빠르게 정량 결과물을 도출할 수 있다.

- 샌드 박스 환경 내에서 모델 정확도 테스트 가능
- 파이프라인을 수동으로 학습시키고 구축, 배포한다.

ML옵스 평가 지표
ML옵스 프로젝트에서 고려할 만한 추가 평가 지표다.

- 모델 업데이트 주기 달성 여부 (그리고 낙후 모델 식별 가능 여부)
- 신규 모델 재학습, 프로덕션 배포, 출시 소요 기간
- 온라인 예측을 위한 모델 엔드포인트 지표 (예: 밀리 세컨드- 1,000분의 1초 단위 응답 시간)
- 호출 횟수 / 모델 엔드포인트 호출 실패율
- 종합 팀의 협업 (예: 데이터 사이언티스트, 엔지니어, IT 운영, 법무, 사업 등)
- 주요 이해관계자의 프로젝트 정기 회의 참석 빈도 (예: 6주에 1회 등)
- IT 부서의 수작업 없이 머신러닝 팀 인프라 확장 가능성

실제 사례 – 두 기업 이야기

아래의 예시는 가상의 이야기이긴 하지만, 지난 5년간 반복적으로 관찰된 실제 패턴을 반영했다.

기업 1

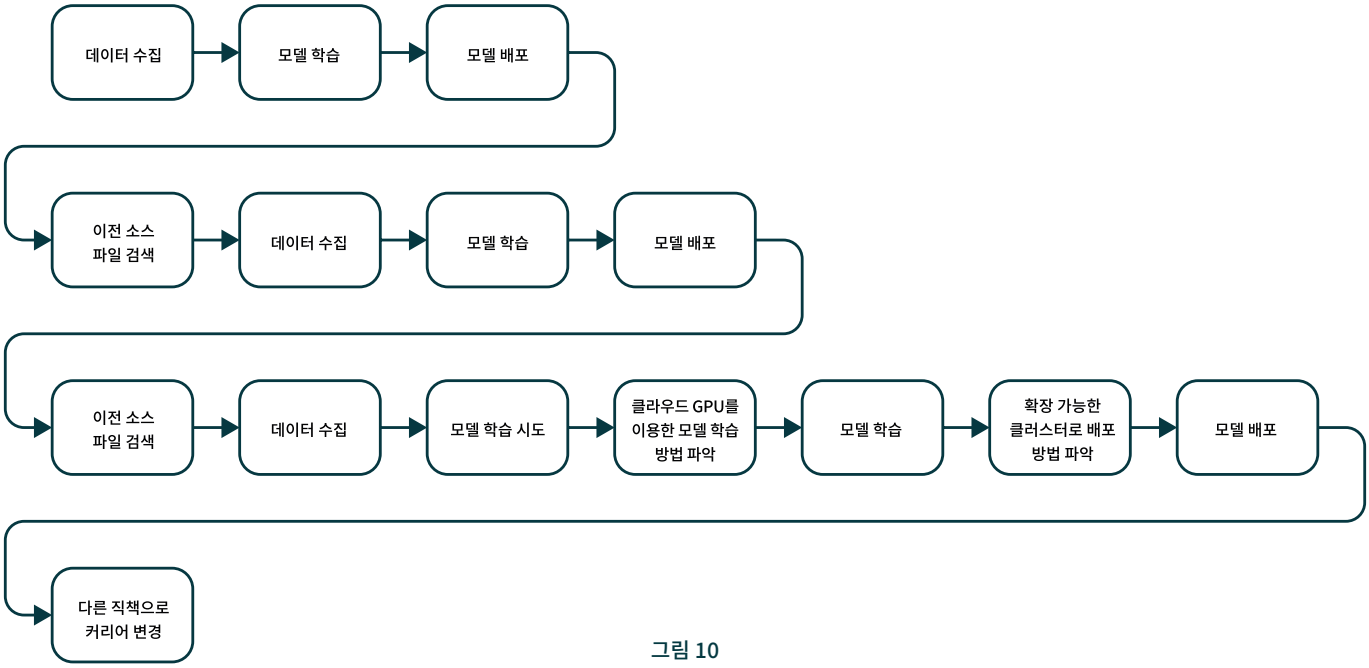


그림 10

기업 1은 실무에 바로 뛰어들어 데이터 수집과 분석 업무에 착수한다. 기업 1의 수석 데이터 사이언티스트는 본인의 노트북 PC에서 Jupyter notebook으로 모델을 개발한다. 해당 모델은 다른 팀에 있는 데브옵스 엔지니어에 의해 바로 프로덕션으로 배포된다. 아이디어 단계에서 프로덕션 배포까지 소요 시간이 매우 짧다.

그러나 시간이 지나면서 문제가 드러난다. 사용자 피드백에 의하면 모델의 예측 성능이 점점 더 나빠진다. 어쩌면 기초 데이터가 변경돼 모델을 재학습시켜야 할

수도 있다. 그러나 최초 작업 내용이 제대로 문서화되지 않았고, 한 대의 컴퓨터에서 상당 부분을 작업해 모델 학습에 필요한 업무를 그대로 재현하기 어렵지만, 그럼에도 어떻게든 모델 재학습이 진행 된다.

이후, 유사한 이슈들이 여러번 발생하며 매번 다양한 우여곡절 끝에 이슈를 해결한다. 수석 데이터 사이언티스트는 이미 퇴사했고, 그의 컴퓨터에서 작업한 내용은 대부분 유실됐다. 모델 학습을 로컬 머신에서 완료하는 건 너무 오래걸려서, 데브옵스 담당자를 투입해 클라우드 GPU 머신 상에서 어떻게 모델 학습을 진행할지 파악한다.

결과적으로 회사는 모델 학습 및 배포 과정의 자동화를 위해 각종 ML옵스 도구/솔루션(tooling) 도입을 결정하고, 지금까지 구현된 모든 것을 처음부터 다시 구축한다.

기업 2

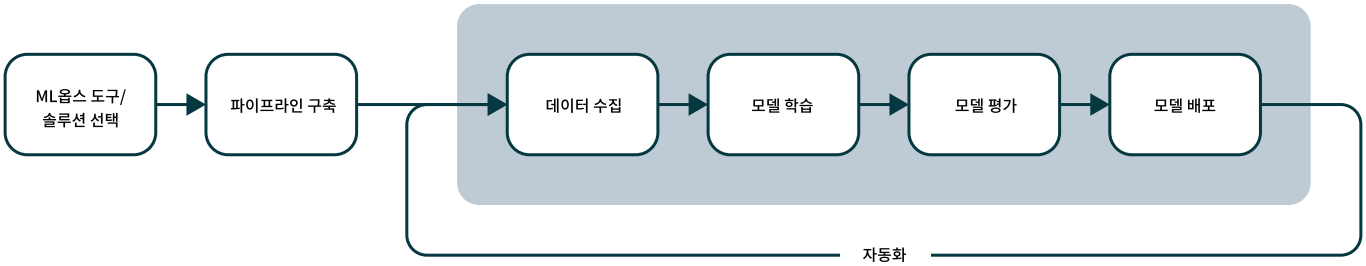


그림 11

기업 2는 머신러닝 모델이 향후 핵심 자산이 될 거라는 걸 인지하고, 최초 개념 증명 (Proof-of-concept) 단계 전 머신러닝 전용 도구/솔루션을 도입하기로 결정한다. 담당 팀은 1개월을 할애해 사용할 도구/솔루션들을 미리 선별하고, 데이터

수집 같은 수작업을 스크립트로 대체하고, 데이터 수집부터 모델 배포까지의 전 과정 (end-to-end) 파이프라인을 구축한다.

기업 2는 최초 출시까지 시간이 더 걸렸지만, 시간이 지날수록 그만큼 이점이 기하급수적으로 빠르게 늘어난다. 모든 업무가 자동으로 재현 가능해 데이터 사이언티스트는 급한 불을 끄는 이슈 대응보다 추가 개발 업무에 더 집중할 수 있다.

시사점

두 기업의 이야기를 통해 머신러닝 워크플로우를 너무 늦게 검토해선 안 된다는 걸 알 수 있다. 단순히 “우리는 아직 그 단계에 미치지 못했다”라고 생각하는 건 착오다. ML옵스 도구들과 워크플로우를 고려하기에 가장 좋은 시점은 데이터 사이언티스트를 최초로 고용했을 때이며, 결코 첫 번째 모델이 프로덕션에 배포된 시점이나 이후 모델 운영에 장애가 여러 차례 발생했을 시점이 아니다.

머신러닝 워크플로우 구축엔 기업의 전략적 결의와 투자가 요구된다. 그런데도 지금 같은 환경에서, 경영진은 선견지명을 갖고 데이터 사이언티스트를 지원하는 일 외에 선택의 여지가 없을 것이다. 시장에서 머신러닝 전문 인력 수요가 많고 비용이 높은 만큼, 고용한 자원을 낭비하는 일은 옳지 못하기 때문이다.

ML옵스 툴체인

이번 장에선 ML옵스 툴체인(toolchain)에 대해 다뤄볼 것이다. 각자 ML옵스 분야 중 어떤 곳에서 ML옵스 도구가 필요한지 파악하고, 어떤식으로 해당 도구를 활용할 수 있도록 돕는 것이 목표다. 본인에게 맞는 ML옵스 툴체인을 구축하는 방법은 다양한데, 여러 ML옵스 영역을 해결할 수 있는 플랫폼 한 가지를 구매하거나, 여러 전문 툴을 조합해 사용하거나, 또는 오픈소스 툴을 활용해 직접 구축할 수 있다.

본 자료에선 관련 툴을 직접적으로 추천하지는 않을 것이다. 오히려 올바른 선택을 하도록 안내하고, 더 중요하게는, 선택한 툴을 최대한 잘 활용하도록 도울 것이다.

데이터 플랫폼

이 장에서는 머신러닝 데이터 플랫폼이 제공하는 편익에 대해 살펴볼 것이다.

반복적인 데이터 확보와 재학습 사이클을 위한 데이터 플랫폼

데이터는 많은 머신러닝 개발 프로젝트에서 병목(bottleneck)으로 꼽힌다. 시장조사기관 코그닐리티카(Cognylitica)의 자료에 따르면, 머신러닝 개발 과정에서 데이터 작업이 차지하는 비중은 약 80%에 이른다. 데이터 작업은 이처럼 절대적인 시간이 많이 필요한 부분이라 이를 효율화하는 것이 머신러닝 프로젝트 전체의 효율화에 큰 영향을 줄 수 있으며, 학습용 데이터의 품질은 출시된 서비스의 성능에도 직접적인 영향을 미친다. 머신러닝 업계에서 가장 유명한 문장 중 하나인 “쓰레기를 넣으면, 쓰레기가 나온다(Garbage In, Garbage Out)” 는 데이터의 품질이 결국 머신러닝의 성능의 핵심임을 잘 보여준다.

더군다나 머신러닝 개발을 위한 데이터 확보와 가공, 관리는 머신러닝 개발 프로젝트 기간 전체와 서비스 운영 기간 전체에 걸쳐 반복된다. 맥킨지(McKinsey)의 분석에 따르면, 34%의 머신러닝 프로젝트 사례에서 데이터 확보와 재학습이 월(Month) 단위로 필요하다고 응답하였으며, 이 중 23%는 이런 반복적인 과정이 적어도 일주일에 한번씩 필요하다고 응답하였다. 머신러닝 생애 주기 안에서 이 반복적인 사이클은 피할수 없는 것이기 때문에, 이 사이클을 관리하는 머신러닝 데이터 플랫폼을 도입함으로 제품 개발까지의 비용을 줄이는 것을 고려해야 한다.

머신러닝 개발 과정 전체에서 '데이터 사이클'이 구축되어야 하는 대표적인 상황은 다음과 같다.

1. 모델이 제대로 동작하지 않을 때(정확도accuracy 가 요구 수준에 미치지 못할 때), 학습용 데이터를 더 많이 수집하고 라벨링해서 학습용 데이터를 더 확보해야 할 필요가 있을지 모른다. 또는 좀 더 섬세하게 분류된 어노테이션을 기존 데이터 셋에 추가하여 작업할 수도 있다. 모델이 특정 케이스에서 성능이 떨어지는 것이 확인되었다면, 데이터셋이 불균형하거나 편형된 것이 그 원인일 수 있고, 이 때는 상대적으로 사례수가 부족한 엣지 케이스(edge-case)를 추가적으로 수집해야 할 수도 있다.

2. 데이터셋에서 변화가 생기면, ML 모델을 변경하고 싶을 수 있다. 더 복잡하고 고차원적인 표현(representations)을 학습할 수 있는 고성능의 모델이 필요해 질 수 있다. 또한 특정한 데이터 부분집합 케이스에 특화된 별도의 모델을 학습시키는 것을 고려할 수도 있고, 하나의 백본(backbone) 위에 여러개의 헤더 네트워크(header network)를 얹은 모델 아키텍처를 활용해서 모델이 학습할 수 있는 클래스의 집합을 확장하고자 할 수도 있다.

3. 모델이 배포된 이후에라도, 데이터 드리프트(머신러닝 컨텍스트에서 모델 성능 저하를 초래하는 입력 데이터의 변경 내용을 말한다)에 노출될 가능성이 있고, 이로 인한 성능저하가 발생할 수도 있다. 이 경우에는, 데이터셋과 모델이 지속적으로 업데이트 되어야 하고, 성능을 유지하기 위한 재학습이 필요하다.

이 단계에서는 파편화된 임시 변통의 솔루션들을 조합하는 것이 궁극적인 해답이 되지 않는다. 팀 내 협업을 원활하게 도와주면서, 빠르게 데이터 셋을 구축하고, 이것의 품질을 극도로 끌어올릴 수 있는 방법을 찾아야 프로덕션 수준의 머신러닝 팀이라고 할 수 있다.

동적인 데이터 관리

데이터 플랫폼은 동적인 데이터의 관리를 돕는다. 시간이 지나도 데이터셋에 어떤 변화도 발생하지 않을거라 예상된다면, 정적인 훈련 방식(static training)을 하는 것이 훨씬 더 비용 효과적일 것이다. 하지만 현 시대의 개발 환경을 고려할 때, 데이터가 정적일 확률은 거의 불가능에 가깝다.

변화 추적 Change Tracking

지속적인 데이터의 변화를 쉽게 추적하고 변경 이력을 관리해야 한다. 원본 데이터(raw data)만 추가된 것인지, 어노테이션을 업데이트한 것인지 등 변경 이력을 상세하게 추적하고, 관련자에게 일을 할당해야 한다. 데이터 라벨링 작업은 인하우스(in-house) 조직 혹은 외부의 용역 업체, 또는 프로덕트 매니저나 프로젝트 매니저가 엔지니어링팀과 연구팀을 위해 업데이트 작업을 책임져야 하는 경우가 대부분이다. 이 과정에서 모든 이해 관계자가 동일한 선상에 있을 수 있게 데이터 전달 과정이 최대한 투명하고 막힘이 없어야 한다.

버전 관리 Versioning

데이터셋이 진화함에 따라, 데이터는 동적으로 변하고, 모델 학습 실험이 지속적으로 반복된다. 이 과정에서 예전의 데이터나 어노테이션 버전으로 회귀를 해야 하는 경우도 심심치 않게 발생한다. 특정한 사례에서 어떤 데이터가 더 적합한지를 알아보는 A/B 테스트를 진행해야 하는 경우도 흔히 발생한다. 데이터 플랫폼은 데이터셋의 변화를 추적하는 기능을 제공한다. 이 경우 데이터 플랫폼은 코드 전체와 데이터의 출처를 제공함으로서 모든 머신러닝 데이터셋의 진화과정을 추적할 수 있게 돕는다. 이 덕분에 재현성이 보장되며, 왔다갔다 여러 번의 실험을 수행할 수 있다. 머신러닝 데이터를 구축하고 관리하는 것도 장기적으로는 데이터셋의 규모와 무관하게 지금 Git과 같은 플랫폼을 활용하여 코드를 관리하고 협업하는 것과 유사해 질 것이다. 데이터셋의 브랜치를 만들고 변화나 업데이트를 커밋(commit)하며, 두 개의 데이터셋을 머지(merge)하고, 충돌난 부분을 해결하는 것이다.

유연한 조작 Flexible Manipulation

데이터 플랫폼은 규모있는 데이터셋의 역동적인 변화를 관리할 수 있는 다양한 기능을 제공한다. 파일명, 데이터의 생성/수정 날짜, 파일 사이즈 등등에 기반한 필터링과 분류 등이 가능하다. 머신러닝 데이터만의 독특한 특성도 있는데, 예를 들어 어노테이션 유형이라던가 라벨이라던가, 데이터 작업의 어느 단계에 현재 데이터가 위치하는지(어떤 모델 학습에 사용되었는지, 데이터의 어노테이션 작업 완료 여부, 어노테이션 작업의 검수 작업 완료 여부) 등이 분류 기준이 되기도 한다. 이런 머신러닝 데이터의 고유 특성 때문에 기존 파일 관리 도구만으로는 데이터를 유연하게 조작하는 것이 어렵다.

인공지능 활용한 데이터 라벨링

머신러닝 개발 프로젝트에서 데이터 라벨링 작업 역시 여러번에 걸쳐 수행된다. 이 때문에 데이터 라벨링 작업의 자동화는 불가피하다. 라벨링 작업의 자동화를 실현하지 못하고, 매뉴얼 라벨링에만 영원히 의존해야 한다면 조직의 스케일업은 불가능하기 때문이다.

AI를 활용한 라벨링 자동화 Auto Labeling with AI

가장 기본적이고 단순한 방법은 이미 학습된 모델을 활용해 라벨링 작업을 자동화하는 것이다. 거대 테크기업이나 스타트업이 이미 개발하여 제공하고 있는 안면 인식, OCR, 일반 사물 인식, OCR과 관련 AI API 서비스가 많이 있다. 데이터 플랫폼은 이런 API를 불러와서 후에 사람이 수정할 수 있는 1차 라벨링 작업을 수행할 수 있다. 인터랙티브 AI(Interactive AI) 모델도 있는데, 바운딩 박스 어노테이션을 폴리곤 세그멘테이션으로 바꾸거나, 사람이 몇 번의 클릭만으로 매끈한 폴리곤 세그멘테이션 어노테이션 작업을 수행할 수 있게 돕기도 한다. 상대적으로 이를 적용하기 쉽다는 점은 장점이지만, 성능이 제한적일 수 있고, 실제 어노테이션 비용을 줄일 수 있을지에 대한 의문은 여전히 남는다.

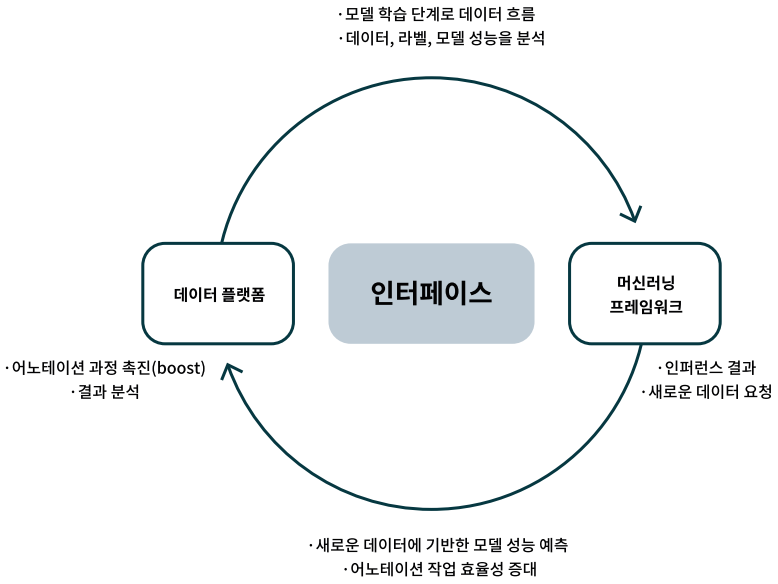
AI를 활용해서 데이터 라벨링 작업을 효율화하는 사례

분류	장점	단점
AI API 서비스	• 바로 사용 가능	• 너무 지엽적임(특정 사례에만 적용 가능) • 너무 일반적임(사용자에 맞춤의 데이터셋에 훈련된 것이 아님) • 재 학습이 불가능
인터랙티브 AI	• 활용하기 쉬움 • 클래스에 구애 받지 않음 (어떤 물체에나 사용 가능)	• 성능과 효과성 측면에서의 의문이 있음(매뉴얼 라벨링보다도 사람의 노동이 더 필요할 수 있음)
커스텀 AI	• 확보한 데이터셋에 맞춰져 있음	• 비용 효과적인지 신중하게 검토해야 함 • 머신러닝이 규모의 경제를 달성하기 위해서는 상당한 기간이 필요함

자사의 AI 모델을 불러와 라벨링 자동화에 활용하는 경우는 콜드 스타트 문제가 있다. 각각의 모델은 충분히 라벨링된 데이터가 모였을때에만 성능을 담보할 수 있기 때문이다. 이 문제를 해결하기 위해, 데이터 플랫폼은 다음과 같은 진보된 머신러닝 기술들을 적극적으로 활용하고 있다. 예를 들자면 다음과 같다.

- **퓨샷 러닝(Few-shot learning)**
  샘플 데이터 몇 개 만으로도 머신러닝 모델을 만들 수 있게 돕는다.
- **반 지도 학습(Semi-supervised learning)**
  적은 수의 어노테이션 만으로도 머신러닝 모델을 만들 수 있게 돕는다.
- **약한 지도 학습(Weakly supervised learning)**
  거친 어노테이션 작업을 진행했더라도 머신러닝 모델을 만들 수 있게 돕는다.
- **전이 학습(Transfer learning)**
  다르지만 유사한 과업에서 얻은 지식을 재사용하여 머신러닝을 개발할 수 있게 돕는다.
- **액티브 러닝(Active learning)**
  모델 학습에 가장 의미있는 데이터를 먼저 어노테이션 작업할 수 있게 돕는다.
- **데이터 증강(Data Augmentation)**
  새로운 데이터셋을 모으지 않고 기존의 데이터셋에서 다양성을 높일 수 있다.
- **데이터 검색 Data Retrieval**
  큰 데이터베이스에서 데이터를 검색하여 가져오는 것이다. 예를 들어, 이미지 유사성(image similarity metrics)에 근거해 다른 데이터 셋에서 이미지들을 가져올 수 있다.
- **데이터 합성 (Synthetic Data)**
  보유하고 있는 데이터와 유사한 가짜 이미지를 합성하여 만든다. (GAN과 같은 생성 모델로 만들 수 있다)

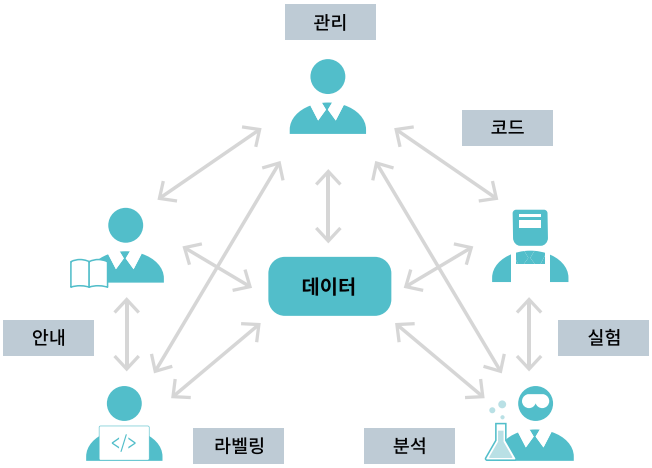
ML구성 요소와 연동



오늘날 머신러닝 산업에 속한 대부분의 기업은 데이터 라벨링, 모델 학습, 모델 배포, 시각화 등등 특정 전문 영역에 집중하고 있다. 훌륭한 데이터 플랫폼은 데이터가 원본 데이터를 라벨링하는 팀에서 부터 모델을 학습시키고 배포시키는 팀에 이르기까지 원활하게 흐를 수 있게 돕는다. 머신러닝 생애 주기에는 다음과 같은 구성요소가 있는데, 머신러닝 데이터 플랫폼은 이들 구성요소들을 최대한 많이 연동하면 좋다

- 클라우드 저장소
- 데이터 수집 파이프라인
- 데이터 라벨링 서비스
- 머신러닝 프레임워크
- 모델 학습 환경 & 서비스
- 모니터링
- CLI/SDK

ML 프로젝트 내에서의 협업



기업 내에서의 협업

큰 기업의 경우 데이터 라벨링팀, 머신러닝 팀이 모두 사내에 있을 수 있다. 라벨링팀은 학습용 데이터셋을 생성하여 데이터 허브에 업데이트하며, 엔지니어팀은 모델 개발을 위해 허브에 접속한다. 이렇게 여러 이해당사자가 허브에 접속하게 될 경우 권한 관리가 필요하다. 또한 작업에 대한 필요가 발생할 경우 관련자에게 할당하고, 이슈가 발생한 경우는 이슈에 대해 소통해야 한다.

기업 간 협업

기업 내에 모든 팀이 있는 것보다도 다른 기업과 협업하는 것이 더 흔한 경우일 수 있다. AI 기술 기업이 데이터 라벨링을 전문으로 수행하는 기업과 협업하는 것이다. 이 경우, AI 기술 기업이 프로젝트를 생성하여 외부의 조직을 초대하는 것이 필요하고, 데이터 라벨링 기업도 라벨링 작업을 효율적으로 수행할 수 있는 데이터 작업 플랫폼이 필요하다.

오픈소스 협업

CAPTCHA와 같이 데이터 작업이 분산되어 오픈소스로 기여하는 방식이다. 지금은 이렇게 작업하는 경우가 많지는 않지만, 미래에는 이런 사례가 더욱 많아질 것이다. 10년 전만 하더라도 소프트웨어는 한 기업의 핵심적인 지적 자산이었다. 마치 오늘날 많은 기술기업에서 머신러닝이 핵심적인 지적 자산인 것과 같다. 시간이 지나면서 소프트웨어 기업들이 많은 소프트웨어 라이브러리와 프로젝트를 오픈소스화 한 것과 같이, 5년 10년 사이에 머신러닝 업계에서도 동일한 현상이 발생할 수도 있다. 누구나 데이터셋을 복제하고, 브랜치를 만들고, 머지(merge)할 수 있고, 데이터셋에 대한 분석과 모델을 공유하고, 데이터셋을 함께 계속 발전시켜 나가고 나아가 데이터셋을 활용한 모델 발전에도 데이터 플랫폼을 통해 공동 기여할 수 있을 것이다.

주요 시사점

1. 머신러닝 데이터 플랫폼은 데이터 확보와 재학습의 사이클을 구축하는 것을 돕는다.
2. 데이터 플랫폼은 동적인 데이터 관리를 가능하게 하고, 데이터 라벨링 작업에 인공지능을 활용하며, 데이터가 머신러닝 개발을 위한 다양한 하위 작업과 데이터셋을 연동하며, 머신러닝 프로젝트 내에서의 협업을 돕는다.

모델과 데이터 탐색

탐색은 머신러닝 프로젝트에서 가장 먼저 하게되는 구체적이고 실무적인 단계다. 따라서 이는 모두에게 가장 익숙한 단계이면서 교육 프로그램, 경진대회, 일반 취미 프로젝트에서 가장 큰 비중을 차지한다. 많은 데이터 사이언티스트에게는 탐색 단계가 머신러닝 프로젝트의 전부로 느껴질 수 있다. 그렇기 때문에 프로젝트가 제품 상용화, 배포 단계까지 진행되며, 탐색 단계에서와는 다른 툴과 사고방식을 요구할 때는 혼란이 자주 발생한다

데이터 탐색

데이터가 없으면 모델도 존재할 수 없기에 모델 탐색 전, 데이터 탐색이 반드시 선행돼야 한다.

데이터 탐색의 목적은 데이터를 이해하는 일이다. 컴퓨터는 우리에게 익숙하지 않은 테이블 형태로 데이터를 보유한다. 수백만 행을 차지하는 미가공 상태의 테이블은 결코 인간의 이해에 최적화된 인터페이스가 아니다. 반면 인간은 세상을 바라볼 때 모양, 차원, 색깔에 의존하는 것에 익숙하다. 특정 변수의 특징과 다른 변수와의 관계를 이해하려면, 데이터를 시각적으로 분석하도록 돕는 툴이 필요하다. 모델 탐색 단계에서 데이터에 대한 이해는 데이터를 유용한 상위 레벨의 특성으로 변환하고 처리하는 일이 쉽도록 돕는다. 데이터 탐색 단계는 기술적 관점, 데브옵스 관점에서 가장 낮은 수준의 사항을 요구한다. 반복적으로 빠르게 데이터 탐색을 할 수 있고, 데이터 시각화 등을 통한 시각적 피드백을 받는 것이 데이터 탐색 업무의 핵심 요소다. Notebook은 코드, 시각화 기능, 빠르고 반복적인 개발 사이클에 최적화된 환경을 모두 지원하기 때문에 다른 툴보다 이 단계에 제일 적합하다. 이 단계에서 많은 Notebook은 일회용으로 사용되고 폐기된다. Notebook으로 얻은 통찰과 발견 사항은 실질적으로 가치가 있고, 이를 가능하게 한 환경 자체를 보존하는 일엔 큰 의미가 없다. 재현성, 라이브러리 종속성, 버전 관리의 가치가 추가로 생길 수 있지만 대부분 필수 요구 사항은 아니다.

모델 탐색

모델 탐색은 데이터 탐색과 중복될 수 있으나, 일반적으로 별도의 단계로 생각하는 게 좋다. 모델 탐색 단계에서 데이터 사이언티스트는 회귀, 결정 트리(decision tree), 랜덤 포레스트(random forest) 같은 당면한 문제에 적합한 다양한 모델의 실행 가능성을 두고 탐색한다. 모델 선택에선 데이터 사이언티스트가 직접 시행착오를 겪으며 모델의 초매개변수(hyperparameters)라고도 불리는 최적의 매개 변수를 찾아내는 일이 필요하다. AutoML 같이 최적의 모델과 매개 변수를 자동으로 찾을 수 있도록 지원하는 툴 또한 존재한다. 그러나 복잡한 프로젝트에서 이런 툴은 잘 동작하지 않고 딥러닝에 완벽한 적용이 어려운 경우가 많다.

기술적 관점, 데브옵스 관점으로 보면 모델 탐색엔 데이터 탐색 단계보다 더 높은 요구 사항이 존재한다. 단일 notebook은 자주 사용되지만, 서드 파티(3rd party) 라이브러리 종속성, 실험 재현성, 확장 가능한 컴퓨팅 인프라 그리고 버전 관리가 더 유용한 가치를 갖는다. 데이터 탐색은 보통 로컬 컴퓨터에서도 가능하지만, 모델 탐색 단계는 훨씬 높은 컴퓨팅 자원이 요구된다. 한 가지 매개 변수(parameter) 값을 가진 모델 하나의 실행 가능성을 테스트하는 것만으로도 몇 시간, 때론 며칠까지 소요된다. 따라서 효율적인 업무를 위해 자동으로 확장 가능한 클라우드 환경은 필수다. 모델 탐색은 시간과 돈이라는 비용이 들어가는 단계로, 각 실험의 버전 관리와 재현성이 그 무엇보다 중요하다.

탐색과 ML옵스

기술적 관점, 데브옵스 관점으로 보면 모델 탐색엔 데이터 탐색 단계보다 더 높은 요구 사항이 존재한다. 단일 notebook은 자주 사용되지만, 서드 파티(3rd party) 라이브러리 종속성, 실험 재현성, 확장 가능한 컴퓨팅 인프라 그리고 버전 관리가 더 유용한 가치를 갖는다. 데이터 탐색은 보통 로컬 컴퓨터에서도 가능하지만, 모델 탐색 단계는 훨씬 높은 컴퓨팅 자원이 요구된다. 한 가지 매개 변수

(parameter) 값을 가진 모델 하나의 실행 가능성을 테스트하는 것만으로도 몇 시간, 때론 며칠까지 소요된다. 따라서 효율적인 업무를 위해 자동으로 확장 가능한 클라우드 환경은 필수다. 모델 탐색은 시간과 돈이라는 비용이 들어가는 단계로, 각 실험의 버전 관리와 재현성이 그 무엇보다 중요하다.

주요 시사점

1. 데이터 탐색은 데이터를 시각화해 이해하기 쉽게 만든 가벼운 오프라인 작업이다.
2. 모델 탐색은 컴퓨팅 연산 능력과 버전 관리 차원에서 더 높은 기준을 요구한다.
3. 탐색 단계와 ML옵스를 분리해 취급하는 일은 데이터 사이언티스트와 엔지니어의 사이가 멀어지도록 만드는 지름길이다.

평가 지표와 모델 최적화

지난 섹션에선 데이터와 연관 모델을 시각화하고 탐색하는 방법을 알아봤다. 이번엔 다양한 모델의 가치를 평가하고, 프로덕션에 가장 적합한 모델을 찾는 방법을 살펴보자. 특히 모델의 초매개변수(hyperparameter)를 선택하는 방법을 깊게 다루겠다.

평가 지표는 모델의 성공 여부를 정의하는 수치이다. 예를 들어 신용카드 사기 거래를 몇 건이나 식별해야 성공적이라고 평가할 것인가? 또는 특정 광고에 반응한 조회자 수는 몇 명인가?, 아니면 6개월 후 철강 가격을 얼마나 잘 예측할 수 있을까? 등이 있다.

하지만 평가 지표를 정의하는 일은 전체 과정 중 절반에 불과하다. 모델을 프로덕션에 배포하기 전에, 데이터 사이언티스트는 주요 평가 지표에서 충분한 성능

을 낼 수 있는 모델을 찾아야 한다. 모델의 성능은 학습률(learning rate), 계층당 노드 개수(nodes per layer) 등 초매개변수(hyperparameter)에 큰 영향을 받는다.

최적화의 의미

소프트웨어 공학(알고리즘) 분야에서 함수 최적화란 함수를 더 작고 빠르게 재작성하는 일을 의미하고, 여기서 다루는 정의와 전혀 달라 가끔 혼란을 주기도 한다.

데이터 사이언스(수학) 분야에서 함수 최적화란, 입력값을 변경해 출력값을 최소화(또는 최대화)하는 걸 의미한다. 손실 함수(loss function)란 모델의 오차를 구하기 위한 수식으로, 손실 함수를 최적화한다는 건 해당 모델의 오차를 최소화한다는 것과 같은 개념이다.

손실 함수를 최적화하는 행위를 주로 학습(training)이라고 부른다. 학습 과정에서 모델 학습용 데이터를 반복 입력해 작은 단위의 수정을 거쳐 오차를 점차 줄여나간다. 학습 과정에서 발생한 오차를 학습 손실(training loss)이라고 부른다.

학습/검증/테스트용 데이터 세트

프로덕션 환경에서 보유한 데이터는 주로 학습, 검증, 테스트 데이터 세트, 이렇게 세 종류로 분류한다.

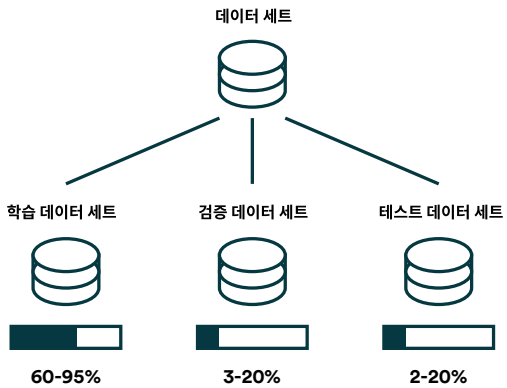


그림 12

이런 데이터 세트는 보통 전체 모집단을 일관되게 대변하려고 계층화된 표본으로 구성한다.

- **학습 데이터 세트** - 보유 데이터의 60~95%는 학습 데이터 세트로 활용되며, 모델의 학습 손실(loss) 연산에 사용된다. 학습 손실 최소화를 거쳐 최적의 랜덤 포레스트 알고리즘의 분할 기준 또는 딥러닝 인공 신경망의 가중치 값을 찾을 수 있다.
- **검증 데이터 세트** - 보유 데이터의 3~20%를 활용해 모델의 검증 지표를 정의하는 데 사용된다. 검증 지표 최적화를 통해 모델의 초매개변수(hyperparameter)를 찾을 수 있다.
- **테스트 데이터 세트** - 보유 데이터의 2~20%를 활용하고, 검증 지표를 별도 데이터 세트에 적용해 연산하는 데 사용된다. 모델 튜닝 이후 프로덕션 단계에서 모델이 실제로 어떻게 동작할지에 대한 객관적인 추정치를 제공한다. 검증 지표는 초매개변수(hyperparameter)를 체계적으로 선별하는 과정에서 핵심 도구다. 만약 초매개변수(hyperparameter)가 학습 손실 최소화를 거쳐 튜닝됐다면 학습 데이터에 과적합할 가능성이 크다. 따라서 별도 검증 데이터 세트를 통해 초매개변수를 선별하면 모델이 처음 접하는 데이터에 일반화하기 더 쉬울 것이다.

튜닝과 프로덕션을 위한 검증 지표

검증 지표는 말 그대로 모델의 프로덕션 성능을 검증하는 지표다. 이는 모델 실사용 시 성능을 반영하는 지표여야 하고, 일반적인 머신러닝 강좌 내용을 따라가거나 구체적인 비즈니스 사례에 특화되기도 한다. 이와 관련한 아래 예시를 살펴보자.

차량을 자전거, 자동차, 트럭 등으로 식별하는 이미지 분류 모델을 예로 들어보겠

다. 이때 정분류율(fraction of correct classifications), 즉 검증 정확도가 간단한 지표가 될 수 있다. 운전자 안내를 위해 차량 내부에 사용될 모델에겐 탑승자 안전을 위한 자전거 오분류율(fraction of misclassifications)이 중요한 지표가 될 수 있다. 반면 자동화 톨게이트에 사용될 모델에겐 트럭에 대한 정분류율이 다른 차량보다 훨씬 더 중요한 지표가 될 것이다. 모델이 보안 출입구를 여닫는 데 사용된다면 추론 시간(inference time)이 중요한 지표가 될 수 있다.

평가 지표 최적화와 의사결정

프로덕션용 머신러닝 파이프라인에선 각종 모델이 알맞은 때에 배포될 수 있도록 모델 튜닝이라고 불리는 초매개변수 최적화 과정이 효율적으로 진행돼야 한다. 그러나 안타깝게도 초매개변수를 선별할 때마다 모델을 매번 새로 학습해야 해서, 모델 튜닝 비용이 커질 수밖에 없었다. 또 검증 지표는 거의 모든 경우 미분값(gradient) 관련 정보를 구할 수 없어서, 학습 손실 최소화에 사용되는 일반적인 최적화 기법을 검증 지표 최적화에 사용하기 어렵다.

베이지안 최적화(Bayesian optimization)는 모델 튜닝에 많이 사용되는 도구다. 미분값 정보를 요구하지 않기에 지저분한 지표들을 대상으로 최적화가 가능하고, 예를 들어 Adam과 Adagrad, RMSProp 중 선택하는 것 같은 범주형 매개변수(categorical parameters)에 적용 가능하다. 이 방식은 기본적으로 초매개변수 제안, 검증 지표 연산 후 최적화 연산을 진행하며, 반복적으로 초매개변수를 제안하고 단계적으로 최적화 하는 과정을 반복한다.

대부분의 프로덕션 환경에서는 여러 평가 지표로 성공 여부를 정의한다. 앞서 살핀 컴퓨터 비전 관련 예시에선 높은 정확도 (더 방대하고 복잡한 모델), 낮은 추론 시간 (더 작고 간단한 모델) 같은 지표 사이에서 균형을 찾아야 했다. 금융 거래를 위한 모델은 고수익률, 리스크 감소와 유동성 증대의 지표 사이에서 (각각 세

부 정의와 무관하게) 적절한 균형을 찾아야 할 것이다. 사기 탐지 모델의 경우 판매량 최대화 (거래 승인 건수 증대)와 손실 최소화 (거래 거절 건수 증대)가 동시에 중요할 것이다.

다중 지표 최적화(multimetric optimization)는 서로 상충하는 평가 지표 사이의 균형점을 탐색하는 방법이다. 이를 통해 모델 개발자들은 핵심 평가 지표 사이의 균형을 이해할 수 있고, 핵심 수요에 부합하는 초매개변수를 고를 수 있다. 추가로, 모델의 특정 성능 지표가 최저 성능 임곗값을 넘어야 한다면 관련 된 지표 임곗값 또는 지표 제한 조건을 적용할 수 있다. 예를들어 컴퓨터 비전 모델에선 정확도를 최대화하되 모델의 추론 시간을 100ms 이하로 제한을 두는 방법이 있다.

하나 혹은 다수의 평가 지표로 모델 최적화를 완료한 뒤에는, 테스트 데이터 세트를 활용하여 프로덕션용으로 검토 중인 초매개변수 지표를 재평가해야 한다. 이를 통해서 상용화 하기 이전에 모델의 일반화 가능성을 확인할 수 있다.

주요 시사점

1. 보유하고 있는 데이터를 학습, 검증, 테스트 데이터 세트로 나눈다.
2. 학습 단계 뿐 아니라 프로덕션의 성공을 나타낼 수 있는 평가 지표를 정의하고 탐구한다.
3. 가능한 최소한의 튜닝 비용을 들여 최상의 초매개변수를 찾아낸다.

프로덕션 준비 – 엔드 투 엔드 파이프라인

일반적으로 데이터 사이언티스트는 컴퓨터 한 대와 정적인 데이터 세트 하나를 갖고 문제 해결을 시작한다.

> 고객이 이탈하는가? 위성 사진에 핵 미사일 저장고가 포착되는가? 사용자가 다음으로 입력할 단어는 무엇인가?

이러한 문제를 해결하기 위해 데이터 사이언티스트는 수많은 커피를 마시면서 노트북(역자: Jupyter notebook 등)으로 데이터 정제, 피처 변환, 다수의 라이브러리 설치, 코드 작성, 다양한 모델 적용 시도 그리고 초매개변수(hyperparameter) 교체 등 일련의 작업을 진행한다. 결국 이 과정에서 문제를 해결할 방대한 노트북이 탄생하고, 이는 데이터 사이언티스트가 작업하는 컴퓨터에 보관한다.

위와 같이 머신러닝의 프로덕션 준비 과정에는 데이터 사이언티스트가 작업하고 있는 컴퓨터 한 대에 국한된 문제 해결 역량을 더 큰 접근과 확장이 가능한 시스템으로 다듬는 일을 뜻한다. 이런 시스템은 확장된 뒤에도 가치를 제공할 수 있어야 하고, 이후 수년간 업데이트하고 개선하고 관리할 수 있어야 한다.

수동 사이클

수동적으로 진행되는 워크플로우엔 실질적인 인프라가 존재하지 않아, 데이터 사이언티스트는 방대한 양의 파이썬 노트북 파일들을 머신러닝 엔지니어에게 전달한다.

"ML옵스: 머신러닝의 지속적 배포와 자동화 파이프라인"이라는 구글 기사에서 발췌

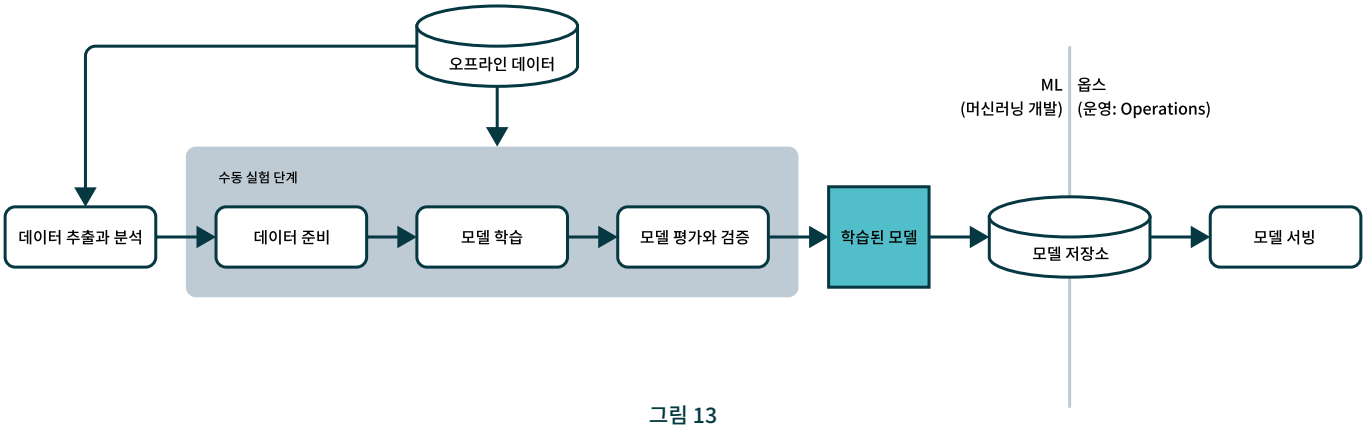


그림 13

담당 엔지니어는 코드 성능 향상뿐 아니라 프로덕션 환경과 연동을 위해 수동으로 코드를 정리하고 리팩토링한다. 이후 필요한 라이브러리, 환경 그리고 실시간 데이터 수집 방식 등을 파악하고, 배포용 엔드 포인트를 구축하고, 최종 모델을 프로덕션에 내보낸다. 이런 워크플로우에선 모델 자체가 제품이다.

일정 시간이 지나면 모델이 비정상적으로 작동한다는 제보가 팀에 들어온다. 기존의 문제 해결 역량이 떨어지는 걸 직감하지만, 확실하게 아는 사람은 아무도 없다. 데이터 사이언티스트는 신규 오프라인 데이터 배치를 수동으로 긁어온 뒤 시스템을 가동해본다. 이제 수동 사이클 전체를 처음부터 다시 시작하는 일만 남는다.

수동 머신러닝 파이프라인 특징:

- 모델 자체가 제품이다
- 수동형이거나 스크립트 기반의 프로세스
- 데이터 사이언티스트와 엔지니어 사이의 간극
- 느리게 반복되는 사이클
- 테스트 자동화와 성능 모니터링의 부재
- 버전 관리의 부재

자동 파이프라인

자동 파이프라인 기반의 워크플로우에선 모델을 구축하고 유지하는 대신 파이프라인을 구축하고 유지한다. 즉, 파이프라인 자체가 제품이다.

자동 파이프라인은 다양한 요소와 이들이 각각 어떻게 연결돼 핵심 요소인 모델을 생산하고 업데이트하는지에 대한 청사진으로 이루어진다.

자동 워크플로우에서 데이터 사이언티스트는 오프라인 데이터가 탑재된 컴퓨터에 방대한 양의 파이썬 노트북 파일들을 만드는 걸 지향하지 않는다. 최초 단계에서는 하나의 파이썬 노트북 파일로로 시작할 수 있지만, 결국 문제해결을 위한 전체 과정을 관리하기 좋은 규모의 요소들로 나눠 해결하는 방식을 지향한다.

"ML옵스: 머신러닝에서의 지속적 제공 및 자동화 파이프라인"이라는 구글 기사에서 발췌

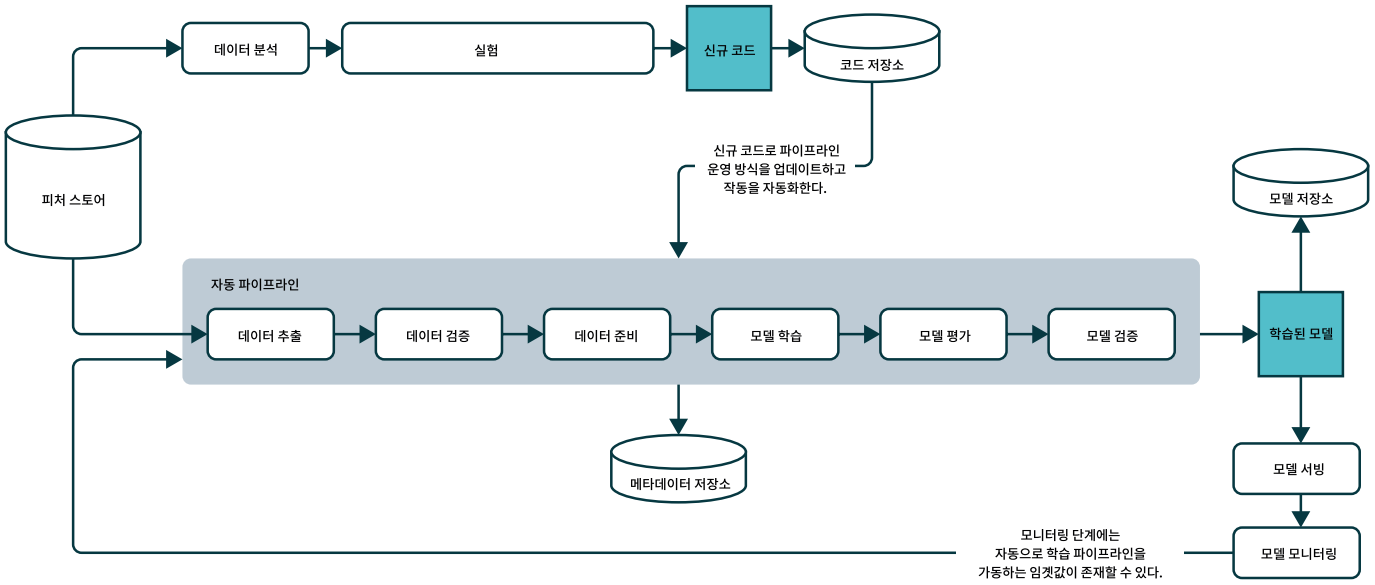


그림 14

다양한 요소들의 예:

- 데이터 검증
- 데이터 정제
- 학습
- 모델 평가
- 모델 검증
- 재학습용 트리거(trigger)

추가로, 파이프라인엔 아래 같은 정적인 요소가 있을 수 있다:

- 피처 스토어
- 배포용 엔드 포인트
- 메타 데이터 저장소
- 소스 코드 버전 관리

이 방식의 시스템은 파이프라인 내 각각의 단일 요소를 실행하고, 반복하고, 모니터링하는 일을 컴퓨터 내 로컬 노트북에서 실행하는 것과 같은 수준으로, 쉽고 빠르게 반복할 수 있도록 돕는다. 또 입력값과 출력값, 라이브러리 종속성과 모니터링할 평가 지표 같은 요구 사항을 정의하도록 돕는다.

이렇게 문제 해결 방식을 재현 가능하고 미리 정의해 실행 가능한 요소로 나누는 이 역량은 조직이 통일된 프로세스를 충실히 지키도록 돕는다. 통일된 프로세스는 데이터 사이언티스트와 엔지니어의 명확한 의사소통을 돕고, 결과적으로 이는 머신러닝 분야에서의 일종의 지속적 통합(CI)이라 할 수 있는 모델 업데이트 자동화 구조로 이어진다.

자동 머신러닝 파이프라인의 특징:

- 파이프라인 자체가 제품이다
- 완전히 자동화된 프로세스
- 데이터 사이언티스트와 엔지니어의 협력
- 빠른 반복 사이클
- 테스트, 성능 모니터링 자동화
- 버전 관리 용이성

주요 시사점

1. 모델이 아니라 파이프라인을 제품으로 인정한다. 모델을 배포하는 게 아니라 파이프라인을 배포한다.
2. 파이프라인을 구축하려면 시스템을 명확히 정의된 작은 요소로 나눈다.
3. 세상이 변하면서 결국 모델의 정확도는 불가피하게 떨어질 것이고, 이에 대비해야 한다.

프로덕션 준비 - 피처 스토어

이전 섹션에서 우리는 주로 모델을 중심으로 한 머신러닝 파이프라인을 다뤘다. 그러나 피처(feature)는 프로덕션용 머신러닝에 또 다른 핵심 요소이고, 모델만큼 관리하기 복잡하다.

전통적인 애플리케이션을 개발한다면, 개발자는 코드를 프로덕션에 배포하는 일만 잘하면 된다. 게다가 잘 갖춰진 데브옵스(DevOps) 도구들과 프로세스로 빠르고 효율적인 코드 배포를 진행할 수 있다. 심지어 하루에 몇 번씩 코드를 프로덕션으로 배포하기까지 한다.

그러나 머신러닝 기반 애플리케이션을 개발하려고 하면, 모델과 피처까지 프로덕션으로 내보내야 해 번거로움이 추가된다. 데이터 사이언티스트는 소프트웨어 엔지니어가 아니다. 이들은 모델과 피처를 개인 노트북에서 구축한다. 이렇게 개발된 모델과 피처가 프로덕션에 배포될 때 어떤 과정을 거치는지 살펴볼 필요가 있다.

모델을 위해서는, 머신러닝 파이프라인을 구축하고 모델 라이프 사이클을 관리하는 일련의 도구가 존재한다. ML옵스 플랫폼은 데이터 사이언티스트가 모델 학습, 실험 실행, 모델 검증과 프로덕션 배포까지 할 수 있도록 돕는다. 피처를 위해서는, 제공되는 도구가 비교적 불완전하다. 피처의 프로덕션 배포는 피처 변환(또는 피쳐 파이프라인)까지 함께 배포해야하고, 모델 학습과 온라인 인퍼런스(inference)를 위해 일관된 피처 값을 제공할 수 있도록 조정하는 과정이 필요하기 때문에 특히 더 어려운 과제다. 데이터 사이언티스트는 주로 피처 변환 결과물을 데이터 엔지니어에게 전달해, 엔지니어들이 프로덕션용 코드로 피처 파이프라인을 재구현하기를 바란다. 이는 상당히 복잡한 절차로, 일반적으로 몇 주 또는 몇 개월의 소요 시간(lead time)이 필요하다.

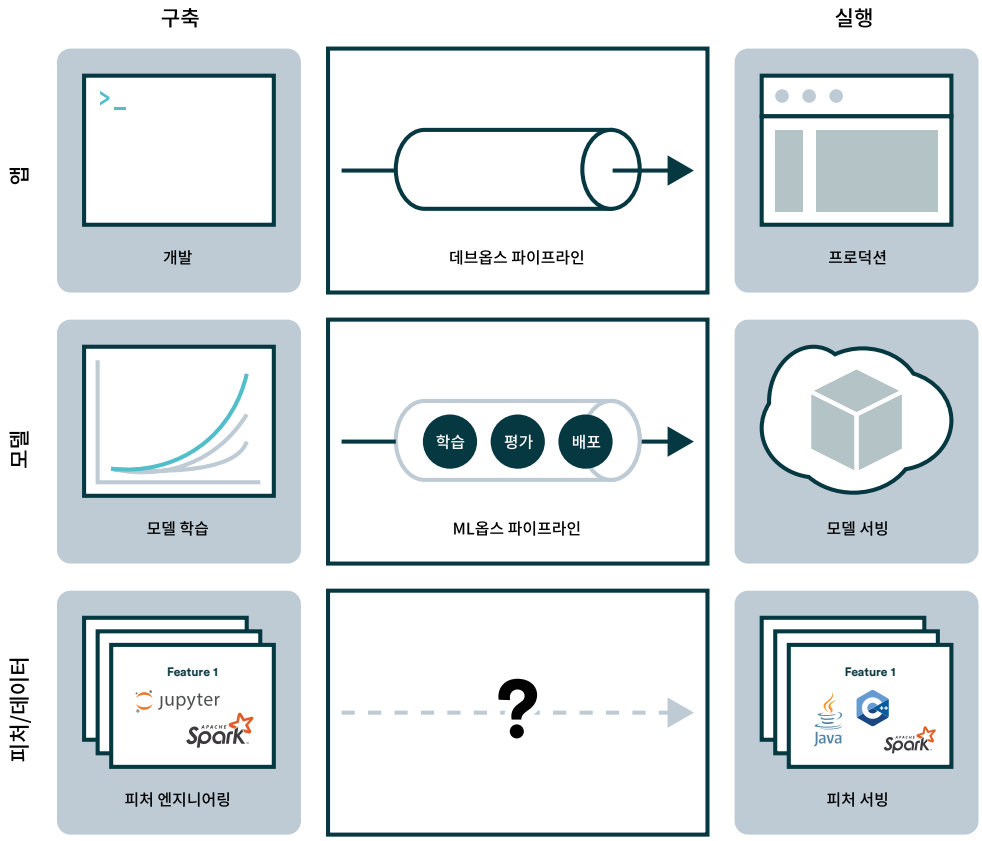


그림 15

피처 관리를 위한 도구는 거의 존재하지 않는다

피처 스토어

이런 상황에서 피처 스토어는 유용하다. 피처 스토어란 한마디로 피처들을 위한 중앙 저장소이다. 피처 스토어는 미가공 데이터를 피처 값으로 변환하고 저장하며, 모델 학습과 온라인 예측을 위해 제공된다. 이 단계를 자동화해, 피처 스토어는 데이터 사이언티스트가 피처를 몇 개월 단위가 아니라 몇 시간 단위 내로 구축하고 배포할 수 있도록 한다. 또 데이터 사이언티스트가 피처 개발부터 배포까지 완전히 통제할 수 있도록 해, 피처 엔지니어링 과정에서도 데브옵스와 유사한 수준의 도구들이 적용될 수 있도록 한다. .

피처 스토어는 데이터 사이언티스트가 아래와 같은 일을 가능하도록 한다.

- 우수한 피처를 모아놓은 라이브러리를 동료들과 함께 구축할 수 있다. 로컬 노트북으로 피처 변환을 관리하는 대신, 일련의 기본적인 피처 정의(definition)를 생성하여 깃(Git) 기반의 저장소에서 관리한다. 이런 피처 정의는 이후 피처 스토어에 적용된다. 결과적으로 이는 피처 정의의 일관성을 향상시키고, 새로운 피처를 사용하여 협업할 수 있도록 돕는다.
- 피처를 프로덕션으로 즉시 배포할 수 있다. 피처 정의가 피처 스토어에 적용되면 피처 변환을 자동화하고 피처 값을 맞춤 조정한다. 이 값은 학습 데이터 세트를 만드는 데 사용되거나 실시간 인퍼런스를 위해 온라인으로 제공될 수 있다.
- 피처를 공유하거나 발견하고 재사용할 수 있다. 피처와 연관 메타 데이터, 변환 로직과 값들을 모두 중앙 피처 저장소에서 관리하고 검색할 수 있다. 데이터 사이언티스트는 손쉽게 기존 피처를 발견하고 여러 모델에 재사용할 수 있다.

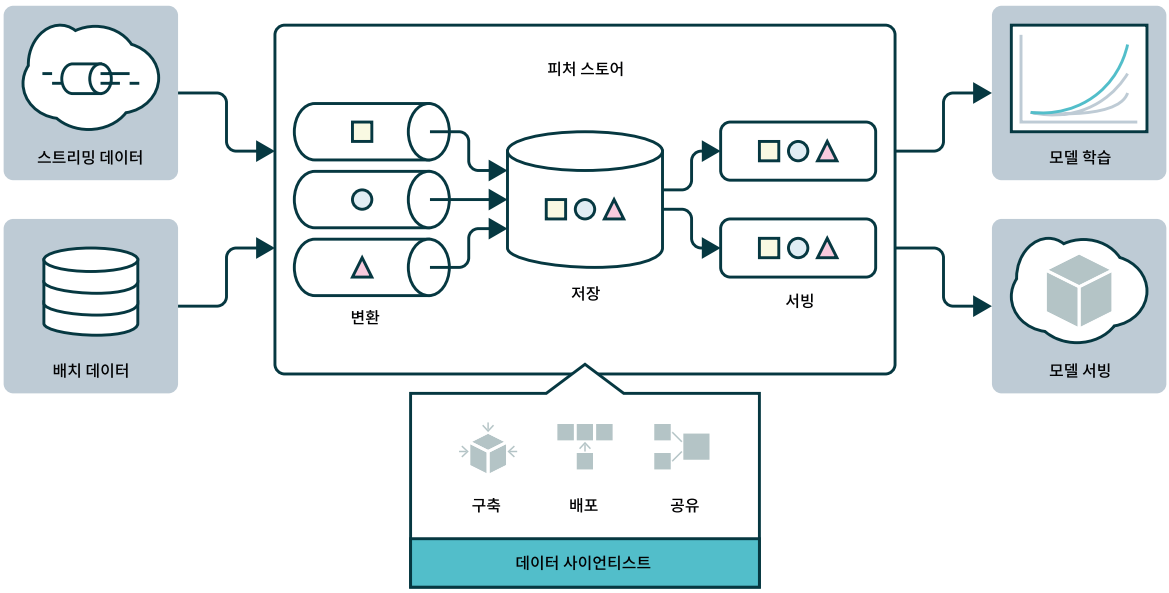


그림 16

피처 스토어: 모델과 데이터 사이의 인터페이스

피처 스토어는 우버 미켈란젤로(Uber Michelangelo) 팀에서 최초로 공개했다. 그 후, 에어비앤비와 넷플릭스 등 많은 기업이 문제 해결을 위해 자체 내부 피처 스토어를 구축했다. 그러나 피처 스토어 구축이 복잡한 만큼 기술력이 약한 조직에겐 여러모로 접근하기 어려운 대상이었다.

그러나 지난 한 해 동안, 다수의 오픈소스 및 상업용 피처 스토어가 등장했다. 이들은 기존의 데이터 레이크, 데이터 창고(warehouse), 이벤트 스트리밍 플랫폼, 데이터 처리 엔진, 파이프라인 관리도구 그리고 머신러닝 플랫폼과 연동되어, 피처 관리 역량을 갖추면서 기존 인프라를 강화한다.

주요 시사점

1. 피처를 구축하고 프로덕션으로 내보내는 과정은 머신러닝 프로덕션 준비 과정 중 가장 어려운 부분에 속한다.
2. 피처 스토어는 데이터 사이언티스트가 피처를 빠르고 쉽게 구축, 배포, 공유할 수 있도록 돕는다.
3. 피처 스토어는 기존의 머신러닝 인프라를 보완해 피처 라이프 사이클에 데브옵스 같은 역량을 갖추도록 한다.

테스트

전통 소프트웨어는 세상에 대한 분명하고 정적인 가정들 위에 일정한 고정 규칙을 부여하면서 구축된다. 모든 내용이 사전에 정의되므로 모든 규칙을 일일이 테스트(유닛 테스트)하거나 그룹 단위로 테스트(연동 테스트)하는 건 비교적 단순한 일이다.

하지만 이름에서 유추할 수 있듯이 머신러닝이란, 지속해서 변하는 데이터에서 동적으로 규칙을 찾는 일을 의미하며 그렇기 때문에 머신러닝의 테스트는 훨씬 더 복잡하다. 머신러닝에서 테스트는 움직이는 표적을 조준해 맞추려는 일 같다. 시스템이 어떻게 동작하는지는 데이터의 동적 특성과 모델 구성을 위한 다양한 선택 요소들에 의해 좌우된다.

테스트는 탐색과 밀접한 관련이 있다. 예를 들어 데이터의 통계 분포와 관련한 정보 등, 테스트의 기준이 되는 정보들을 탐색을 통해 알 수 있기 때문이다.

데이터 테스트

머신러닝 프로젝트에선 데이터가 코드만큼(또는 그 이상으로) 중요하다. 코드의 유닛 테스트가 입력 추정 값을 정의하고 테스트하는 것처럼, 데이터 검증 테스트는 학습과 인퍼런스에 사용되는 데이터 입력에 대해서 같은 역할을 해야 한다. Null 값, 피처 내 비정상적 통계 분포 그리고 피처들 사이의 관계를 테스트해야 한다.

예를 들어 입력값이 랜덤한 영문 텍스트로 예상될 때, 이를 테스트하는 방법 중 하나는 ‘the’가 가장 자주 발견되는 단어일 것이라는 이미 잘 알려진 가정을 활용하는 것이다. 만약 ‘the’가 아닌 다른 단어가 가장 자주 발견된다면, 아마 해당 데이터는 예상치 못한 편향성이 있다는 걸 의미하거나, 어쩌면 데이터 일부가 우연히

스페인어로 구성돼 있을 가능성도 있다. 가정 검증에 유용한 또 다른 방식은 남녀 비율이 동등하다고 전제할 때 실제 입력 데이터에서도 비율이 동등하게 나뉘어 있는지 확인해 보는 방법이다.

또 피처들의 상관관계도 테스트해야 한다. 두 개 이상의 피처에서 높은 수준의 상관관계가 발견된다면, 모델의 성능과 정확도에 부정적인 영향을 미칠 수 있다. 예를 들어, 제품 관련 데이터 내에 부가세 포함 가격과, 미포함 가격과 같은 유사한 칼럼(column)이 있다면 주의깊게 살펴봐야 한다.

모델 테스트

데이터와 관련된 가정이 모두 검증되었다면 모델과 학습 테스트로 넘어갈 수 있다. 오프라인 데이터에 긍정적인 정확도를 보여 주는 모델을 무작정 배포한 뒤, 실제로 잘 작동할 거라고 낙관해선 안 된다.

학습 단계에선 각각의 초매개변수(hyperparameter)의 영향을 테스트할 수 있다. 격자 탐색(grid-search) 또는 더 고차원의 탐색 전략을 실행해 신뢰성 이슈를 찾아내고, 예측 성능을 개선할 수 있다. 또 학습 세트와 검증 세트에서 분리된 추가 테스트 세트를 최대한 활용해야 한다.

모델 배포 시, 오프라인 평가 지표와 상용화 모델의 실제 영향도 사이의 상관관계를 테스트해야 한다. 예를 들어 소규모 A/B 테스트 환경에서 모델의 오프라인 정확도와 실제 웹사이트의 클릭률 등의 실제 영향도 사이의 상관관계를 측정할 수 있다.

또 다른 스모크 테스트(smoke test) 기법은 신규 배포할 최신 모델을 가장 단순한 기초 모델과 비교해 테스트하는 것이다. 모델을 완전히 투입하기 전에 실제 프

로덕션 데이터 중 일부를 신규 모델에 적용해 카나리아 테스트(canary test)를 진행할 수 있도록 한다.

인프라 테스트

머신러닝 파이프라인은 하루 단위로 최대한 재현 가능해야 한다. 완벽하게 모든 것을 컨트롤 하기는 어렵지만, 최대한 노력이 필요하다.

학습 파이프라인의 재현 가능성을 테스트하려면 우선 두 개 이상의 모델에 같은 데이터를 적용해 나란히 학습시킨 뒤, 평가 지표별로 격차를 측정한다. 또, 학습 중간 체크포인트에서 부터 이어서 예측한 대로 학습을 지속할 수 있는지 측정한다. 처음 데이터 검증 단계에서부터 모델 배포 단계까지, 모든 파이프라인에 연동 스모크 테스트를 생성하는 일도 잊지 말아야 한다.

예상치 못한 상황과 인적 과실이 발생했을 때 안정적인 모델로 롤백하는 일도 중요하다. 롤백 인프라는 다른 모든 테스트가 실패했을 때 사용할 최후의 보루로, 항상 꾸준히 테스트해야 한다.

주요 시사점

1. 머신러닝의 동적 특성 때문에, 테스트는 더욱더 중요하다.

2. 코드 테스트는 당연한 과정이며 데이터 테스트는 가장 중요하다.

3. 파이프라인의 재현 가능성은 안전한 배포 과정의 핵심이다.

배포와 인퍼런스

배포란 API, 애플리케이션으로 또는 다른 방법으로 머신러닝 모델을 외부에 제공하는 행위를 말한다. 인퍼런스(inference)란 모델을 배포한 후 수행하는 행위로, 결과 예측, 입력값 분류 또는 데이터 클러스터링 모두 인퍼런스에 해당한다.

가장 먼저 물어야 할 질문은 배치 인퍼런스(batch inference)와 온라인 인퍼런스(online inference) 중 무엇을 목표로 하느냐다. 다음 필요한 질문은 전형적인 중앙 클라우드 인퍼런스 방식과 연산 요구 사항을 고객 하드웨어에 분산하는 방식(엣지 인퍼런스, edge inference) 중 어떤 방식을 선택하느냐다.

배치 인퍼런스

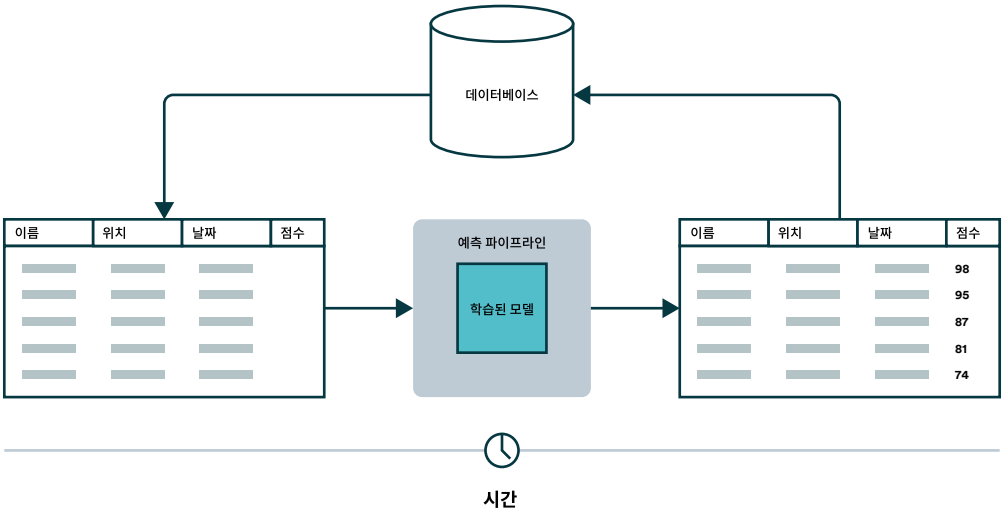


그림 17

배치 인퍼런스란 여러 요청 사항을 배치(batch)로 묶어서 한번에 같이 추론(inference)하는 일이다. 배치 인퍼런스는 요청이 들어올 때마다 즉시 실시간으로 값을 제공하지 않고, 어느정도 시간이 소요된 후에 일련의 질문에 대한 결과를 낸

다. 예를 들어 하루 1회, 고정 간격이 지난 후 배치가 처리된다. 배치 인퍼런스는 내부적으로는 소프트웨어 개발의 전형적인 캐싱 전략과 유사한 특성을 보인다.

배치 인퍼런스는 레이턴시(latency)가 문제가 되지 않는 모든 시나리오에 적합하다. 예를 들어, 영업 조직을 위해 신규 잠재적 고객들의 적합도 점수 계산에 사용되는 모델에선, 점수 계산에 24시간 레이턴시가 있더라도 큰 문제가 되진 않을 것이다.

ML옵스의 맥락에서, 고정 간격에 기반한 인퍼런스는 엔지니어가 병렬 작업을 더 효율적으로 진행할 수 있도록 돕고, 더 예측 가능한 방식으로 연산을 처리할 수 있다는 점에서 의미가 있다. 모델을 실시간 온라인 API로 연결하고 예측하기 어려운 양의 트래픽을 처리할 필요가 없기 때문에, 외부 환경과 접해있는 인프라는 관리하고 모니터하기에 훨씬 간편해진다. 그리고, 문제가 발생하더라도 고객에게 노출되기 전에 대처할 시간을 확보할 수도 있다.

온라인 인퍼런스

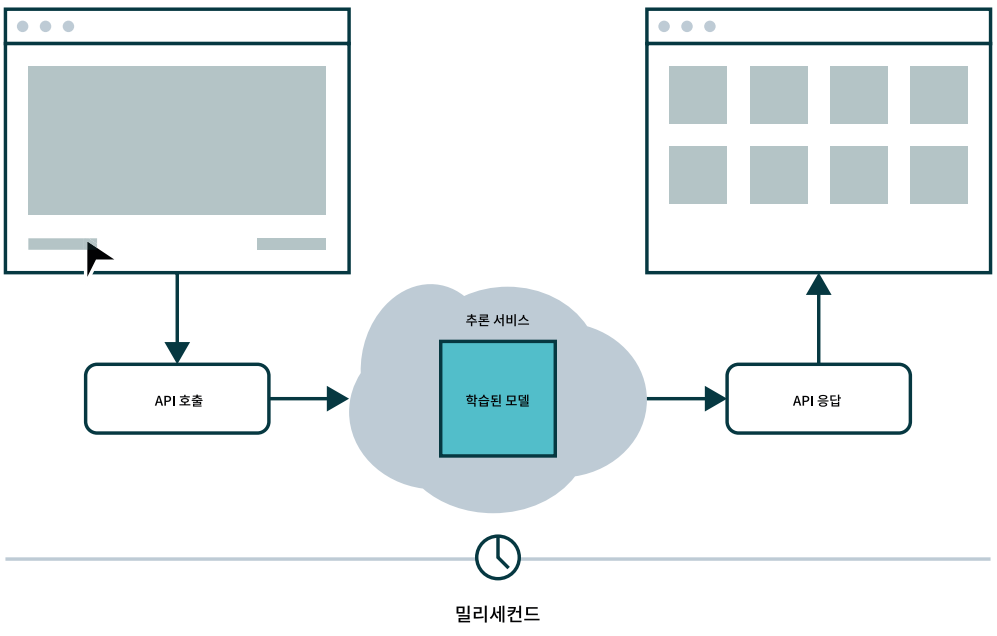


그림 18

온라인 인퍼런스란 실시간으로 추론을 실행하는 절차다. 모든 요청은 모델에서 즉각 처리된다.

온라인 인퍼런스는 모델이 제공하는 값이 즉시 필요한 모든 상황에서 적합하다. 예를 들어, 구급차가 가야 할 최적의 경로를 예측하는 모델이든, 향후 10분 동안 주식 시장의 변동성을 예측하는 모델이든 모두 일분일초가 시급하다. 이때, 모델의 추론이 지연되면 추론값의 가치는 사라진다.

ML옵스의 맥락에서, 온라인 인퍼런스는 더 많은 것이 요구된다. 요청이 즉각 처리되어야 하고, 모델이 예측이 어려운 실시간 트래픽에 직접 연결되는 상황이라면, 상황은 굉장히 빠르게 악화될 수 있다. 예측에서 오류나 편향이 생기면 곧바로 고객에게 노출되기 때문이다. 또 일시적인 트래픽 과부하를 감당하려면 시스템이 자동으로 확장 가능해야 한다. 일반적으로, 확장은 쿠버네티스(Kubernetes) 같은 자동 확장 클러스터로 처리한다. 모니터링 요구사항은 훨씬 더 높고, 이슈 발생 시 실시간에 가까운 반응 시간으로 대응해야한다.

엣지 인퍼런스

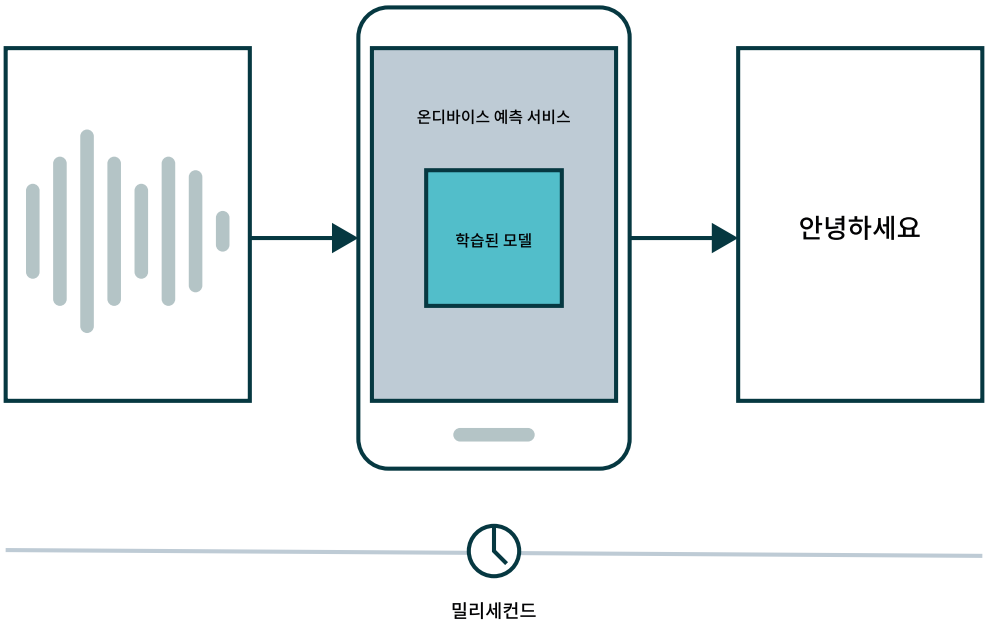


그림 19

전형적인 배포와 인퍼런스 방식은 중앙화 시스템을 사용하는 것이다. 주로 모델의 요청을 처리하도록 클라우드 업체 중 한 곳 위에서 쿠버네티스 클러스터를 구성한다. 최근에 인기 있는 또 다른 옵션은 고객 하드웨어의 연산 능력을 활용하는 건데, 이 방식을 바로 엣지 인퍼런스라고 한다.

애플리케이션이 중앙 클라우드 API를 사용하도록 구현하는 대신, 모델을 애플리케이션 일부로 포함해 사용자 디바이스 또는 브라우저에 직접 배포한다. 휴대전화에서 구동되는 음성인식(speech-to-text) 모델이 좋은 예다. (높은 데이터 전송량과 레이턴시 요구 사항으로 인해) 오디오가 서버로 전송되지 않는 대신, 휴대전화 디바이스에 탑재된 모델이 오디오를 텍스트로 즉시 변환할 수 있다. 텍스트 형태로 변환되면, (낮은 데이터 전송량과 레이턴시로) 서버에 더 단순하게 텍스트 기반 요청을 보낼 수 있고, 서버에는 이러한 변환된 텍스트 입력값을 이해할 수 있는 별도의 모델이 존재한다.

엣지 인퍼런스의 명확한 장점은 요청이 많을수록 사용 가능한 엣지 디바이스가 늘어나기 때문에, 비용을 들이지 않고도 완벽한 확장 가능성을 구현할 수 있다는 점이다. 심지어 엣지 디바이스 모델은 완전히 자체적으로 작동하기 때문에 클라우드 서비스는 인퍼런스가 진행되는지 여부조차 인지할 필요가 없다. 그러나 단점은 각각의 디바이스에 탑재된 수많은 버전을 유지하는 일이 까다로울 수 있다는 거다. 업데이트 방식에 따라, 프로덕션에서 동시에 구동되고 있는 다양한 버전의 모델을 관리할 준비가 필요할 수 있다.

또 서로 다른 디바이스와 환경에 따른 문제를 직면해야 한다. 휴대전화 디바이스 1,000대는 작게나마 서로 다른 환경이 1,000개라는 걸 의미한다. 주변 환경에서 엣지 인퍼런스를 얼마나 잘 캡슐화(encapsulate)하느냐에 따라, 엣지 디바이스를 다루기 위한 복잡한 테스트 인프라에 투자해야 하는 상황이 발생할 수 있다.

또 다른 잠재적 문제는 보안이다. 물론 모든 데이터를 외부 디바이스로 전송하는 건 아니겠지만, 우리는 해당 데이터로 학습된 모델을 외부로 전송해야 한다. 이 점은 일반적으로 악용하기 매우 어렵지만, 여전히 고려할 때 빠트려서는 안 될 보안 문제다.

주요 시사점

1. 가능한 한 배치 인퍼런스를 사용한다. 온라인 인퍼런스는 최후의 수단이다.
2. 가능한 한 엣지 인퍼런스를 사용한다. 비용을 들이지 않고도 완벽한 확장성을 달성하기 때문이다.
3. 복잡한 셋업은 강력한 모니터링을 요구한다.

결론

머신러닝에 투자를 한다면, 기존에 해결할 수 없었던 비즈니스 상황들을 해결할 수 있다. 예로, 이미지를 콘텐츠 별 자동 분류하는 일 같은 문제가 여기에 포함될 것이다. ML옵스는, 머신러닝과 다르게, 모든 비즈니스 문제를 직접 해결할 수 있는 건 아니다. 대신, ML옵스의 가치는 머신러닝에 대한 투자가 실질적인 가치 창출로 더 빠르게 이어질 수 있도록 한다는 점에 있다.

조금 더 전통적인 산업군에 비유하면, 머신러닝은 화물 운송에 가까우며 ML옵스는 컨테이너화(containerization)에 가까운 개념이라고 할 수 있다. 국제 운송에서 컨테이너 수송의 역할처럼, ML옵스는 하나의 프로세스인 동시에 인프라이기도 하다. ML옵스는 단기간에 가시적인 결과를 내기 위한 용도는 아니며, 광범위한 분야의 이해관계자의 참여를 필요로 한다. 그럼에도 조직의 머신러닝 업무가 확장될 수록 누릴 수 있는 혜택이 증가하는 효과를 볼 수 있다.

탄탄한 소프트웨어 공학 배경지식을 갖춘 데이터 사이언티스트 같은 일부 사람들에겐, ML옵스가 그렇게 새로운 개념으로 다가오진 않을 것이다. 그러나 다양한 관계자들과 협업을 하는 상황에서는, 최대한 모범사례를 따를 수 있도록 강조하는 것이 중요할 때가 많다.

ML옵스 도입 시 아래 네 가지 핵심 모범사례를 도입하길 권장한다:

- 모델의 재현 가능성을 갖추기 위한 버전화(versioning)
- 더 나은 시스템 구축 협업을 위한 파이프라인 도입
- 프로덕션 모델의 기준을 확립하기 위한 테스트
- 시간 절약하고 최대한 자동복구가 가능한 시스템을 위한 자동화

시장엔 다양한 ML옵스 도구들이 있고, 자체 구축하거나 외부 엔드 투 엔드 플랫폼을 구매하는 등 전체 ML옵스 툴체인 구축에 관련해서도 수많은 전략이 존재한다. 본 자료에선 이 분야에 대해 고려해야 할 핵심 영역을 살펴보았으며, 필요에 따라 결정을 도울 수 있도록 주요 시사점을 제시했다.

ML옵스의 궁극적 목적은 최대한 짧은 기간 안에 아이디어 단계의 모델을 프로덕션까지 배포하고, 나아가 최소한의 위험 부담으로 시장에 출시할 수 있도록 기술적 마찰을 줄이는 것이다. 이 목적을 염두하여 어떤 ML옵스 도구를 선택할지에 대한 결정을 내리길 바란다.

저자 소개

학습시키고, 평가하고, 배포하고, 반복하라. Valohai는 데이터 추출부터 모델 배포까지 모든 단계를 자동화하는 유일한 ML옵스 플랫폼입니다. Valohai는 PARC, LEGO Group, Twitter, JFrog 등 스타트업부터 대기업까지 모든 유형의 조직이 사용하고 있습니다.

자세한 내용은 www.valohai.com 홈페이지를 방문하세요.

SigOpt는 AI 모델 개발 프로세스를 확장하기 위한 실험 관리와 모델 최적화를 위한 가장 완성도 높은 ML옵스 플랫폼을 제공합니다.

자세한 내용은 www.sigopt.com 홈페이지를 방문하세요.

Uber Michelangelo 제작진이 설립한 Tecton은 신규 피처 엔지니어링부터 실시간 예측을 위한 온라인 서빙까지, 피처의 모든 라이프 사이클을 관리하는 최초의 엔터프라이즈용 피처 스토어를 제공합니다.

자세한 내용은 www.tecton.ai 홈페이지를 방문하세요.

Superb AI

Superb AI는 머신러닝 데이터 가공, 관리, 분석을 자동화 & 효율화하는 플랫폼인 Superb AI Suite를 제공하고 있으며, 데이터 관점에서 MLOps를 이끌어 나가고 있습니다. Suite를 통해 데이터 라벨링에서 시간과 비용을 절약하고, 엔터프라이즈 레벨의 데이터 파이프라인을 구축하세요. Suite는 AI 엔지니어가 데이터 이슈보다 머신러닝 개발에 더 집중할 수 있도록 돕겠습니다.

자세한 내용은 www.superb-ai.com홈페이지를 방문하세요.