EBOOK

현실적인 머신 러닝 엔지니어링

ML 프로젝트 계획에서 프로덕션까지 아우르는 단계별 가이드

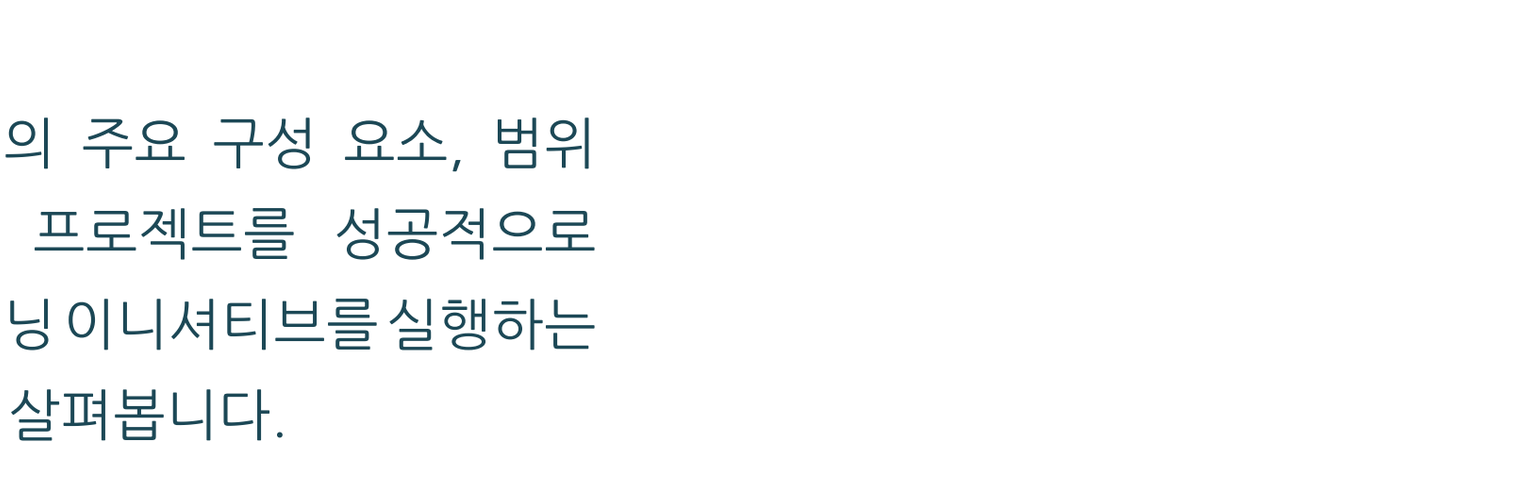
MANNING

이 eBook은 Manning에서 출간한 “Machine Learning Engineering in Action”(저자: **Ben Wilson**)에서 발췌했습니다.

databricks

서문

이 eBook에서는 머신 러닝 엔지니어링의 주요 구성 요소, 범위 설정에서 배포까지 대규모 머신 러닝 프로젝트를 성공적으로 구현하는 방법, 기업에서 성공적인 머신 러닝 이니셔티브를 실행하는 데 참여하는 주요 팀 및 페르소나에 대해 살펴봅니다.



목차

섹션 01

ML 엔지니어링이라는 개념

섹션 1

ML 엔지니어링이라는 개념

머신 러닝(ML)은 흥미롭습니다. 전문가가 아닌 일반인의 눈에 머신 러닝은 마법에 가까운 예언 능력을 발휘하고, 까다로운 문제에 신기하고 기적적인 답을 제시하는 것 같이 보일 정도입니다. ML은 기업에는 돈을 벌어주고, 감당하기 어려운 대규모 작업에 자율적으로 착수하며, 단조로운 업무를 처리해야 하는 부담을 덜어줍니다. 다만, 당연한 사실을 지적하자면 머신 러닝은 쉽지 않습니다. 전문 ML 실무자는 수천 개의 알고리즘을 사용하고, 데이터 엔지니어링(DE)부터 고급 통계 분석과 시각화에 이르기까지 다양한 기술을 활용해야 하므로 실로 복잡하고 기가 죽는 것이 당연할 정도입니다.

ML 엔지니어링은 이런 엄청난 복잡성에 시스템을 적용하는 개념입니다. 일련의 표준, 도구, 프로세스는 물론 방법론까지 포함한 구성으로, 비즈니스 문제점이나 요구 사항을 해결할 때 중도 포기하거나 잘못 이해하거나 관련성이 없는 업무에 낭비되는 시간을 최소한으로 줄이는 것이 목표입니다. 말하자면 ML 엔지니어링이란 본질적으로 ML 기반 시스템을 만드는 로드맵이라 할 수 있습니다. 이것을 프로덕션에 배포하는 데만 그치는 것이 아니라 앞으로 몇 년 동안 유지 관리, 업데이트하여 비즈니스에서 ML로 인해(제대로 활용했을 때) 얻을 수 있는 것으로 입증된 효율성, 수익성과 정확성과 같은 보상을 얻게 해줍니다.

이 eBook은 여러분에게 이 시스템을 안내할 로드맵입니다. 그림 1.1을 보시면 머신 러닝으로 프로젝트의 계획 단계를 다루는 검증된 프로세스들이 나와 있습니다. 여기에는 난해하고 헷갈리기 쉬운 비즈니스 요구 사항을 ML 작업용 언어로 살펴보는 단계도 포함됩니다. 다음으로 실험 작업의 표준 방법론을 소개합니다. 종합적이고 유지 관리 가능한 최소 기능 제품(Minimum Viable Product, MVP) 제작을 위한 도구와 코딩 표준 위주로 알아보겠습니다. 마지막으로, 프로덕션 등급의 유지 관리 가능한 코드를 작성하는 데 쓰이는 다양한 관련 도구, 기법과 미묘한 차이점에는 어떤 것이 있는지 알아봅니다. 관건은 확장 가능하면서 동시에 문제 해결도 쉬운 코드를 작성하는 것입니다.

그러나 ML 엔지니어링에서 그림 1.1에 나와 있는 경로만 중요한 것은 아닙니다. 이러한 각 단계에 속하는 방법론도 중요합니다. 그러한 방법론 각각이 프로젝트의 성패를 좌우할 수도 있기 때문입니다. 이러한 방법론에는 데이터 사이언스 팀이 회사에 문제를 언급하는 방식, 연구를 수행하는 방식, 실험의 상세한 정보, 코드 작성 방식, 로드맵을 정한 경로를 따라 사용한 다양한 도구와 기술 등이 있습니다. 결국 이 모든 것은 어떤 프로젝트든 최악의 결말, 즉 '중도 포기' 비율을 대폭 줄일 수 있는 방편입니다.

궁극적으로, ML 작업의 최종 목표란 바로 문제 해결입니다. ML 엔지니어링의 개념을 받아들여 효과적인 프로젝트 작업의 길을 따라가다 보면, 유용한 모델링 솔루션을 얻는다는 최종 목표를 훨씬 빠르고 저렴하게 달성할 수 있으며 그저 '임기응변'으로 움직이며 운에 맡길 때보다 성공 확률을 크게 높일 수 있습니다.

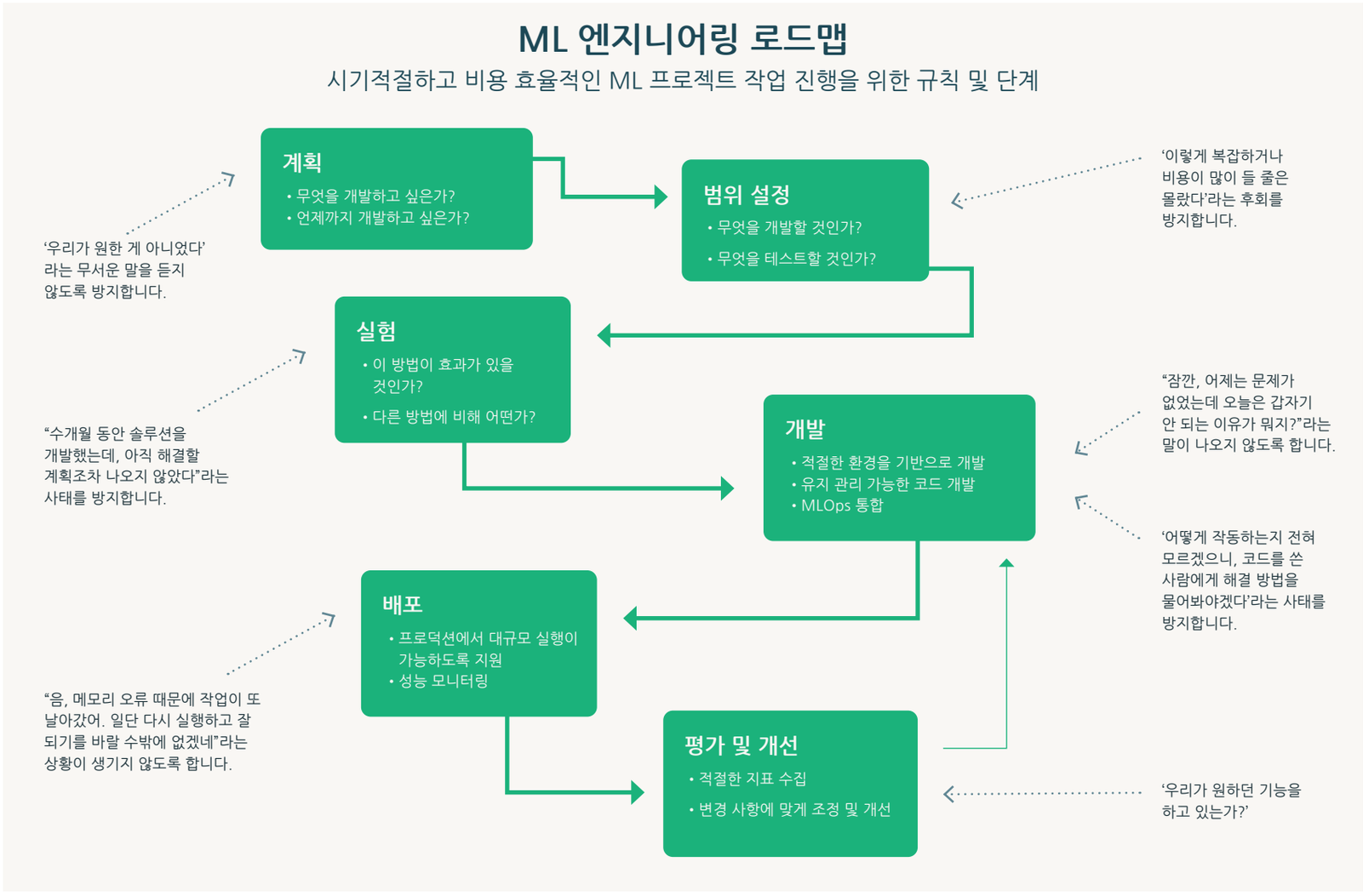


그림 1.1
ML 엔지니어링 로드맵을 보면 성공적인 ML 솔루션 제작에 유익한 것으로 검증된 작업 단계가 무엇인지 알 수 있습니다. 프로젝트에 따라 단계를 추가해야 하는 경우도 있지만 (특히 엔지니어링 팀이 더 많이 투입되는 경우) 여기에 소개한 것이 기본 단계로 각종 ML 기반 프로젝트에 이 정도는 꼭 포함해야 합니다.

섹션 1: ML 엔지니어링이라는 개념

1.1 ML 엔지니어링이 필요한 이유

간단히 말해, ML은 *어렵습니다*. 게다가 개연성 있는 예측을 대규모로, 그것도 안정적인 빈도로 제공한다는 의미에서 '제대로' ML을 해내기란 더욱 어려운 일입니다. 무수히 많은 전공 분야가(NLP, 예측, 딥러닝, 기존의 선형 모델링, 트리 기반 모델링 등) 존재하고, 진행 중인 리서치에는 엄청난 이목이 쏠리는 데다 특정 문제를 해결할 목적으로 빌드된 알고리즘도 워낙 많은 상황이라 배워야 할 것은 넘치게 많고, 그중 지극히 미세한 부분만 배우기도 놀랄 만큼 어렵습니다. 이런 복잡한 상황에 설상가상으로 모델을 개발할 수 있는 플랫폼마저 라즈베리파이(Raspberry Pi)부터 초대형 NVIDIA GPU 클러스터에 이르기까지 무궁무진하므로, 세상에 존재하는 플랫폼 자체만으로 완전히 새로운 정보가 한 세트 생깁니다. 한 사람이 평생 걸려도 다 배우기란 불가능할 정도입니다.

또한, 데이터 사이언티스트가 친숙해져야 할 추가적인 영역도 있습니다. 예를 들어, 중급 데이터 엔지니어링 기술(데이터 사이언스에 쓸 데이터를 입수할 출처가 필요하므로)과 소프트웨어 개발, 프로젝트 관리, 시각화 및 프레젠테이션 기술 등이 필요합니다. 게다가 이런 영역은 계속 늘어나기 때문에 필요한 경험을 전부 해보기란 아무래도 쉽지 않습니다. 그러니 이 모든 상황을 고려해 봤을 때 프로덕션 등급 ML 솔루션 개발에 필요한 모든 기술을 갖추기란 대부분의 개인에게는 벅찬 일입니다.

ML 엔지니어링의 목표는 이런 기술 목록을 되짚으며 데이터 사이언티스트가 각각의 기술에 통달하게 하는 것이 아닙니다. 그보다는 그런 여러 기술의 특정 측면을 취합해 하나의 집합체로 취급하여, 데이터 사이언티스트에게 개연성 있는 의미를 지니도록 세심하게 구성한 집합체로 제시해야 합니다. 이 모든 것은 *ML 프로젝트를 프로덕션으로 발전*시킬 가능성을 높이고, 지속적인 유지 관리와 개입이 없어도 존속할 수 있게 하는 것이 목표입니다.

사실 ML 엔지니어가 범용 알고리즘 사용 사례에 맞춰 애플리케이션과 소프트웨어 프레임워크를 만들 줄 알아야 할 필요는 없습니다. 대규모 스트리밍 수집 ETL 파이프라인을 직접 작성해야 할 가능성도 낮습니다. JavaScript로 상세한 애니메이션 프런트엔드 시각 자료를 만들 필요도 없을 것입니다.

ML 엔지니어는 모듈식 코드를 작성하고 유닛 테스트를 구현할 정도의 *소프트웨어 개발 기술만* 알면 됩니다. 논블로킹(nonblocking) 비동기 메시지 중개 같은 복잡한 것은 몰라도 됩니다. 모델에 대한 피처 데이터 세트를 개발할 수준만큼의(여기에 대해 ETL을 예약할 수 있을 정도의) *데이터 엔지니어링 기술*만 알면 될 뿐, PB 규모 스트리밍 수집 프레임워크는 구축하지 않아도 됩니다. 자신의 조사와 모델의 기능을 명확히 전달하는 도표와 차트를 만들 수준의 시각화 기술만 필요하고, 복잡한 UX 구성 요소로 동적인 웹 앱을 개발할 필요가 없습니다. 또한, *프로젝트 관리 경험*은 프로젝트를 적절히 정의하고 범위를 설정하여 관리할 정도만 알면 될 뿐, PMP 인증을 받을 필요가 없습니다.

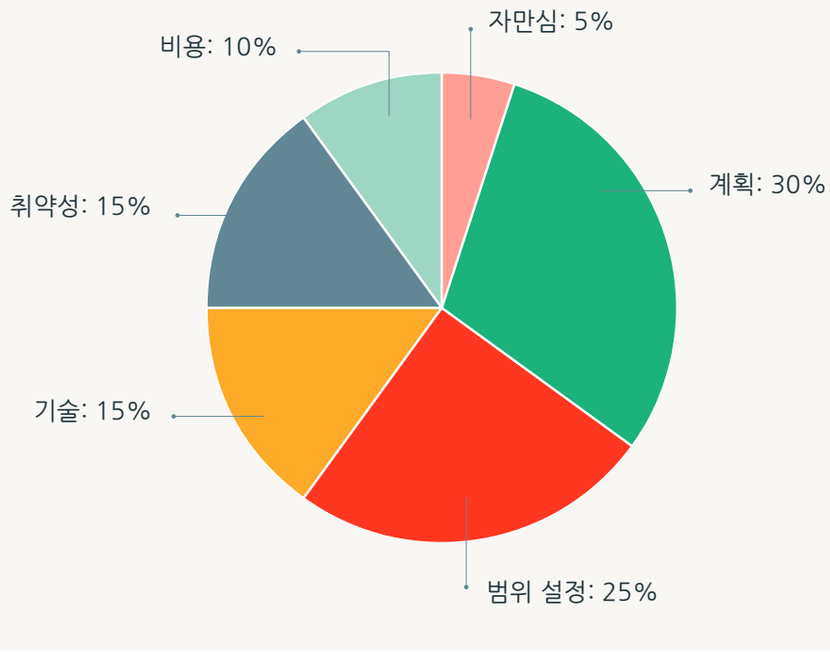


그림 1.2
ML 프로젝트가 실패하는 주된 이유. 그림 1.2는 프로젝트가 실패하는 주된 이유를 대략적인 추정치로 보여줍니다. 가장 흔한 이유는, 데이터 사이언스 팀이 대규모 프로덕션 등급 모델을 사용하여 특정한 요구 사항을 해결하는 데 익숙하지 않거나, 기업에서 원하는 결과가 무엇인지 이해하지 못했기 때문이었습니다.

많은 기업에서 ML에 집중 투자하여 엄청난 연봉을 주고 데이터 사이언티스트를 고용해 대규모 팀을 꾸리고 막대한 재정적, 시간적 리소스를 투입하고 있지만, 이런 프로젝트는 결국 엄청나게 빠른 속도로 실패를 거듭하고 있습니다. 이 eBook에서는 6대 프로젝트 패인을 다루며, 왜 이런 이유로 그렇게 많은 프로젝트가 실패하고, 중도 포기하거나 프로덕션에 돌입하기까지 불필요하게 오랜 시간이 걸리는지 알아봅니다. 각 섹션에 따라 이와 같이 보편적인 문제점의 솔루션은 무엇인지, 그러한 솔루션에 도달하기 위해 거친 프로세스를 설명하였습니다. 이 글을 프로젝트가 정해진 경로를 이탈할 확률을 대폭 낮출 참조 자료로 활용하시면 되겠습니다.

이런 문제는 악의적 의도에서 생기는 것이 아닙니다. 그보다는, 대부분의 ML 프로젝트가 매우 까다롭고 복잡한 데다 일반인에게 설명하기 어려운 알고리즘 소프트웨어 툴링으로 구성되어 있다는 점이 가장 큰 원인입니다(따라서 대부분 프로젝트는 사업부와의 의사소통에서 실패를 경험하게 됩니다). 이렇게 복잡한 솔루션을 운용하고, 활발하게 움직이는 부분도 많은 데다 ML을 활용한 이 새로운 '데이터 중심' 군비 경쟁에서 승리를 거두고 최대한 빨리 수익을 올리려는 기업들이 치열한 경쟁을 벌이고 있습니다. 그러니 주어진 솔루션 하나를 프로덕션 안정기까지 이끌고 가는 위태로운 여정이 중도에 실패하는 일이 그렇게 잦은 것도 놀랄 일은 아닙니다.

이 eBook에서는 이런 요소들이 어떻게 프로젝트에 위험이 되는지 보여주고 각각의 위험을 최소화하는 데 도움이 되는 도구를 알려드리고자 합니다. 성실하고 신중하게 각각의 영역을 살펴보다 보면 이런 위험 대부분을 완전히 제거하기는 어렵더라도 상당히 완화할 수 있습니다.

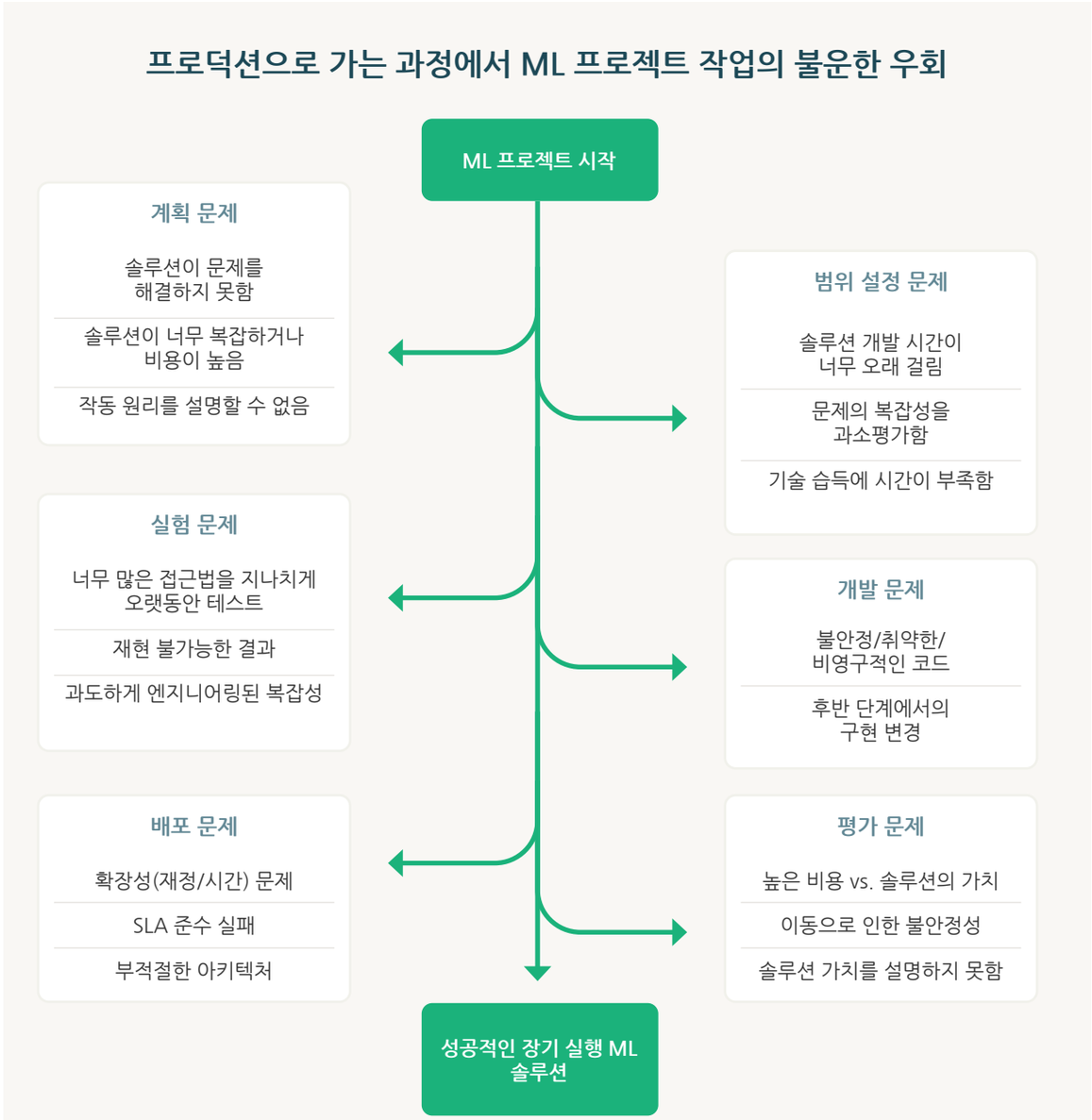


그림 1.3
대부분의 ML 프로젝트에서 실패로 이어지는 경로. 이 6가지 핵심 영역에 집중하지 않는 ML 솔루션은 대부분 프로덕션에 도달하기 전이나 프로덕션에서 실행한 직후에 중도 포기할 가능성이 매우 높습니다.

그림 1.3에는 ML을 사용하여 문제를 해결할 때 누구나 거치는 경로를 표현했습니다. 프로젝트를 시작하는 시점부터 원하던 성공을 이루고 장기적인 실행 및 유지관리 상태에 이르는 여정은 그동안의 고된 노력을 수포로 되돌릴 수 있는 우회로로 가득합니다. 하지만 지식과 기술, 프로세스와 툴링 활용 능력을 키우는 데 집중한다면 이런 6가지 문제 영역은 대개 피할 수 있습니다. 적어도 프로젝트의 완전한 실패를 초래하지는 않을 정도로 문제를 해결할 수는 있습니다.

ML 엔지니어링은 그림 1.3에 표시된 주된 실패 모드를 각각 해결하기 위해 고안한 것입니다. 이 방법론은 실패 가능성을 없애는 것이 핵심입니다. 이를 위해서는 더 나은 의사결정을 내리고, 내부 고객과 손쉽게 소통하고, 실험과 개발 단계에서 재작업을 제거하고, 쉽게 유지 관리할 수 있는 코드 베이스를 개발하고, 데이터 사이언스에 큰 영향을 받는 모든 프로젝트에 모범 사례를 도입하는 프로세스를 적용해야 합니다. 수십 년 전 소프트웨어 엔지니어가 기존의 대규모, 폭포수(Waterfall) 구현 모델을 버리고 좀 더 유연하고 생산적인 애자일 프로세스로 전환한 것과 마찬가지로, ML 엔지니어링 역시 새로운 관행과 도구를 정의하여 데이터 사이언티스트의 소프트웨어 개발이라는 완전히 고유한 영역을 최적화하고자 하는 것입니다.

섹션 1: ML 엔지니어링이라는 개념

1.2 ML 엔지니어링의 핵심 원리

ML 엔지니어링이 무엇인지 대략적으로 이해했으므로 이제 몇 가지 핵심 요소를 살펴보겠습니다.

계획

프로젝트를 철저히 계획하지 않는 것이야말로 프로젝트 실패 원인 중 압도적인 1위입니다. 게다가 프로젝트가 취소된 원인 중 이만큼 팀원의 사기를 저하하는 요소도 없습니다. 여러분이 지금 일하는 회사의 1호 데이터 사이언티스트라고 생각해 봅시다. 입사 첫 주에 마케팅 부서의 임원이 찾아와 마케팅 용어로 현재 어떤 심각한 비즈니스 문제가 있는지 설명했습니다. 마케팅 팀에서는 이메일을 통해 고객이 관심을 보일 만한 판매 소식이 예정되어 있다고 미리 알려줄, 효율적인 소통 방식을 알아내야 합니다. 그런데 이 임원은 상세한 정보는 거의 제공하지 않고, 그저 “클릭률과 이메일 오픈율을 올려야 한다”고만 합니다.

제공된 정보가 이것뿐이고, 마케팅 팀원에게 여러 번 물었는데도 단순히 ‘클릭률과 이메일 오픈율을 높여야 한다’라고 최종 목표만 밝힐 뿐이라면 오히려 선택지가 무수히 많아 갈피를 잡을 수 없을 것입니다. 여러분이 알아서 해야 하는 상황입니다. 어떤 선택을 하게 될까요?

- 콘텐츠 추천에 집중하고 각 사용자에 대한 맞춤형 이메일을 개발할까요?
- 각 사용자에 맞는 제목을 작성하는 NLP 기반 시스템으로 예측을 제공할까요?
- 매일 판매되는 정보를 통해 고객 기반과 가장 관련이 있는 제품 목록을 예측할까요?

단순한 것부터 복잡한 것까지 수많은 선택지와 접근 방식이 있지만 누구의 지도도 받지 못한다면, 이 임원의 기대에 부응하는 솔루션을 만들 가능성은 매우 낮습니다.

제대로 계획을 의논했다면, 진짜 기대하는 바가 무엇인지 알아낼 수 있었을 것입니다. 사용자가 이메일을 읽을 가능성이 가장 높은 시기를 예측하는 것이 핵심입니다. 이 임원은 그저 고객이 일을 하거나, 출퇴근 중이거나 잠자고 있는 중이 아닐 가능성이 가장 높은 시간대를 파악해 다양한 고객 코호트에 시간대별로 적절하게 이메일을 일괄 발송하려 했을 뿐입니다.

안타깝게도 대부분 ML 프로젝트가 이런 방식으로 시작하는 것이 현실입니다. 프로젝트 시작에 관한 소통은 거의 없는 경우가 잦고, 대체로 “데이터 사이언스 팀에서 알아서 할 것”이라고 생각합니다. 하지만 무엇을 개발해야 하고 어떤 기능을 해야 하며 예측의 최종 목표가 무엇인지 적절히 지도하지 않는다면 프로젝트는 실패할 운명이라고 해도 무방하겠습니다.

말하자면, 몇 달 동안 개발에 수고를 들여 특정 사용 사례를 위해 콘텐츠 추천 시스템을 구축했는데 실제로 필요했던 것은 그저 IP 지리적 위치에 기반한 아주 단순한 분석 쿼리가 전부였다면 무슨 결과가 발생했을까요? 프로젝트 취소는 당연하고, 이 시스템을 왜 만들었고 개발 비용은 왜 그렇게 많이 들었는지 윗선의 질문이 쏟아질 것입니다.

그림 1.4와 같이 초기 단계에서 매우 간단하게 계획에 대해 논의를 했더라면, 몇 가지 신중한 질문과 명확한 답변만으로 이런 상황에서 모든 데이터 사이언티스트가 원하는 목표, 즉 빠른 성과에 도달할 수 있었을 것입니다.

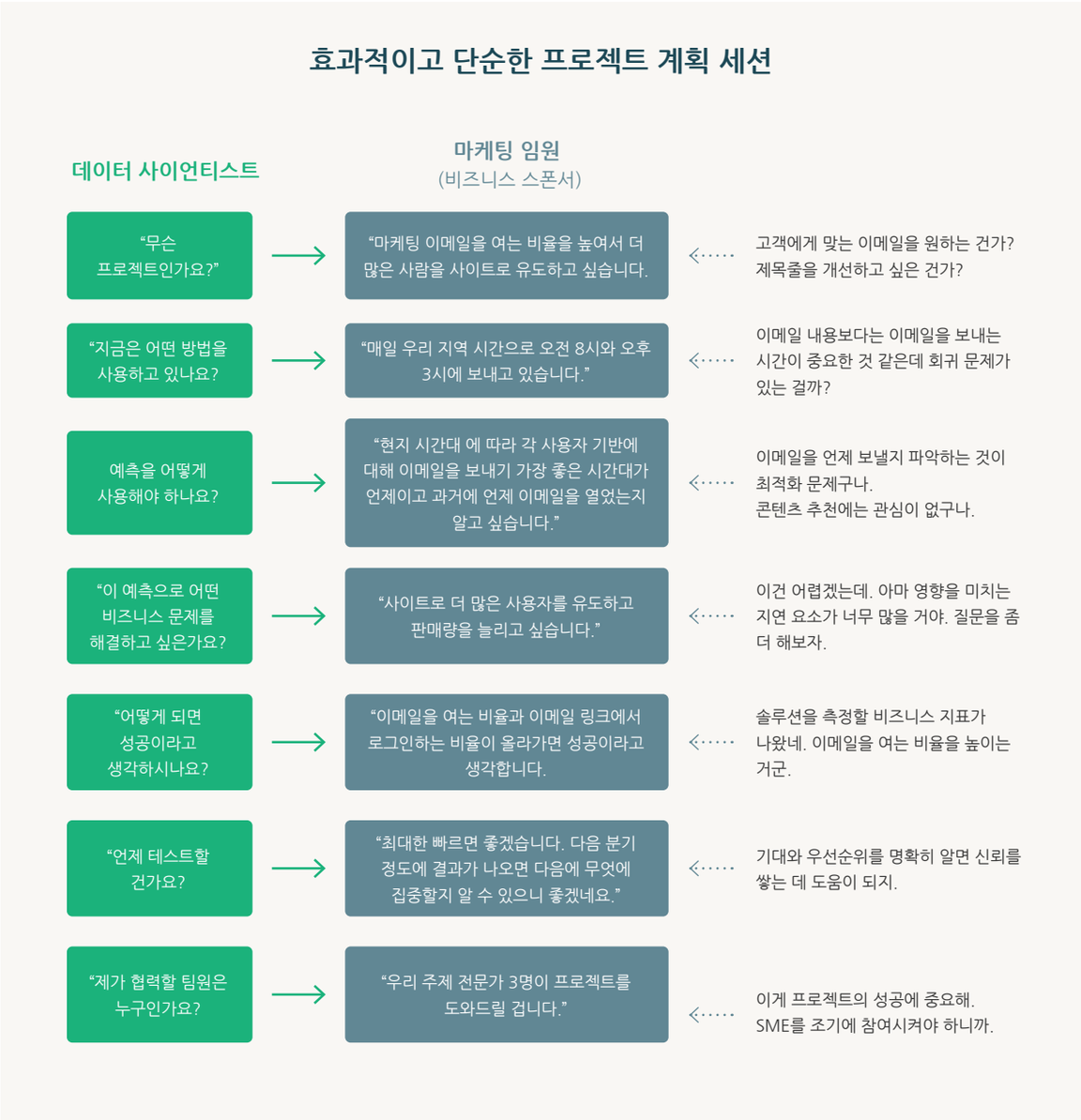


그림 1.4
간단한 계획 논의를 통해 내부 고객(이 경우, 이메일 오픈율을 높이고 싶은 마케팅 임원)이 실제로 솔루션에 요구하는 목표가 무엇인지 근본적으로 확인

그림 1.4를 보면 알 수 있듯이, 문제는 처음에 세운 가정이 아닙니다. 이메일 내용, 제목과의 관련성이나 이메일에 포함할 항목에 관한 논의는 전혀 없습니다. 단순히 고객이 위치한 시간대를 알아내 각 고객이 과거에 이메일을 오픈한 시간을 현지 시각으로 분석하는 간단한 분석 쿼리일 뿐입니다. 잠시만 시간을 들여 계획을 수립하고 사용 사례를 온전히 이해하면 몇 주(심하면 몇 달)나 낭비되는 수고, 시간과 비용을 절약할 수 있습니다.

무엇을 개발하고, 왜 개발해야 하는지에 집중하면 데이터 사이언스 팀과 사업부 모두 논의를 통해 보다 유익한 결실을 얻을 수 있습니다. 개발 방식에 초점을 맞춘 대화를 삼가면 데이터 사이언스 팀원이 당면한 문제에 집중할 수 있습니다. 개발 시점을 염두에 두지 않으면 프로젝트 요구 사항 위주로 논의를 진행할 수 있습니다.

이 단계에서 구현에 대한 세부적 논의를 피하면 데이터 사이언스 팀은 문제에 집중할 수 있는데, 이 점이 중요합니다. 대규모 팀과의 논의에서 소수만 아는 알고리즘과 솔루션 설계에 대한 세세한 내용을 언급하지 않으면 사업부 구성원들이 논의에 계속 집중할 수 있습니다. 말하자면 반죽에 들어가는 달걀 수, 달걀의 색, 심지어 달걀을 낳은 닭의 품종 같은 것은 상관없고, 중요한 것은 케이크를 완성시켜 먹는 것이라는 말입니다.

범위 설정 및 조사

범위 설정 및 조사 단계에서는 내부 고객(사업부)이 프로젝트에 대해 가지고 있는 가장 큰 질문 두 가지를 해결하는 데 집중해야 합니다.

- 내 문제를 해결해줄 것인가?
- 얼마나 오래 걸릴 것인가?

이와 유사한 다른 시나리오를 살펴보면서 이 단계의 ML 프로젝트 개발이 어떻게 틀어지는지 알아보겠습니다. 이 예시에서는 하나의 회사에 두 개의 데이터 사이언스 팀이 운영되고 있습니다(그림 1.5의 팀 A, 그림 1.6의 팀 B). 각 팀은 서로 경쟁하면서 회사 청구 시스템에서 발생하는 사기 사건을 신고하는 솔루션을 개발하고 있습니다.

팀 A의 조사 및 범위 설정 프로세스는 그림 1.5와 같습니다.

팀 A는 대부분 신입 데이터 사이언티스트로 구성되어 있고, 팀원 모두 학계에서 오래 머무르지 않고 사회 생활을 시작했습니다. 이들은 프로젝트 상세 정보와 자기 팀에서 해야 하는 일을 알아낸 다음 곧장 블로그 게시물을 검색하기 시작합니다. 이 팀은 인터넷에서 "결제 사기 탐지"와 "사기 알고리즘"을 검색한 결과, 컨설팅 기업에서 올린 게시물 수백 개, 자기들과 비슷한 수준의(실제로 모델을 프로덕션까지 진행해본 적이 없을 가능성이 큼) 초보 데이터 사이언티스트가 올린 극히 간략한 블로그 게시물 몇 개, 그리고 몇몇 아주 기초적인 오픈 소스 데이터 예시를 입수했습니다.

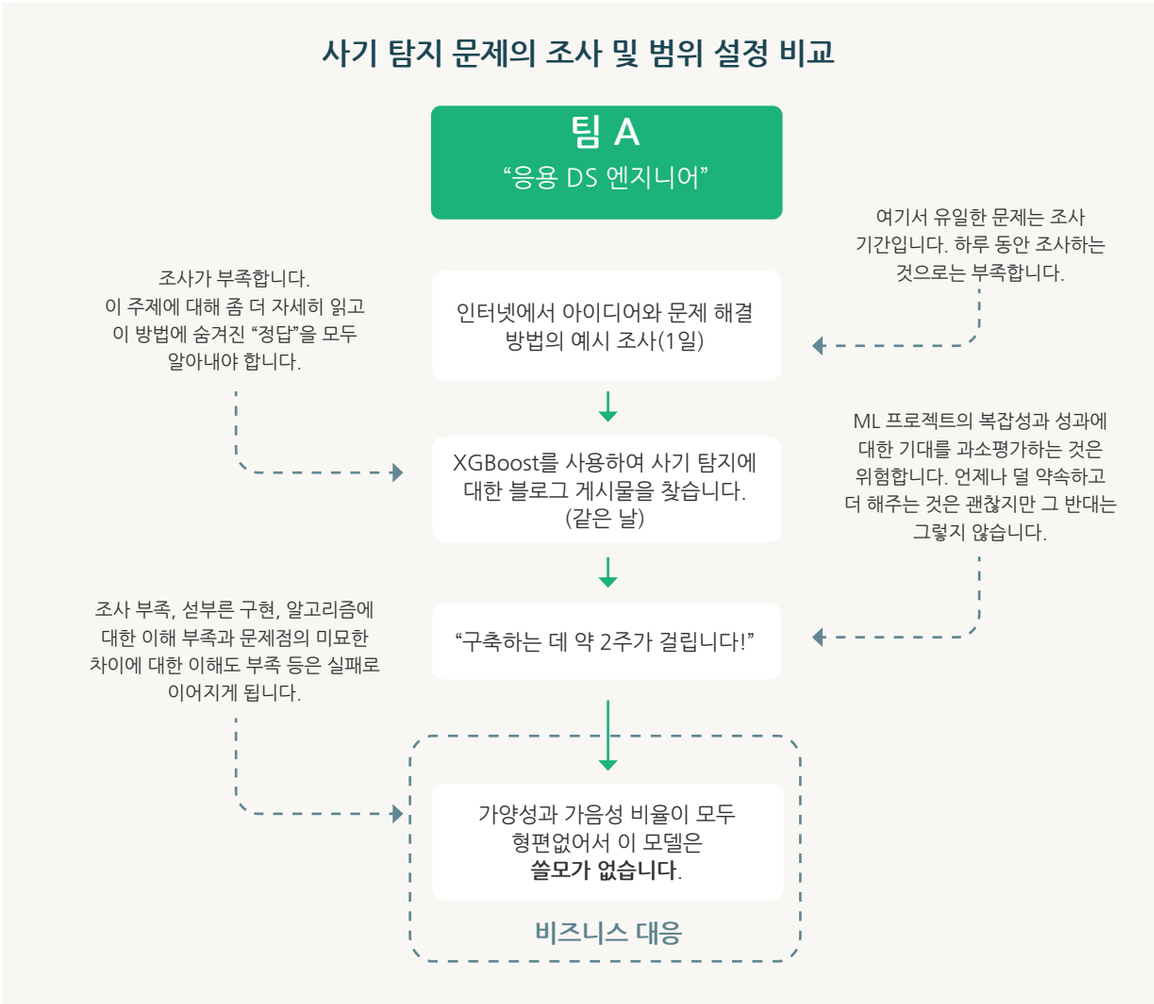


그림 1.5
열정은 있으나 경험이 부족한 데이터 사이언티스트로 구성된 신입팀의 사기 탐지 문제에 대한 조사 및 범위 설정

그림 1.6에 나와 있는 팀 B의 조사와 범위 설정 방식은 완전히 반대입니다.

팀 B는 박사 학위를 받은 연구진으로 구성되었습니다. 이들은 학계에서 적용하던 조사와 아이디어 개진 방식에 익숙해서 가장 먼저 사기 모델링 주제에 대한 논문을 찾기 시작합니다. 며칠을 투자해 학술지와 논문을 탐독하고 나니, 이제 사기 행위 탐지에 관해서라면 최첨단 연구까지 망라하는 각종 이론으로 무장하게 되었습니다.

이 두 팀에 솔루션을 내놓으려면 어느 정도로 수고가 들겠느냐고 물으면 서로 완전히 다른 답을 내놓을 것입니다. 팀 A라면 XGBoost 바이너리 분류 모델을 구축하는 데 약 2주가 걸릴 것이라고 밝힐 가능성이 큽니다(팀원들이 찾아낸 블로그 게시물을 근거로, 자신들이 이미 코드를 확보했다고 잘못 생각하고 있음).

팀 B의 입장은 완전히 다를 것입니다. 이 팀의 경우, 높은 평가를 받는 백서에서 찾은 신형 딥러닝 구조를 구현, 훈련, 평가까지 마치는 데 몇 달은 걸릴 것으로 예상할 텐데, 이 백서는 이 사용 사례에서 필연적으로 구현된 어느 알고리즘보다 정확도 면에서 훨씬 우월합니다.

두 팀은 조사와 범위 설정 방법이 양극단에 있지만 서로 완전히 다른 이유로 프로젝트 실패라는 같은 결말을 맞게 될 것입니다. 팀 A의 프로젝트가 실패하는 이유는 문제에 대한 솔루션이 자신들이 조사한 블로그 게시물에 나와 있는 예시보다 훨씬 복잡하기 때문입니다(클래스 불균형 문제만 해도 블로그 게시물이라는 좁은 공간에 효과적으로 기술하기는 어려운 주제입니다). 팀 B는 솔루션 자체는 매우 정확하겠지만, 회사 입장에서 이렇게 위험한 솔루션을 1차 사기 탐지 서비스 솔루션으로 구축하라고 리소스를 배정해 줄 리가 없습니다(단, 2.0 버전을 구현할 때는 훌륭한 후보가 될 수 있겠습니다).

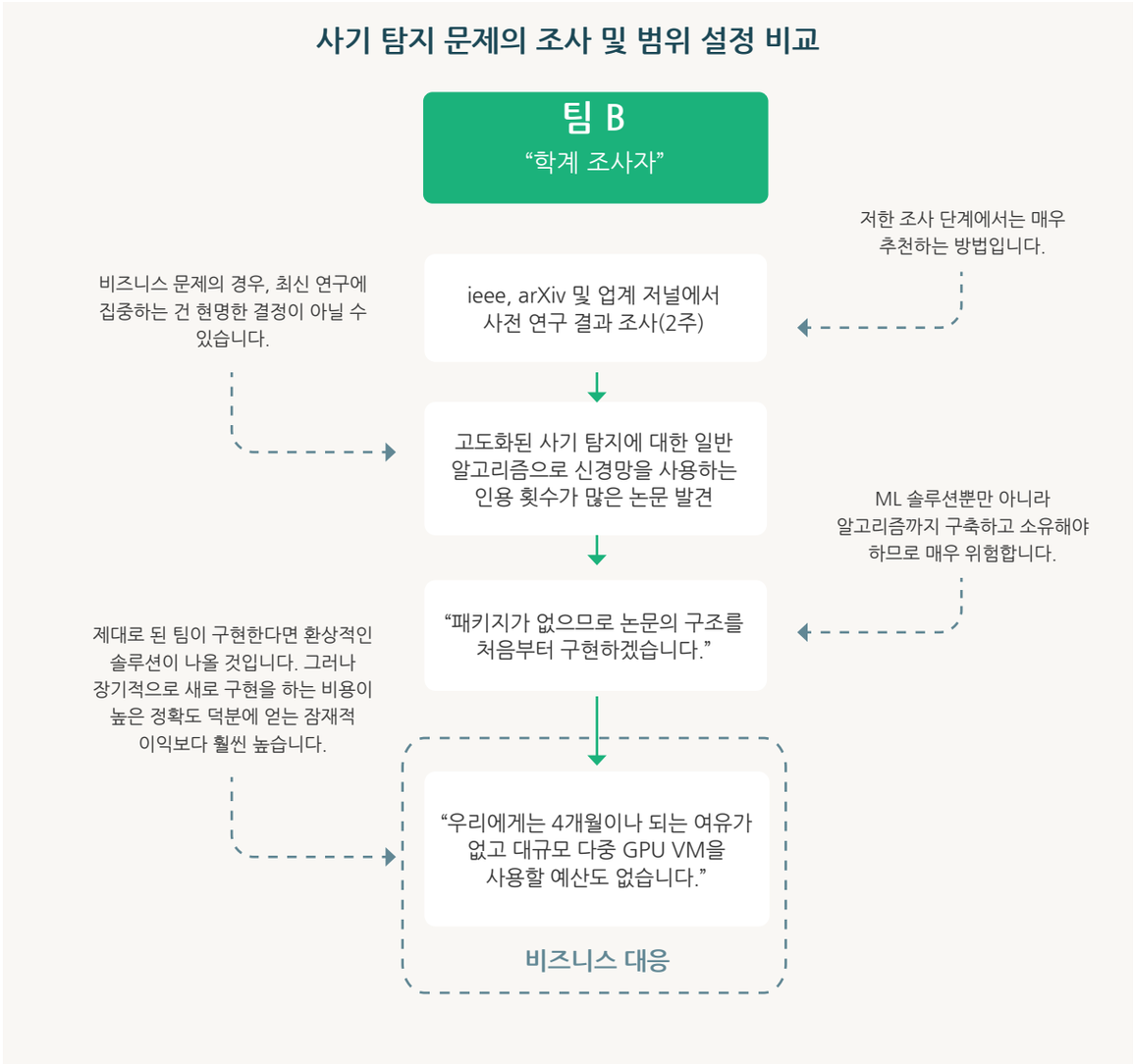


그림 1.6
연구에 집중한 연구자 그룹의 사기 탐지 문제에 대한 조사 및 범위 설정

ML의 프로젝트 범위 설정은 매우 까다롭습니다. 아무리 경험이 풍부한 ML 베테랑이라고 해도 프로젝트의 예상 소요 시간, 가장 성공률이 높은 접근 방식, 필요한 리소스의 양을 추측하는 일은 대체로 무익하고 불만족스러운 과정입니다. 엉뚱한 예측을 하면 발생할 위험이 상당히 큰 것이 사실이지만, 예측이 크게 빗나갈 확률을 최소한으로 줄이는 데 도움이 되도록 적절하게 범위를 설정하고 솔루션을 조사하도록 구조를 잡을 방법이 몇 가지 있습니다.

대부분 기업은 앞서 언급한 두 가지 양극단의 시나리오에 있는 팀원 유형이 섞여 있습니다. 즉 지식 발달과 연구 결과를 알고리즘으로 실현하여 업계 내부로부터 앞으로 새로운 연구 결과를 발견할 길을 닦는 것만이 유일한 목표인 학자 타입이 있고, ML을 비즈니스 문제점을 해결할 도구로 사용하고자 할 뿐인 "ML 응용 분야" 엔지니어 타입이 있습니다. 이와 같은 ML 작업에 관한 여러 원칙의 두 가지 측면을 골고루 받아들여 균형을 이루고 프로젝트 조사와 범위 설정 단계를 거치면서 타협점을 찾는 것이 매우 중요합니다. 여기서 절충안이란 프로젝트를 실제로 프로덕션까지 추진하는 최선의 경로를 찾는 것입니다.

실험

실험 단계에서 프로젝트 실패를 초래하는 가장 큰 원인은 실험이 지나치게 오래 걸리거나 (지나치게 많은 것을 테스트하거나 방법을 세세하게 조정하는 데 너무 오랜 시간을 투자) 프로토타입의 품질이 너무 형편없어서 회사에서 포기하고 다른 것을 개발하기로 결정하는 것입니다.

섹션 1.2.2의 예시는 이 두 가지 방식이 리테일 매장에 진열된 제품을 탐지하는 이미지 분류 도구를 구축하는 기업에 어떻게 적용되는지 살펴봅니다. 두 집단이 취하는 실험 경로(두 가지 양극단의 실험 방식)는 그림 1.7과 1.8에 나와 있습니다.

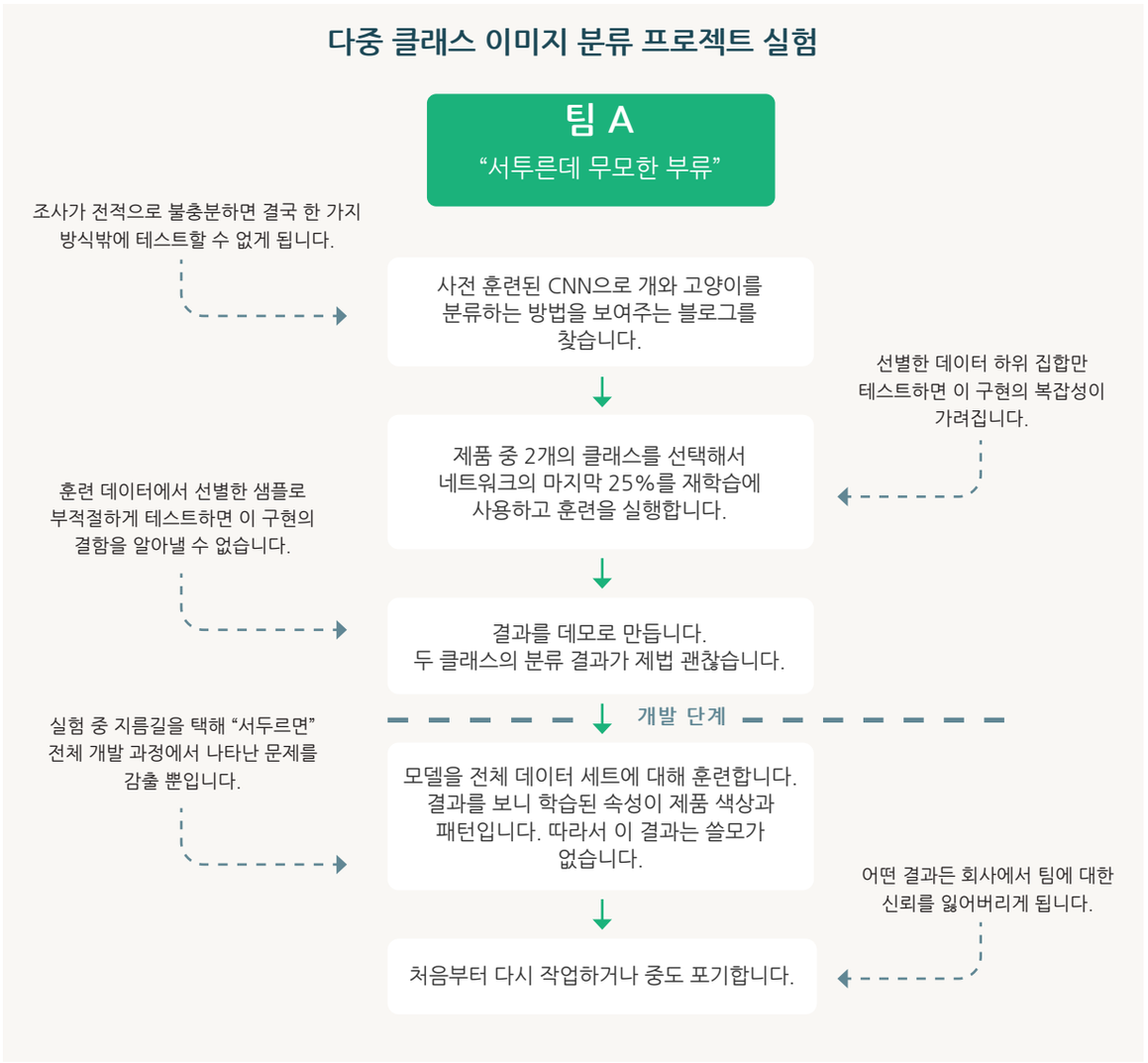


그림 1.7
경험이 부족한 데이터 사이언티스트로 구성된 팀이 서둘러 진행한 실험 단계

그림 1.7의 팀 A는 경험이 없다시피 한 데이터 사이언스 팀을 매우 과장해서 표현하였고, 수박 겉핥기식의 연구만 진행합니다. 이미지 분류 작업에 대해 발견한 예시 블로그 게시물 하나만 사용하고, 예시 코드를 복사하여 블로그에 인용된 사전 훈련된 TensorFlow-Keras 모델만을 사용합니다. 이 모델을 (수천 개 중) 불과 두 개의 제품만으로 구성된 이미지 수백 개만으로 다시 훈련해서 두 가지 클래스에서 얻은 이미지에 대해 상당히 우수한 분류 결과를 제출합니다.

그러나 철저한 조사를 하지 않았기 때문에 자신이 선택한 모델의 한계를 이해하지 못했습니다. 서둘러 데모를 만들어 얼마나 이미지를 잘 분류할 수 있는지 보여주기 위해 오직 두 개의 클래스로만 구성된 지나치게 단순한 테스트를 선택했습니다. 선별된 결과와 그 방법에 대한 평가가 매우 형편없었기 때문에 이 프로젝트는 개발 프로세스 초기에 실패하거나(경영진 누군가가 진행 상황을 확인할 경우), 나중에 프로덕션 일정을 조정하기 전 마지막 납품 단계에서 실패할 가능성이 큽니다(사업부의 내부 고객까지 이 방식이 얼마나 형편없는 솔루션인지 알게 된 경우). 어느 쪽이든 이렇게 서두른 데다 게으르게 테스트한다면 프로젝트는 중도 포기하거나 취소되는 결말을 맞을 수밖에 없습니다.

이 문제에 대한 팀 B의 실험 방식은 그림 1.8과 같습니다.

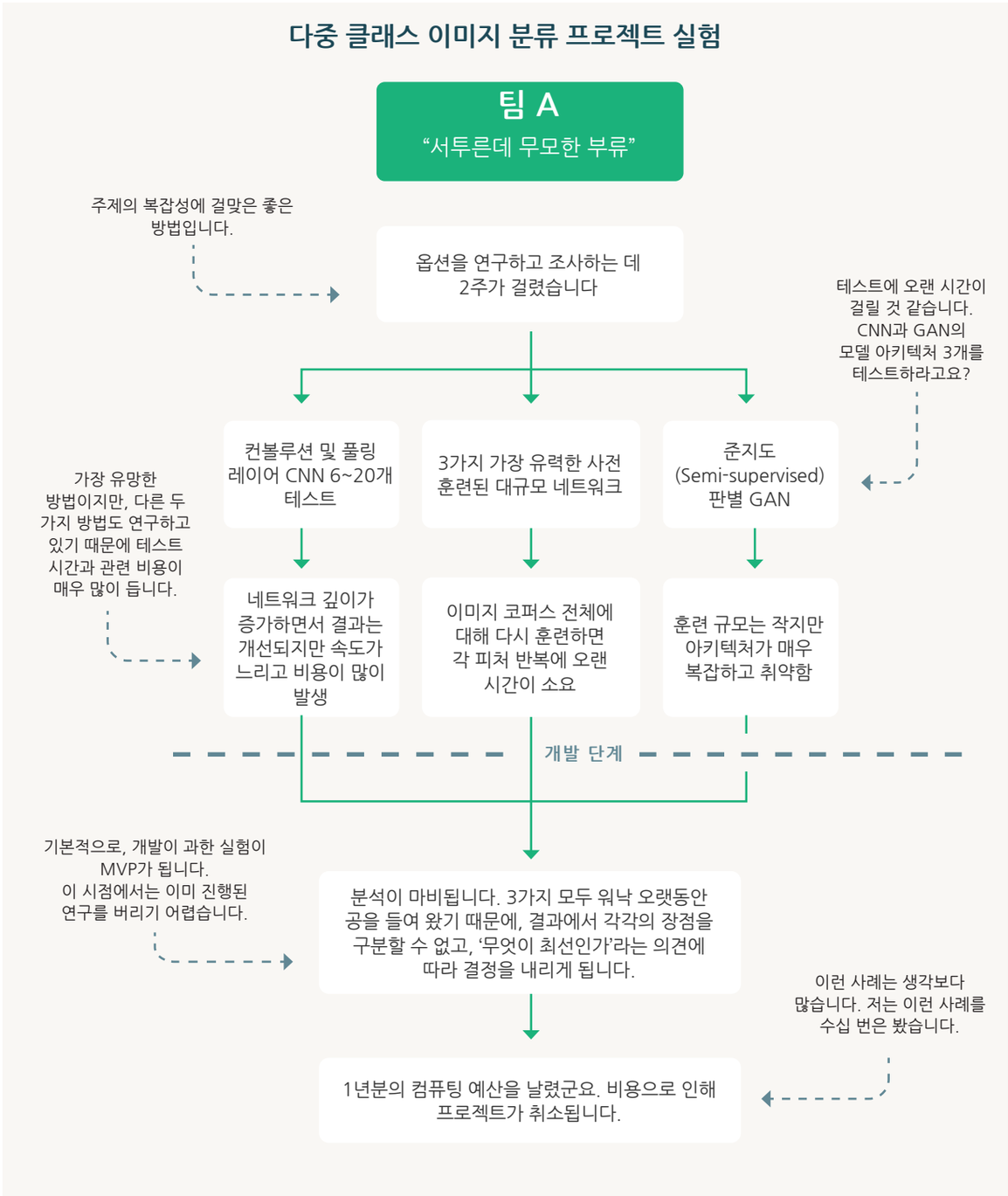


그림 1.8
과도하게 철저한 실험 단계로 인해 프로젝트에 3가지 MVP를 구축

그림 1.8에 나와 있는 팀 B는 팀 A와는 완전히 반대입니다. 팀 B는 "순수 연구자"의 표본입니다. 이들은 비록 회사에서 일하고 있지만 여전히 박사 논문을 쓰는 것처럼 행동합니다. 팀 B는 몇 주간 최신 논문을 조사하고, 학술지를 읽으며, 다양한 컨볼루션 신경망(CNN) 방법과 관련된 이론을 이해하는 것으로 문제를 해결하고자 합니다. 그 결과 세 가지 광범위한 잠재적 솔루션을 얻었습니다. 각 솔루션은 여러 번 테스트를 실행하고 훈련 이미지 데이터 세트의 전체 컬렉션에 대해 평가가 필요합니다.

이 경우, 연구의 깊이가 부족해서 실패한 것이 아닙니다. 이 사용 사례에는 이런 연구가 적절했습니다. 문제는 너무 많은 것을 하려고 했다는 것입니다. 맞춤 개발된 CNN의 구조와 깊이가 다르기 때문에 (수백 번까지는 아니라도) 수십 번의 반복을 거쳐야 해결하고자 하는 사용 사례에 "적절한" 모델을 얻을 수 있습니다. 이는 평가가 아니라 프로젝트 개발 단계에서 예상했어야 하는 작업입니다. 맞춤형 CNN에 대해 신속히 판별하는 대신, 사전 훈련된 대규모 CNN 3개의 전이 학습을 테스트하고 생성적 적대 신경망(Generative Adversarial Network, GAN)을 구축하여 분류가 필요한 방대한 클래스 코퍼스에 준지도 학습을 적용하기로 했습니다.

간단히 말해 팀 B는 실험 단계에서 지나치게 많은 작업을 했습니다. 자신의 솔루션을 데모로 보여줘야 하는 단계에서는 정작 아무런 결정도 내릴 수 없이 마비되고 엄청난 클라우드 서비스 GPU VM 청구서만이 남았습니다. 최적의 솔루션에 대한 실질적 결론이 없고 프로젝트에 이미 엄청나게 막대한 금액을 지출했기 때문에 프로젝트 자체가 폐기될 가능성이 매우 큽니다.

실험 단계가 프로젝트 실패의 결정적 원인이 될 수는 없지만 이 단계를 잘못하면 원래는 좋은 성과를 냈을 프로젝트를 중단시키거나 취소하게 될 수 있습니다. 가상의 팀에서 사용한 이런 연구 방식은 극단적 예시입니다. 두 가지 방법 모두 적절하지 않으므로, 두 가지가 적절히 균형 잡힌 방식이 가장 좋습니다.

개발

ML 프로젝트의 개발 과정의 비효율이 프로젝트 취소에 직접적인 영향을 미치는 결정적 요인은 아니지만 그 자체만으로도 여러 가지 방식으로 프로젝트를 완전히 중단시킬 만한 양상이 일어나게 됩니다. 일반적으로는 다른 주요 원인보다 가시적으로 드러나지는 않습니다. 그러나 코드 베이스가 취약하고 설계가 미비한 데다 개발 과정까지 부실하면 프로젝트를 진행하기가 더욱 어렵고, 프로덕션에서 장애가 일어나기 쉬우며, 시간이 갈수록 개선하기는 더욱 힘들어집니다.

예를 들어, 모델링 솔루션을 개발하면서 자주 발생하는 간단한 수정 과정을 살펴보겠습니다. 바로 피처 엔지니어링 변경입니다. 그림 1.9에서는 두 명의 데이터 사이언티스트가 모놀리식 코드 베이스를 여러 가지로 변경하려고 합니다. 이 개발 패러다임에서는 모든 작업의 로직을 스크립팅된 변수 선언과 함수를 통해 하나의 노트북에 작성합니다.

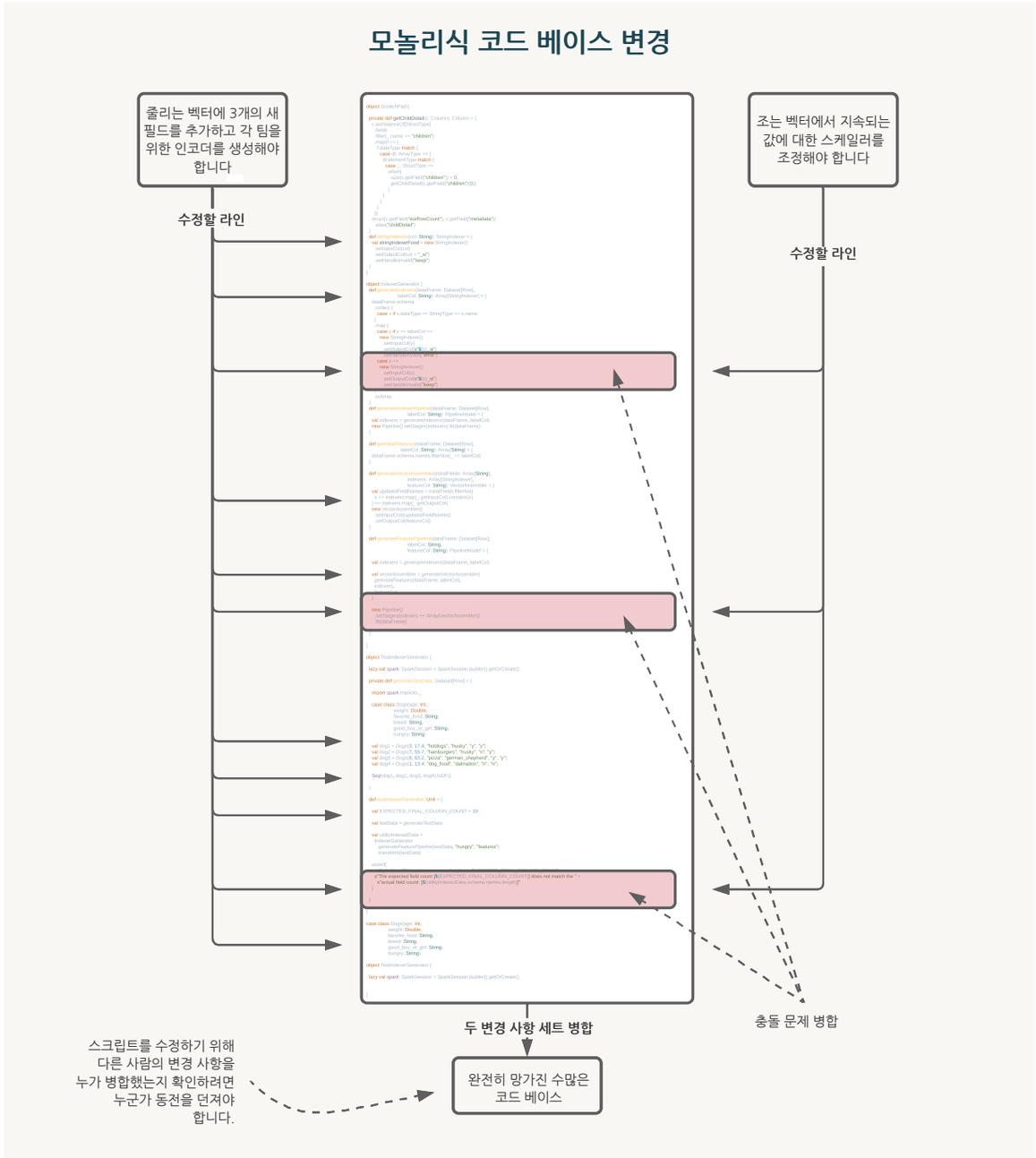


그림 1.9
ML 프로젝트 작업의 모놀리식 코드 베이스(스크립트) 편집

줄리는 모놀리식 코드 기반으로 작업하며, 검색과 스크롤링을 많이 거쳐야 할 가능성이 큽니다. 피처 벡터가 정의된 위치를 일일이 찾아 새 필드를 컬렉션에 추가하는 것입니다. 줄리의 인코딩 작업은 수정해야 하고, 올바른 위치에서 스크립트 전체에 걸쳐 수행해야 하기도 합니다. 이 방식은 상당히 복잡한 ML 코드 베이스(스크립팅 패러다임으로 개발 시 피처 엔지니어링과 모델링을 위해 작성한 코드가 수천 줄에 달하는 경우)에서는 감당하기 어려운 작업일 뿐만 아니라, 누락, 오타, 기타 전사 오류의 형태로 번거로운 오류가 발생하기 쉽습니다.

한편, 조는 편집할 내용은 훨씬 적지만 여전히 긴 코드 베이스를 검색하고 하드코딩된 값을 올바르게 수정해야 합니다.

모놀리식 방식의 진짜 문제는 각 변경 사항을 하나의 스크립트 사본에 적용하려고 할 때 발생합니다. 줄리와 조는 각자의 작업이 서로 영향을 받기 때문에 두 사람 모두 자신의 코드를 업데이트하고 프로젝트의 “마스터”역할을 할 사본을 하나 선택하여 상대방 작업물의 변경 사항을 복사해야 합니다. 이는 시간이 오래 걸리고 고된 과정이며, 귀중한 개발 시간을 낭비하고 제대로 완성하려면 엄청난 양의 디버깅을 거쳐야 할 가능성이 큽니다.

그림 1.10은 ML 프로젝트 코드 베이스를 유지 관리하는 다른 방법을 나타낸 것입니다. 여기에서는 모듈식 코드 아키텍처를 사용하여 그림 1.9와 같은 긴 스크립트 내의 긴밀한 결합을 분리합니다.

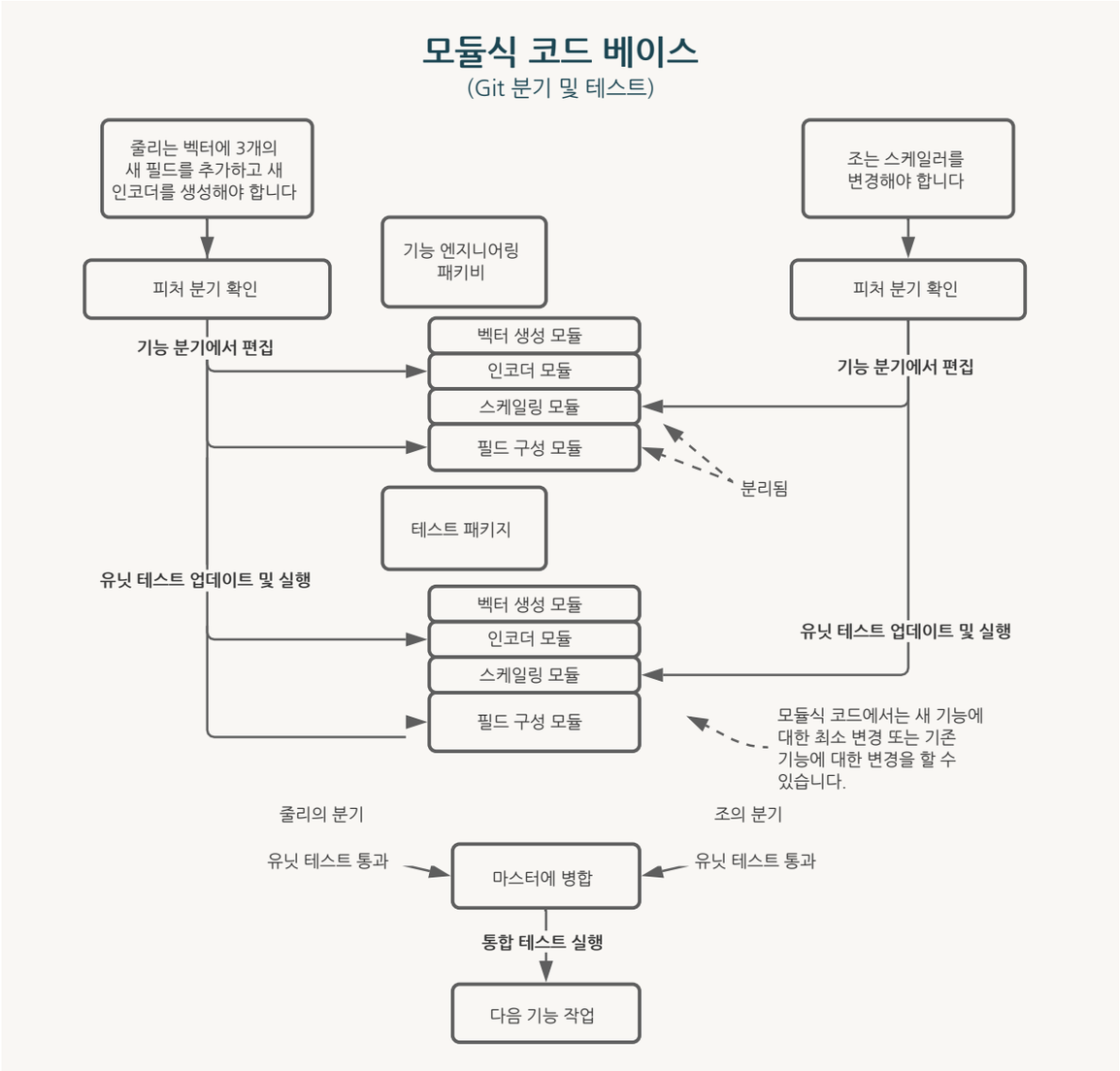


그림 1.10
모듈식 ML 코드 베이스를 업데이트하여 재작업을 방지하고 충돌 병합

그림 1.10은 통합 개발 환경(IDE)에서 작성된 모듈식 코드 베이스를 보여줍니다. 그림 1.9에서 두 데이터 사이언티스트가 적용한 변경 사항은 동일하지만(줄리는 피처 벡터에 몇 가지 필드를 추가하고 새 필드에 대한 인코딩을 업데이트하는 반면, 조는 피처 벡터에 사용한 스케일러를 업데이트), 이러한 변경 사항이 서로 함께 작용하도록 설정하는 데 들인 시간과 수고는 크게 다릅니다.

Git에 완전 모듈식 코드 베이스가 등록되어 있기 때문에 두 사람은 마스터에서 피처 분기를 확인하고, 자신의 피처에 포함된 모듈을 약간 수정해서 새 테스트를 작성(필요한 경우)하고 실행하여 풀 요청을 제출하면 됩니다. 이 작업이 완료되면 구성 기반 코드를 사용하는 데다 각 모듈 클래스의 메서드를 작업 구성을 통해 프로젝트의 데이터에 실행할 수 있으므로 각 피처 분기는 서로 영향을 미치지 않고 원래의 설계대로 작동합니다. 두 변경 사항의 릴리스 분기를 잘라서 하나의 빌드에 넣고, 전체 통합 테스트를 실행한 다음, 완전 통합 테스트를 진행해서 마스터에 안전하게 병합할 수 있습니다. 이때 작업에 오류가 생길 염려는 하지 않아도 됩니다.

대부분 데이터 사이언티스트는 이런 방식(모듈식 설계, IDE 작성)으로 코드를 작성하는 것으로 시작합니다. 우리 엔지니어는 인터랙티브 노트북을 통해 배웠고 (저를 포함한) 대부분 엔지니어는 여전히 프로토타이핑 아이디어, 실험, 작업 분석에 노트북을 상당히 자주 활용합니다. 그러나 이런 대안적인 ML 코드 작성 방법(프로토타입 스크립트와 함수를 개체 지향적 또는 기능적 프로그래밍 패러다임으로 포팅)을 도입하면 프로젝트에서 새로운 기능을 동시에 개발하는 여러 사용자를 지원하면서도, 각각의 새로운 아이디어와 약간의 기능을 완전하게 테스트하여 추적하기 어려운 버그를 제거할 수 있습니다. 이렇게 오래전에 검증이 끝난 소프트웨어 개발 패러다임을 기반으로 ML 코드 프레임워크를 만드는 데는 시간과 수고 면에서 오버헤드가 많이 듭니다. 하지만 일단 코드 베이스를 두 번째로 변경해야 하는 시점이 오기만 하면 그만한 오버헤드를 들일 가치가 있었음이 전적으로 입증됩니다.

배포

ML 프로젝트 작업에서 가장 혼란스럽고 복잡한 문제는 매우 정확한 모델을 구축한 이후 오랜 시간이 지난 시점에야 생겨난다고 할 수 있습니다. 모델을 개발해서 예측을 실제로 사용할 수 있는 수준까지 제공하기에 이르는 과정도 그만큼 어렵고, 예측 요구 사항을 해결하기 위한 모델이 여러 개이기 때문에 구현도 그만큼 다양할 수 있습니다.

이 섹션에서는 패스트푸드 산업에 분석 서비스를 제공하는 회사를 예로 들어보겠습니다. 이 회사는 수년간 지역 수준 그룹별로 재고 관리에 대한 예측을 상당히 성공적으로 제공하였고, 주간 예상 고객 수에 따른 일일 수요를 대규모 일괄 예측하여 매주 예측 데이터를 일괄 추출 형식으로 제출합니다.

데이터 사이언스 팀은 지금까지는 사실상 그림 1.11과 같은 ML 아키텍처에 익숙해져 있었습니다.

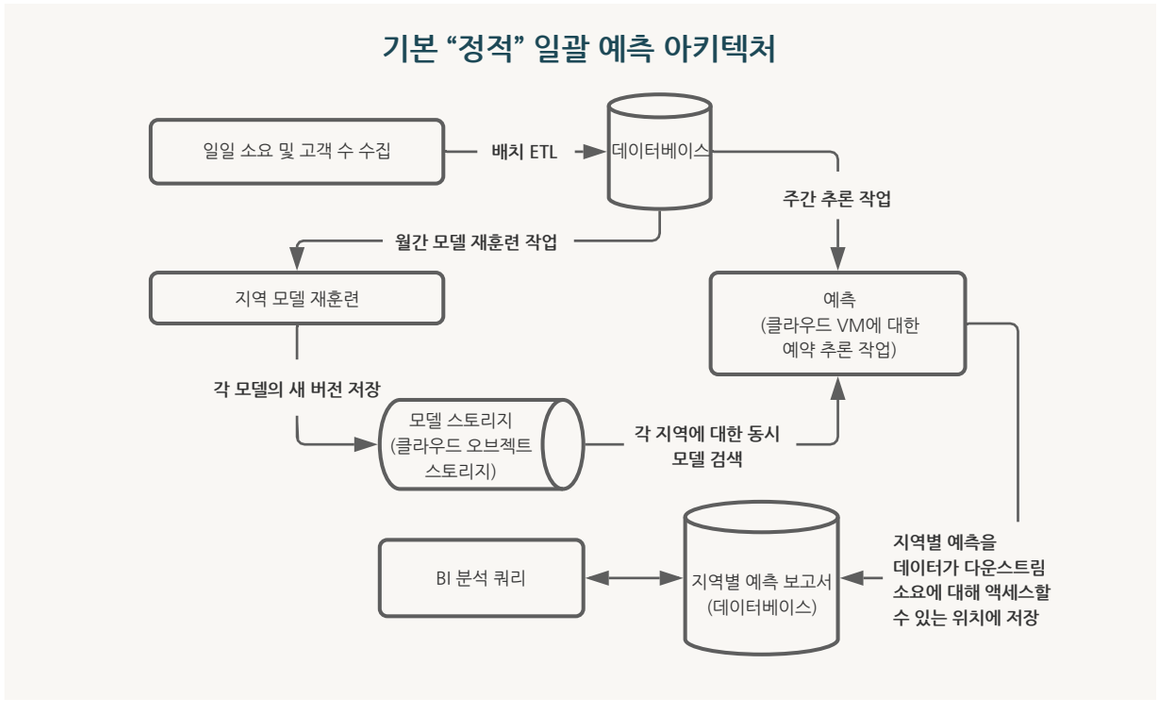


그림 1.11
비교적 단순한 예약 배치 내부 예측 아키텍처

예약된 일괄 예측(그림 1.11 참조)을 제공하는 데 사용하는 비교적 표준화된 아키텍처는 추론 결과를 내부 분석팀에 노출하는 데만 초점을 맞추어 그다지 복잡하지 않고, 분석팀에서도 매우 친숙하게 여기는 패러다임입니다. 설계는 예약된 동기식 방식을 따르고 이후의 재훈련과 추론 사이에는 상당한 시간이 걸리기 때문에 기술의 대체적인 복잡도는 그다지 높지 않습니다(이는 장점입니다. 아래의 참고 사항을 참조하세요).

단순 아키텍처에 대한 간단한 참고 사항

ML 분야에서 아키텍처를 구현할 때는 언제나 최대한 단순한 설계를 사용하려고 노력해야 합니다. 프로젝트의 추론 주기가 1주일일 경우, (실시간 스트리밍이 아니라) 배치 프로세스를 사용합니다. 데이터 용량이 MB 단위라면 데이터베이스와 (25노드 Spark 클러스터가 아닌) 단순한 VM을 사용합니다. 훈련 런타임을 분 단위로 측정할 경우, (GPU가 아니라) CPU를 사용합니다.

아무 목적 없이 복잡한 아키텍처, 플랫폼, 기술을 사용한다면 반드시 후회하게 됩니다. 이미 복잡한 솔루션에 불필요한 복잡성이 추가되기 때문입니다. 새로운 복잡성이 추가될 때마다 (대개는 매우 복잡한 방식으로) 무언가 장애가 일어날 가능성이 커집니다. 프로젝트의 즉각적인 비즈니스 요구 사항을 해결할 정도로 기술, 스택, 아키텍처를 단순하게 유지하는 것은 사업부에 일관적이고 안정적이면서도 효과적인 솔루션을 제공하기 위해 지켜야 할 모범 사례입니다.

시간이 지남에 따라 이 회사에서 예측 모델링의 장점을 깨닫게 되었고 사업부에서는 새로운 사업을 시작하면서 데이터 사이언스 팀에 새로운 예측 시스템을 구축해달라고 요청합니다. 이 새로운 서비스는 매장별로 재고 예측이 필요하고, 하루 내내 실시간에 가까운 속도로 대응형 예측을 제공해야 한다는 요구 사항이 수반됩니다. 데이터 사이언스 팀은 완전히 다른 모델 구축만으로는 이 사용 사례를 해결할 수 없다는 것을 깨닫고 대부분의 시간과 에너지를 프로젝트의 ML 개발에 투자했습니다. 이 솔루션의 서비스 구성 요소는 애플리케이션을 통해 매장 소유주에게 각각 데이터를 제공할 때 REST API에 의존해야 할 뿐만 아니라, 매장별 예측을 꽤 잦은 빈도로, 하루에도 여러 번 업데이트해야 한다는 사실을 깨닫지 못한 것입니다.

비즈니스 요구 사항을 지원하는 아키텍처를 개발한 뒤 (프로젝트를 시작하고 나서 몇 개월 후, 프로젝트의 모델링 부분이 끝나고도 한참 후에) 몇몇 Java 소프트웨어 엔지니어의 도움을 받아 구축을 시작했습니다. 실제로 사용하고 나서 첫 주가 지나서야 클라우드 내에서 이 아키텍처를 개발하는 비용이 이 서비스를 통해 얻는 수익의 몇 배에 달한다는 것을 깨달았습니다. 비즈니스 요구 사항을 해결하는 데 필요한 새로운 아키텍처는 그림 1.12와 같습니다.

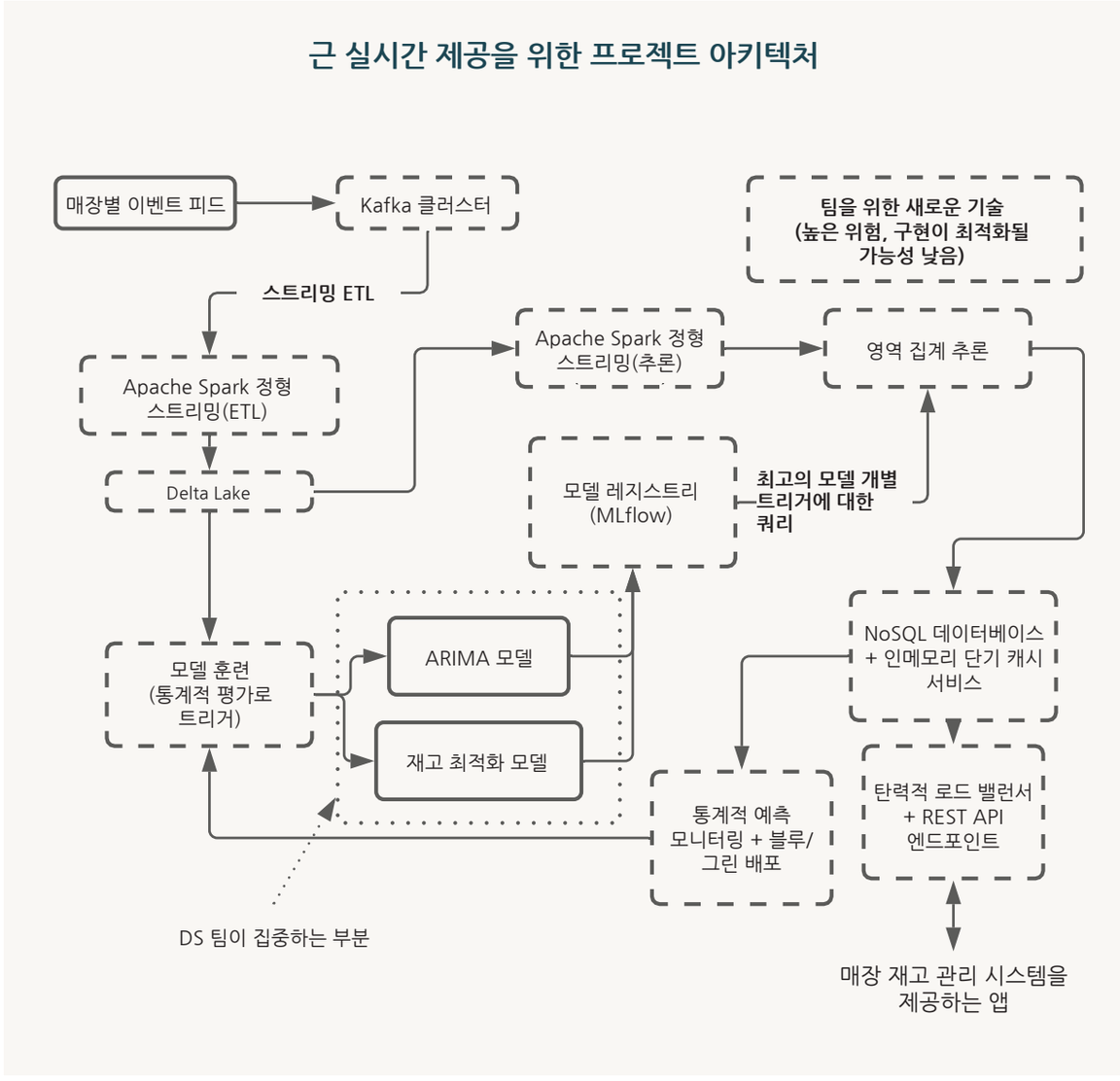


그림 1.12
프로젝트의 비즈니스 요구 사항을 해결하는 데 필요한, 더욱 복잡한 모의 실시간 아키텍처

얼마 지나지 않아 프로젝트는 취소되고, 비용을 낮추기 위해 아키텍처와 모델링을 완전히 재설계하는 작업을 외부로 돌리게 됩니다.

이런 사례는 ML을 구현하여 새롭고 흥미로운 문제를 해결하고자 하는 기업들에서 흔히 찾아볼 수 있습니다. 배포와 서비스에 집중하지 않으면 개발이 불가능해서가 아니라, 솔루션 엔지니어링 비용이 프로젝트 예산을 훨씬 초과하여 프로젝트가 성공하지 못할 수 있기 때문입니다.

실행, 관리, 모니터링에 들어가는 비용이 얼마나 될지 비판적으로 생각하며 배포와 서비스를 바라보는 것은 좋은 습관입니다. 이런 습관은 솔루션 개발뿐만 아니라 프로젝트에서 구현하려는 일반적인 아이디어의 실현가능성을 알리는 데 도움이 됩니다. 사실 ML 프로젝트를 접은 원인 중에서도 좋은 솔루션인데 실행 비용이 너무 많이 든다는 이유만으로 포기할 수밖에 없는 것이 가장 안타까운 사례입니다.

그림 1.13은 예측 결과를 제공하는 것과 관련된 (극히 일부의) 몇 가지 요소를 보여줍니다. 각 섹션에 대한 상대적 비용을 잘 확인해보세요. 프로젝트 초기에 이 정보를 분석하면 앞으로 이 프로젝트에 얼마나 비용이 들어갈지 예상할 수 있으므로, 실행 비용이 회사 입장에서 마음에 들지 아닐지 실제로 회사에 개발 비용이 발생하기 전에 미리 측정할 수 있습니다.

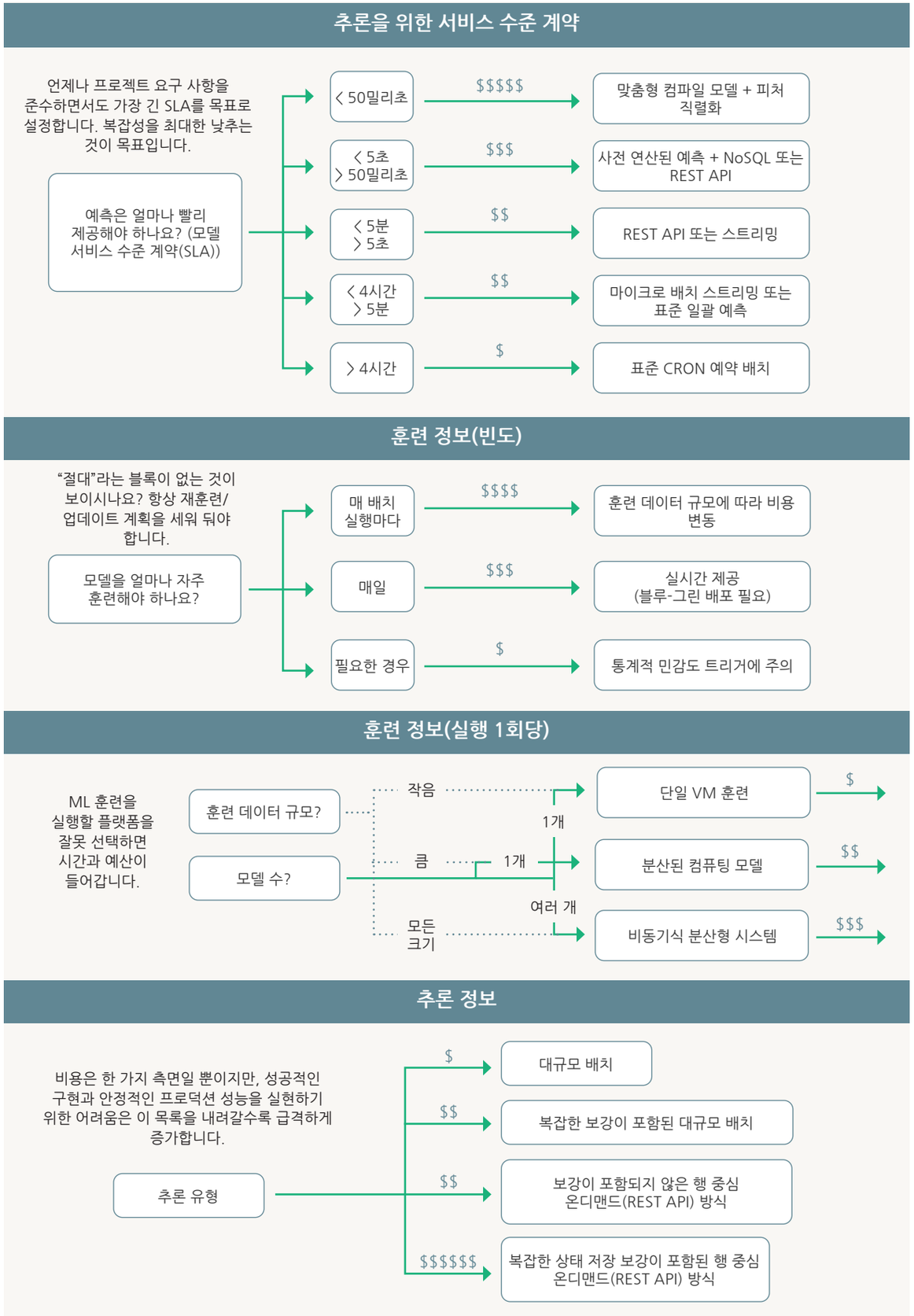


그림 1.13
각 ML 제공 패러다임에 대한 하드웨어 (플랫폼) 비용과 인적 자본(개발 시점, 관리의 어려움) 측면에서 배포 고려 사항

Manning Publications에서 출간한 “Machine Learning Engineering in Action”에서는 그림 1.13(및 ML 비용에 영향을 미치는 기타 요소)의 각 섹션, ML 제공 시 고려 사항, 플랫폼 선택, 구축과 구매 시 비교, 모델링과 관련된 데이터 볼륨 비용에 대해 다룹니다. 이것은 ML 엔지니어링에서 흥미로운 부분은 아니고, (보통, 너무 늦기 전까지는) 자주 생각하는 부분도 아니지만 프로젝트 취소를 초래하는 가장 확실한 길이 될 수 있으므로 상당히 신중하게 고려해야 합니다.

평가

ML 프로젝트를 취소하거나 중도 포기하는 원인 중에서도 ‘예산 때문’이 단연 가장 속상한 일입니다. 일반적으로 프로젝트가 프로덕션까지 갔다면, 솔루션 개발과 관련된 선행 투자금은 이미 회사 경영진에게 이해를 시키고 승인을 받은 상태입니다. 회사에 어떤 영향을 미칠지 몰라 프로덕션까지 도달한 프로젝트를 취소하는 일은 완전히 다른 문제입니다. 솔루션의 가치를 입증할 수 없다면 언젠가 예산을 절약하기 위해 프로젝트를 중단하라는 말을 듣게 될 가능성이 큽니다.

최근 6개월간 새로운 이니셔티브에 부단히 투자해서 예측 모델링으로 매출을 높이려는 기업이 있다고 가정해 봅시다. 이 회사는 프로젝트를 개발하는 동안 모범 사례를 충실히 따랐습니다. 회사에서 요구하는 그대로 개발하였고 개발 노력을 유지와 확장이 가능한 코드에 집중하여 솔루션을 프로덕션까지 가져왔습니다. 이 모델은 지난 3개월 동안 훌륭한 성과를 냈습니다. 과거의 사건을 기반으로 미래에 대한 예측을 분석할 때마다 놀라울 정도로 정확했습니다.

그림 1.14의 시나리오에서는 이 ML 솔루션을 운영하는 비용에 대해 우려하는 임원 한 사람의 단순한 질문에서부터 틀어지기 시작합니다.

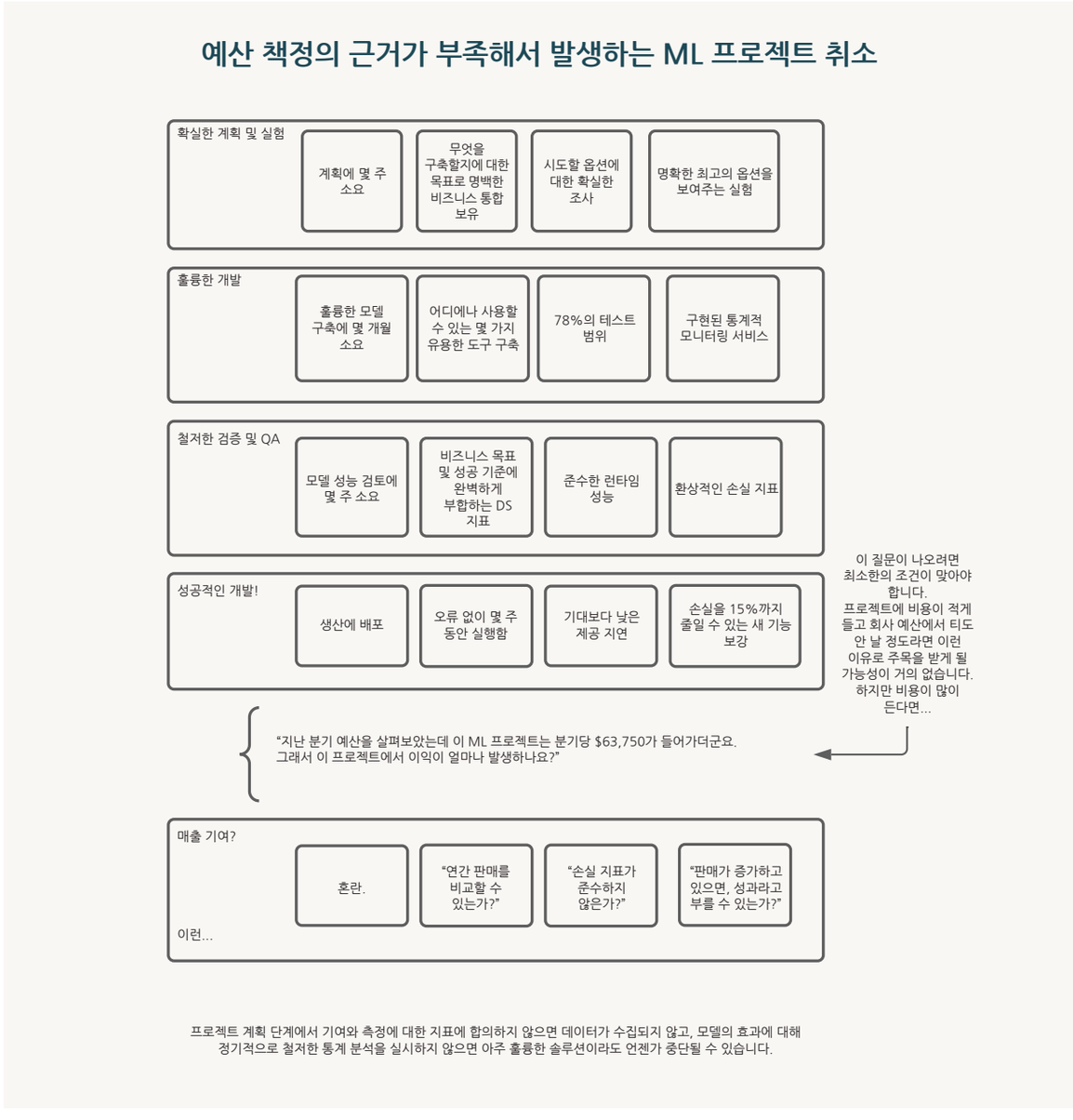


그림 1.14
A/B 테스트와 통계적으로 유효한 기여 측정이 없어서 거의 흠 잡을 곳이 없던 ML 프로젝트가 취소

이 개발팀은 훌륭한 ML 프로젝트를 개발했지만 한 가지 잊은 것이 있습니다. 바로 그림 1.14와 같이 ML 모델의 예측과 ML 모델의 존재를 정당화해주는 비즈니스 측면을 연결하는 것입니다. 이들이 개발하고 현재 프로덕션에서 운영 중인 모델은 매출을 높이도록 설계되었지만, 개발팀에서는 사용 비용을 자세히 살펴본 후 자신들이 솔루션의 가치를 증명할 기여 분석 방법론을 생각하지 않았다는 사실을 깨달았습니다. 매출을 합산하고 그걸 모델에서 발생했다고 간주하면 되지 않느냐고요? 아니요, 전혀 말도 안 되는 소리입니다. 작년 매출과 비교하면 되지 않느냐고요? 그것도 정확하지 않습니다. 매출에 영향을 미치는 지연 요소가 너무 많기 때문입니다.

모델에 대한 기여를 측정하려면 A/B 테스트를 실시하고 효과적인 통계 모델을 사용하여 (추정 오류가 포함된) 매출 향상 수준을 계산한 값을 산출하여 모델로 인해 발생한 추가적 매출이 얼마인지 보여주는 방법뿐입니다. 하지만 이미 엎질러진 물입니다. 모델은 이미 모든 고객에게 완전히 배포되었으니까요. 모델을 계속 유지하기 위한 근거를 제시할 기회를 놓쳤습니다. 이 프로젝트가 즉시 중단되지 않더라도 회사에서 예산을 줄여야 할 경우 도마에 오르게 될 것이 뻔합니다.

언제나 미리 생각하고 이런 경우에 대비하는 것이 좋습니다. 여러분에게 벌써 이런 일이 일어났든 그렇지 않든, 언젠가는 분명 일어날 겁니다. 모델을 계속 유지해야 할 근거를 보여주는 통계적으로 유의하고 검증된 테스트로 무장하고 있으면 자신의 작업물을 훨씬 쉽게 보호할 수 있습니다.

섹션 1: ML 엔지니어링이라는 개념

1.3 ML 엔지니어링의 목표

가장 기본적인 수준에서 데이터 사이언티스트의 기본 목표는 통계, 알고리즘 및 예측 모델링을 사용하여 인간이 하기에는 매우 번거롭거나, 단조롭거나, 오류가 발생하기 쉽거나, 너무 복잡하고 까다로운 문제를 해결하는 것입니다. 가장 좋은 모델을 구축하는 것도, 솔루션에 대한 가장 인상 깊은 연구 논문을 작성하는 것도, 프로젝트 작업에 적용할 최첨단 기술을 찾아내는 것도 아닙니다.

소프트웨어 엔지니어링의 기본(DevOps)을 ML이라는 분야에 적용하는 목표는 다른 무엇보다도 DevOps를 만든 바로 그 이유와 일맥상통합니다. 즉 프로젝트의 효율을 높이고 매우 복잡한 코드베이스를 쉽게 관리할 수 있도록 하기 위해서입니다. 또한 프로젝트가 취소되고, 코드를 폐기하고, 솔루션이 빛을 보지 못할 가능성을 낮추기 위해서입니다.

이 직업에 종사하는 우리 모두는 문제 해결을 목표로 합니다. 우리에게는 사용할 수 있는 도구가 여러 가지 있고, 고려해야 할 옵션도 많으며, 효과적으로 개발하기 위해서 감당하기 어려울 정도로 복잡한 지식을 습득하고 유지해야 합니다. 이런 태산 같은 복잡성과 디테일에 어찌할 바를 모르게 되기 쉬우며, 우리 프로젝트는 (예산, 시간, 복잡성으로 인해) 취소되거나, (코드의 관리가 어렵거나, 취약하거나, 또는 그 둘 모두로 인해) 중도 포기하거나, 목표가 모호하거나, 예측이 불안정하거나, 비즈니스 요구사항이 없어서 우선순위가 조정되는 어려움을 겪습니다. 이는 몇 가지 핵심적 규칙을 신중히 적용한다면 모두 예방 가능한 문제입니다.

섹션 1.2에서 강조하고 이 eBook 전체에서 자세히 설명하는 프로젝트 작업의 핵심적 측면에 집중한다면, ML 개발을 원하는 상태, 즉 모델을 프로덕션에서 실행하고 실질적 비즈니스 문제를 해결하는 상태로 만들 수 있습니다. 이런 방법론, 기술 및 설계 패턴을 사용할 때의 목표는 여러분이 목적으로 하는 문제를 해결하는 데 시간과 에너지를 집중해서 더 많은 문제를 해결하고 회사와 여러분 모두 예측 모델링이 제공하는 모든 장점을 누리며 성공을 거두는 것입니다. 실패율과 중도 포기율을 낮춰서 앞으로의 작업에서는 잘못 구상한 솔루션에 관해 사과하고, 재작업하고, 끊임없이 유지 관리에 골몰할 필요 없이 다른 데 집중할 수 있도록 해야 합니다.

할 수 있다!

세상에는 아예 여러분의 무능력함을 확신시키려고 마음 먹고 탄생한 업종도 있습니다. 즉 이런 복잡한 작업은 그런 업종의 업체들을 꼭 고용해야만 한다고 주장하는 것입니다. 이 사람들은 이런 일로 거액의 수익을 거둡니다.

하지만 여러분이 이런 핵심 개념을 배워서 ML 작업에 접근하는 한 가지 방법론을 따르는 팀을 직접 구축하면 프로젝트 작업 성공률을 대폭 높일 수 있습니다. 처음에는 복잡하고 다소 혼란스러울 수 있지만, 이런 가이드라인을 따르고 적절한 도구를 사용하면 어느 팀이나 대규모 예산이 필요하거나 데이터 사이언스 팀이 잘못 구현된 솔루션을 "유지"하느라 여가 시간을 모조리 투자하지 않아도 상당히 정교한 ML 솔루션을 개발할 만큼 복잡성을 관리할 수 있습니다.

여러분은 할 수 있습니다.

ML 엔지니어링의 방법론과 접근법 각각을 자세히 살펴보기 전에 그림 1.15에 나와 있는 개요를 확인해봅시다. 이는 프로덕션 ML 작업을 위한 프로세스 흐름도입니다. 여러분의 작업은 이것과 충격적일 정도로 흡사할 수도 있고, 여기 표시된 것보다 훨씬 덜 복잡할 수도 있습니다. 이 그림을 여기 제시한 것은 여러 가지 주제의 개념을 소개하고자 했기 때문입니다.

앞서 언급했듯이, 이것이 바로 여러분의 ML 프로젝트 작업이 실제로 기본 목표에 부합하도록 보장하는 검증된 방법입니다. 관리하기에 어렵지 않은 유용한 무언가를 개발하고, 데이터 기술, 수학적 알고리즘 학문, 우리의 무한한 상상력을 동원하여 문제를 해결하는 것이 그러한 기본 목표라 하겠습니다.

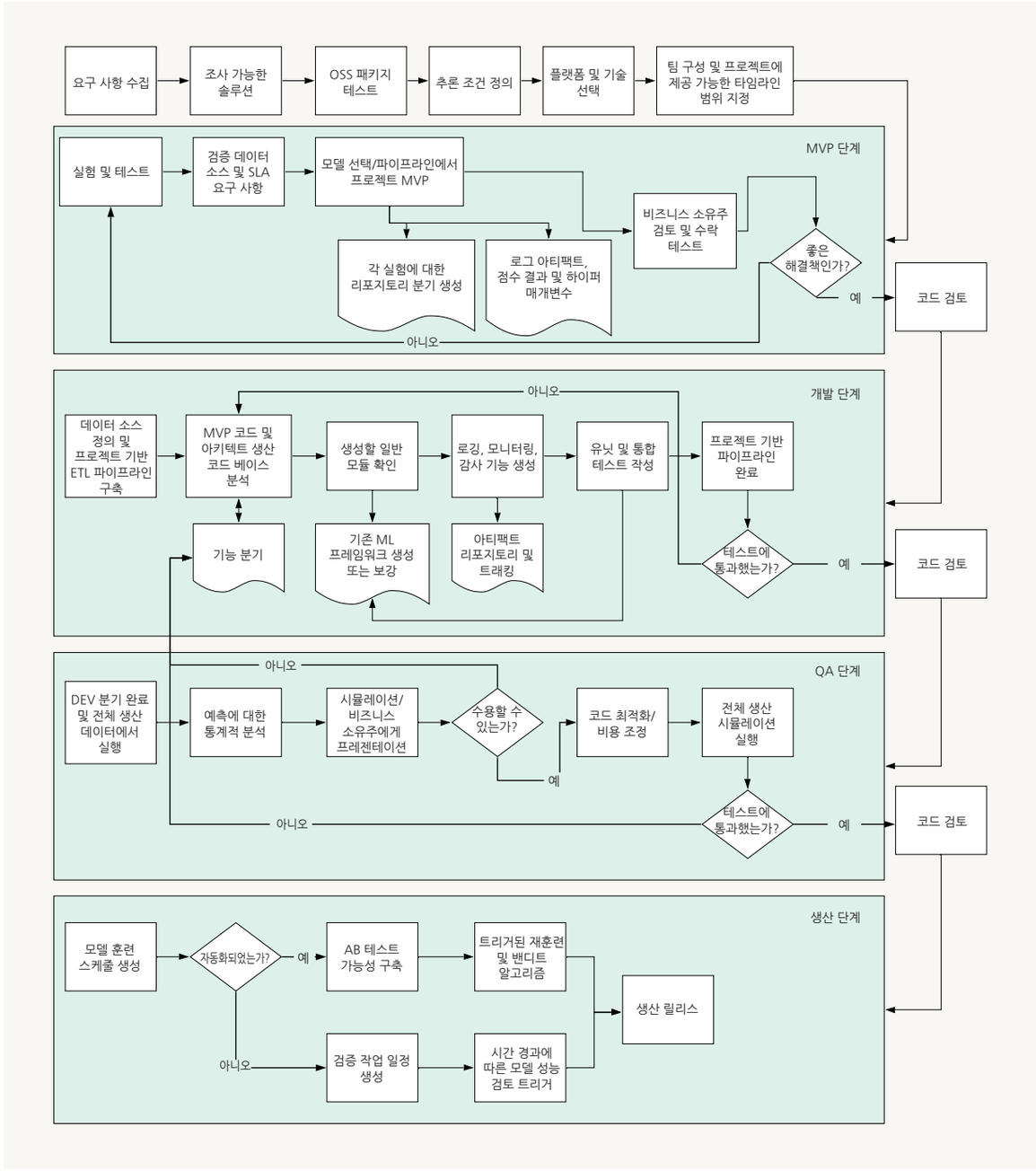


그림 1.15
ML 엔지니어링 방법론 구성 요소 맵이자 이 eBook의 시각적 목차.

섹션 1: ML 엔지니어링이라는 개념

1.4 요약

기업에서 ML 프로젝트 작업 실패율이 높은 주요 원인은 ML 엔지니어링의 6가지 원칙을 따르지 않았기 때문입니다(또는 이를 모르기 때문입니다).

ML 엔지니어링은 일종의 가이드입니다. 또한 데이터 사이언티스트가 참여하는 프로젝트 작업이 모범적인 엔지니어링 원칙을 따르고 비즈니스 요구 사항을 해결하는 데 집중할 수 있는 도구 상자이자 지도이기도 합니다.

앞서 그 핵심 요소를 매우 간략히 살펴보았습니다. 섹션 2에서는 실제 예시, 프로젝트 솔루션, 결정 시뮬레이션을 통해 각 주제를 깊이 있게 살펴보고 머신 러닝을 사용하여 성공적이고 관리 가능하면서도 회복력 있는 코드 베이스를 개발하는 데 필요한 도구를 소개하겠습니다.

섹션 02

데이터 사이언스에서 엔지니어링의 활용

섹션 2

데이터 사이언스에서 엔지니어링의 활용

섹션 1에서는 데이터 사이언스(DS)에 대한 전문적이고 성공적인 전략 위주로, ML 엔지니어링이 필요한 이유에 대해 다루었습니다. 이 섹션에서는 그 "이유"에 집중합니다. 전 세계 기업에서 데이터 사이언스가 지속적으로 성장하면서 데이터 사이언스 프로젝트 작업에 대한 기본적인 방법론이 정립되었을 뿐만 아니라, 기업에서 ML을 적용하여 문제를 해결하고 매출을 높이기 시작하며 광풍이 불었습니다.

안타깝게도 현실적으로는 갑작스럽게 ML에 능숙한 엔지니어 수요가 늘면서, 과도한 수요를 감당하기에는 경험이 풍부한 인력(오랫동안 그 직무를 수행하면서 몸으로 체득한 인력)이 부족합니다. 게다가 데이터 사이언스 작업의 성격이 극히 심층적이고 복잡해서 능숙해지는 데 오랜 기간이 걸리기 때문에 많은 기업에서는 ML을 활용한 프로젝트에서 엄청난 좌절과 실망을 맛보는 경우가 많고 극단적인 경우에는 프로젝트 자체를 중도 포기합니다.

이 섹션에서는 ML 엔지니어링의 에코시스템을 정의하고 이러한 표준이 개발된 핵심적 이유와 그 내용을 자세히 설명하겠습니다.

ML 엔지니어링(또는 "MLOps" 일반 소프트웨어 개발의 도구, 방법론, 프로세스를 모아 애자일 측면을 강조한 용어인 "DevOps"에서 파생된 용어)은 특정 직책에 부여하는 업무가 아닙니다. 데이터 사이언스와 모순되는 작업도 아닙니다. 또한 데이터 사이언스 솔루션과 완전히 분리된 책임 영역도 아닙니다. 오히려 데이터 사이언스를 위한 추가적인 도구, 프로세스, 패러다임을 모은 *보완적*(매우 필수적이라고도 할 수 있는) 세트입니다. ML 엔지니어링의 최종 목표는 훨씬 낮은 총소유비용으로 비즈니스 문제를 해결하는 더욱 지속 가능한 솔루션을 개발하는 것입니다. 결국, **모든 데이터 사이언스의 기본적 핵심은 *문제 해결*에 있습니다**. 유지 가능성과 효율에 집중한 검증된 방법론에 대한 작업 패턴을 준수하면 **훨씬 적은 노력으로 더 많은 문제를 해결하는 것으로 직결됩니다**.

섹션 2: 데이터 사이언스에서 엔지니어링의 활용

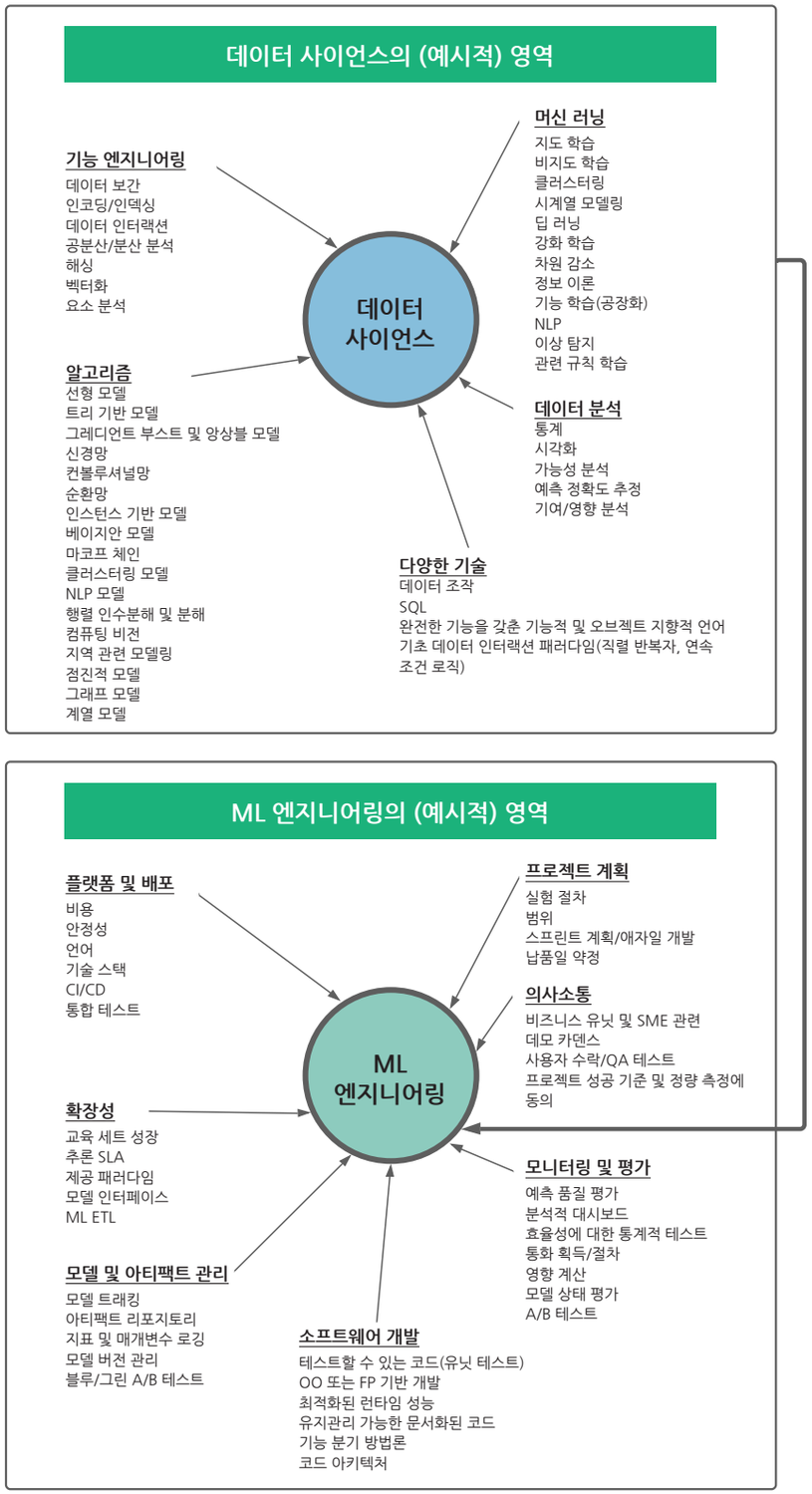
2.1 프로젝트를 더 큰 성공으로 이끄는 프로세스로 복잡한 업무 보강

데이터 사이언스"라는 용어가 가장 처음 등장한, 1996년에 출간된 서적 "Data Science, Classification, and Related Methods"(편집: C. Hayashi, C. Yajima, H. H. Bock, N. Ohsumi, Tanaka, Y. Baba)에서는 이를 세 가지 핵심 영역으로 정의했습니다.

- **데이터 설계**: 특히, 정보를 수집하는 방법과 특정 문제를 해결하기 위해 정보를 획득하는 구조에 대한 계획
- **데이터 수집**: 그러한 데이터를 획득하는 행위
- **데이터 분석**: 통계적 방법론을 사용하여 데이터에서 인사이트를 얻어 문제 해결

최신 데이터 사이언스는 대체로 데이터 분석과 관련이 있는데(데이터 사이언스 팀이 자체 ETL을 개발할 수밖에 없는 상황이 많기는 하지만), 이는 첫 두 가지 중점 영역을 데이터 엔지니어링 팀이 담당하는 것이 보통이기 때문입니다. "데이터 분석"은 광범위한 용어이지만, 요즘 데이터 사이언티스트의 주력 분야는 대체로 통계 기술 적용, 데이터 조작 활동 및 통계적 알고리즘(모델)을 활용한 인사이트 획득, 데이터 예측 등을 포함한 이 카테고리에 속한다고 볼 수 있습니다.

그림 2.1 위쪽에서는 (간략하게 요약한) 기술적 관점에서 본 현대적 데이터 사이언티스트의 주력 분야를 보여줍니다. 이것들은 데이터 사이언티스트의 업무에 대해 설명할 때 대부분의 사람이 초점을 맞추는 직업적 요소입니다. 데이터 액세스에서, 다양한 알고리즘 접근법과 고급 통계를 활용하여 복잡한 예측 모델을 구축하기에 이르기까지 다양한 요소가 있습니다. 이 그림은 데이터 사이언티스트가 프로젝트 작업에 참여할 때 하는 일을 정확히 나타낸 것은 아니고, 문제 해결에 사용하는 도구와 작업에 초점을 맞추었습니다. 이런 방식으로 데이터 사이언스에 대해 생각하는 것은 소프트웨어 개발자의 직무를 언어, 알고리즘, 프레임워크, 컴퓨팅 효율 및 기타 기술적 고려 사항으로 분류하려 드는 것만큼 도움이 안 되기는 마찬가지입니다.



이러한 기술을 프로덕션 등급 솔루션 개발과 배포 패러다임에 적용

그림 2.1
데이터 사이언스의 핵심 기술(위)과 성공적인 데이터 사이언스 프로젝트로 이끄는 광범위한 방법론, 도구, 프로세스 영역에서 데이터 사이언스의 위치(아래). 이런 다양한 기술을 마스터하면 프로젝트 성공 확률을 크게 개선(그 결과를 실제로 사용)할 수 있습니다.

그림 2.1 위쪽에서 설명하는 데이터 사이언스의 기술적 관점은 (대부분 실무자들이 여기에만 집중하기는 하지만) 아래에 나와 있는 광범위한 시스템의 한 가지 측면일 뿐입니다. 이는 ML 엔지니어링의 영역에 포함됩니다. ML 엔지니어링은 보완적 도구, 프로세스, 패러다임이 지침 프레임워크를 제공하며, 기본적으로 데이터 사이언스 기술의 핵심적 측면의 지원을 받아 더욱 건설적인 방식으로 일할 수 있도록 합니다. ML 엔지니어링이라는 개념은 일종의 패러다임이라고 볼 수 있습니다. 즉 실무자가 프로젝트 작업 중 정말 중요한 측면, '실제로 효과가 있는 솔루션으로 문제를 해결하는 것'에만 집중할 수 있게 해줍니다.

하지만 어디서부터 시작해야 할까요?

섹션 2: 데이터 사이언스에서 엔지니어링의 활용

2.2 단순성의 기반

데이터 사이언티스트의 실제 업무가 무엇인지 설명하려면 “수학을 데이터에 창의적으로 적용하여 문제를 해결한다”라는 말로 명료하게 표현할 수 있습니다. 그 정의가 광범위한 만큼, 기록된 정보(데이터)에서 개발할 수 있는 다양한 솔루션을 나타내기도 합니다. 데이터 사이언티스트가 비즈니스 문제 해결책을 모색하는 과정에서 알고리즘, 접근 방식이나 기술에 관해 무슨 일을 하든 세간의 예상에는 해서는 안 되는 일 같은 것은 없습니다(적어도 제가 아는 한에서는). 사실, 그 반대라 해도 과언이 아닙니다. *데이터 사이언티스트*는 다양한 기술과 접근법으로 *문제를 해결하는 사람*입니다.

이 분야에 처음 입문한 이들에게는 유감스럽게도, 대다수의 데이터 사이언티스트는 자기가 최근 출현한 기술 중에서도 최신, “제일 훌륭한” 기술을 사용해야만 회사를 위해 가치를 창출한다고 믿습니다. 반면 경험이 풍부한 데이터 사이언티스트는 정기적으로 발행되는 백서나 블로그 게시물에 화려하게 광고된 새로운 접근법을 둘러싼 입소문에 휘둘리지 않고, 어떤 방법을 사용하든 문제를 해결하는 것이 가장 중요하다는 점을 알고 있습니다. 새로운 기술과 접근법이 흥미롭기는 하지만 데이터 사이언스 팀의 능률은 이들이 제공하는 솔루션의 품질, 안정성, 비용으로 측정됩니다.

그림 2.2와 같이, ML 작업의 가장 중요한 부분 중에는 각종 문제에 직면했을 때 복잡성을 처리하는 것이 있습니다. 회사에서 새로운 요청이 있을 때 이런 생각을 ML 원칙의 진리(비즈니스 문제를 최대한 단순한 솔루션으로 해결하는 데 집중)라고 생각하고 접근하면, 특정 접근법이나 멋져 보이는 새로운 알고리즘 대신 솔루션 자체에 집중할 수 있습니다.

그림 2.2와 같이, 문제를 해결하기 위해 최대한 단순한 구현 방안을 추구하는 원칙에 집중한다면 이는 ML 엔지니어링의 다른 모든 측면을 구성하는 기반이 됩니다. 이 원칙이야말로 ML 엔지니어링에서 단연 가장 중요한 요소입니다. 다른 모든 프로젝트 작업, 범위 설정, 구현에 영향을 미치기 때문입니다. “최대한 쉽게 해결책을 찾는 것”이야말로 프로젝트의 성패를 가르는 가장 큰 요소일 수 있습니다.

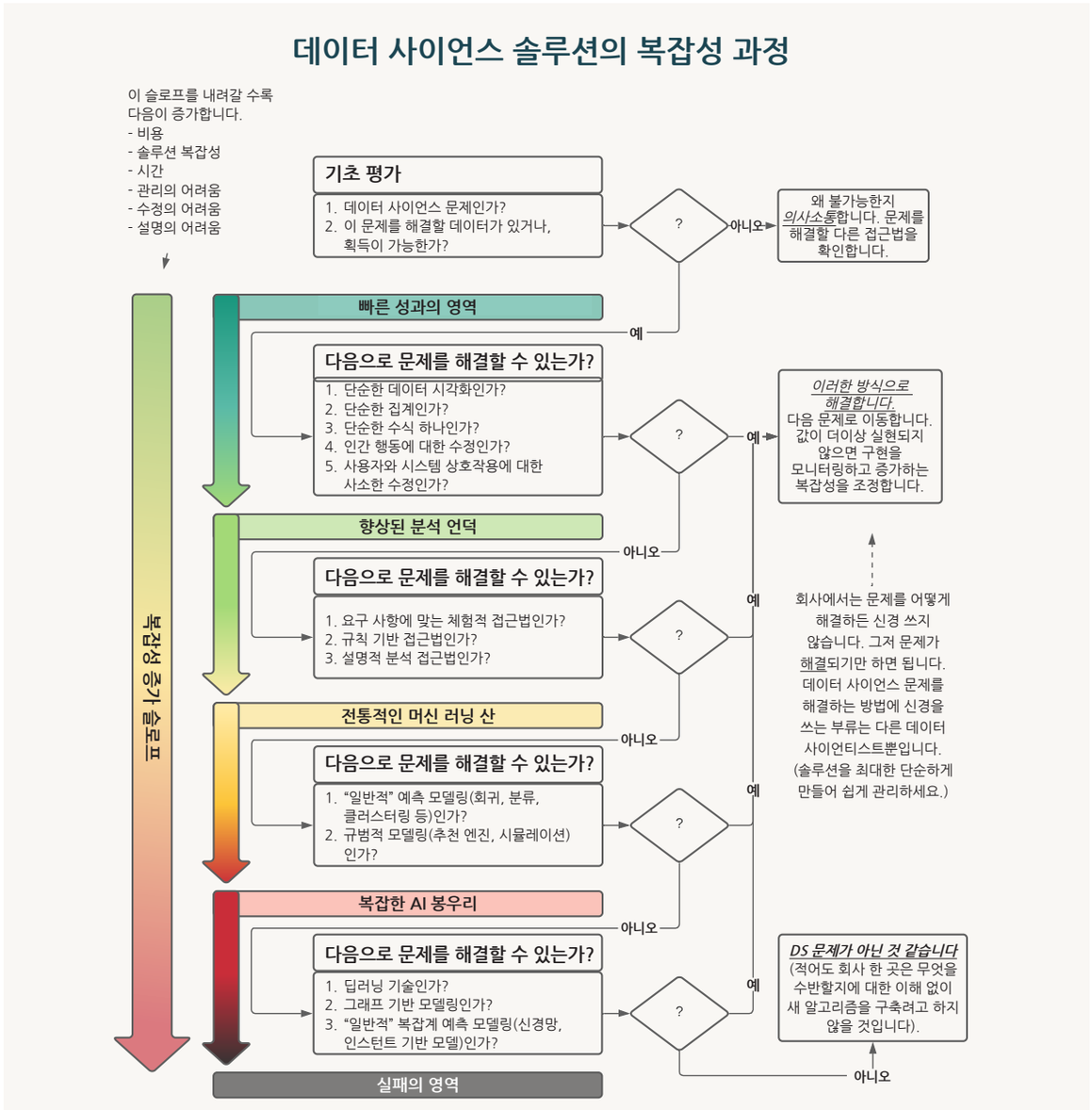


그림 2.2
데이터 사이언스 프로젝트 구현 시 복잡성을 최소화하는 결정 경로. 기본 평가 시점(이 문제가 데이터 사이언스 문제가 맞기는 한가?)에서부터 끝까지(기존 기술로는 도저히 해결이 불가능) 프로젝트를 진행하면서 복잡성이 커지는 순서에 따라 결정을 내리는 것이 좋습니다. 최대한 간단한 솔루션으로 문제를 해결할 수 있다면, 그 방법이 가장 적절한 솔루션입니다. 가장 “인상적인” 솔루션이나 최신 유행을 따르면 복잡성, 비용, 개발 시간, 해석 능력 부족으로 프로젝트가 실패할(비즈니스에서 사용하지 못할) 위험이 커지게 되어 있습니다.

섹션 2: 데이터 사이언스에서 엔지니어링의 활용

2.3 애자일 소프트웨어 엔지니어링의 포용(co-opting) 원칙

DevOps는 소프트웨어 개발에 성공적인 엔지니어링 개발에 대한 가이드라인과 입증 가능한 패러다임을 제공했습니다. 애자일 매니페스토가 등장하면서, 노련한 업계 전문가들은 소프트웨어 개발 방식에 문제가 있다는 것을 깨달았습니다.

그림 2.3에서 볼 수 있듯이, 애자일 개발의 원칙을 약간만 바꾸면 데이터 사이언스를 비즈니스 문제에 적용할 때의 가이드라인으로 만들 수 있습니다.

이 eBook에서는 이러한 주제를 모두 다루고, 이 원칙들이 중요한 이유와 함께 비즈니스 문제 해결에 적용하는 방법의 예시를 설명할 것입니다. 원래의 애자일 원칙과는 상당히 달라진 것들도 조금 있지만, ML 프로젝트 작업은 응용력이 뛰어난 관계로 저희 팀은 물론 다른 많은 이들에게도 거듭해서 효과가 있어 일정한 성공 패턴이 생겼습니다.

단, ML 프로젝트 작업에 적용하면 데이터 사이언스 팀이 업무에 접근하는 사고방식을 대폭 개선해 주는 애자일 개발 요점으로 정말 중요한 것이 두 가지 있습니다.

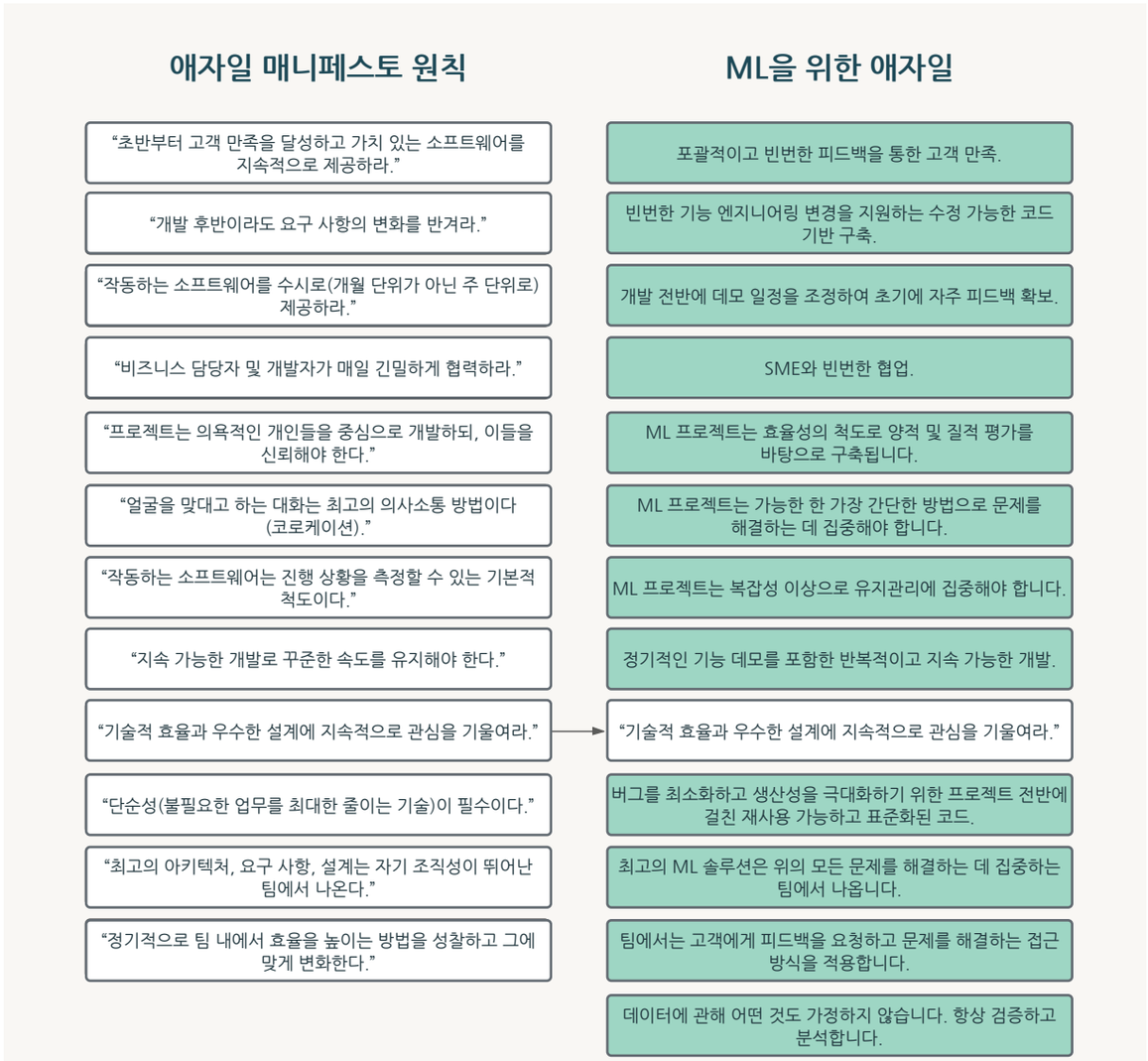


그림 2.3
ML 프로젝트 개발에 맞게 수정한 애자일 매니페스토

의사소통 및 협력

이 eBook에서 (특히 다음 두 섹션에 걸쳐) 여러 번 설명하겠지만 성공적인 ML 솔루션 개발의 핵심적 원칙은 '사람'에 있습니다. 이 말은 수학, 과학, 알고리즘, 영리한 코딩으로 둘러싸인 직종의 성격과는 완전히 반대되는 것 같습니다. 그러나 좋은 문제 해결 솔루션은 절대 거저 구현되지 않습니다. 솔루션 개발을 둘러싼 도구와 공식적 프로세스(또는 문서화)보다는 프로젝트 및 그 현황과 관련된 의사소통과 사람에 더욱 집중한 프로젝트가 가장 큰 성공을 거두었습니다.

기존 애자일 개발에서와 마찬가지로, ML 개발에서는 솔루션을 코딩하는 사람과 **솔루션을 요청하는 사람 사이의 상호작용**이 더욱 중요합니다. 이는 솔루션 구축과 관련된 복잡성이 크기 때문입니다. ML 개발 작업 대부분은 일반인에게는 익숙하지 않은 일이 대부분이고 수년간 이 분야를 연구하고 지속적으로 학습해야 마스터할 수 있기 때문에 의미 있고 유용한 논의를 하는 것이 중요합니다.

재작업을 최소화하면서 프로젝트를 성공시킬 수 있는 가장 큰 비결은 ML 팀과 사업부가 서로 *협업하며 개입*하는 것입니다. 두 번째로 중요한 비결은 ML 팀 내부의 의사소통입니다.

프로젝트에 '외로운 늑대' 마인드로 접근하면(학생 때 대부분의 사람들이 취하는 태도) 까다로운 문제를 해결하는 데 역효과를 낳습니다. 그림 2.4는 이런 위험한 행동을 보여줍니다.

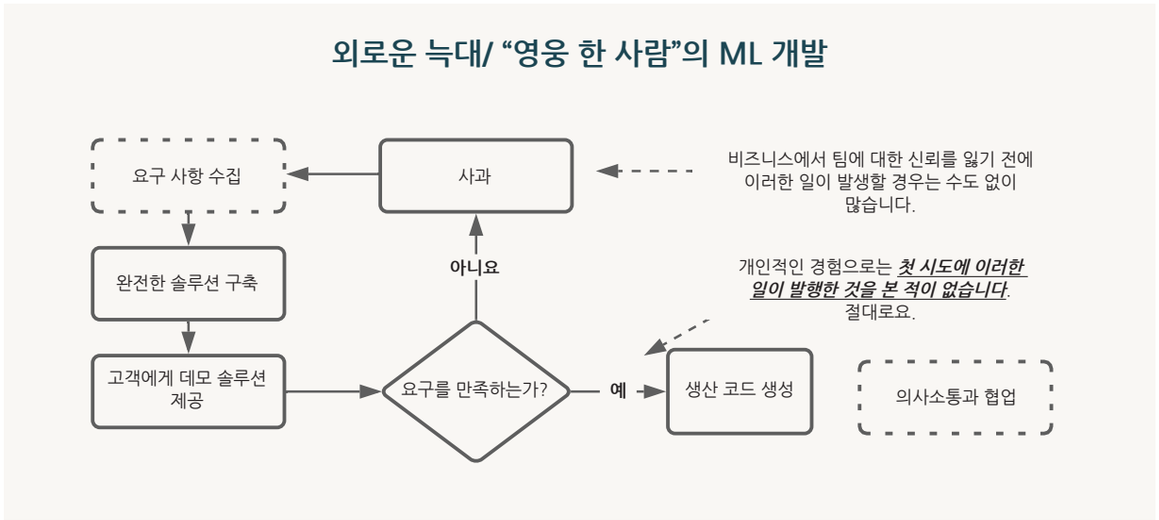


그림 2.4
모든 ML 솔루션을 혼자 개발하면서 힘겹게 얻은 교훈. 이런 경우는 대부분 끝이 좋지 못합니다.

그림 2.4와 같은 개발 스타일을 추구하는 이유는 다양할지 몰라도 최종 결과는 대체로 동일합니다. 재작업을 많이 하거나, 사업부가 엄청난 불만을 품게 됩니다. 다른 데이터 사이언스 팀원이 없다 하더라도(1인 체제 "팀"인 경우) 피어 리뷰를 요청하고, 다른 소프트웨어 개발자, 아키텍트, 솔루션을 요청한 사업부의 주제 전문가에게 솔루션을 보여주는 것이 좋습니다. 요구 사항을 들은 후, 누구와도 대화를 나누지 않고 바로 코딩으로 문제를 해결하려는 행동이야말로 절대 해서는 안 되는 일입니다. 프로젝트 요구 사항에 전부 부합하고, 극단적인 변수를 바로잡아 고객이 기대하는 결과물을 구축할 가능성은 극히 미미합니다. 이게 가능한 사람이라면 운이 차고 넘치는 사람이니 복권이라도 사보는 것이 좋을지 모르겠습니다.

더욱 종합적이고 애자일에 맞는 ML 개발 프로세스는 일반 애자일 소프트웨어 개발과 매우 유사합니다. 주된 차이는 소프트웨어 개발에는 대체로 추가적인 내부 데모가 필요 없다는 것뿐입니다(이 경우 피어 리뷰 피처 분기만으로 충분). ML 개발의 경우 코드로 전달되는 데이터에 미치는 영향을 기준으로 성능을 보여주고 기능과 시각적 결과 자료를 보여주는 것이 중요합니다. 그림 2.5는 대내외적인 협업과 의사소통을 중심으로 한 모범적인 애자일 ML 개발을 나타냅니다.

팀원들끼리 상호작용 수준이 높을수록 더 많은 아이디어와 의견이 나오고, 가정한 사실에 대한 이의 제기가 많아 더 수준 높은 솔루션을 도출할 수 있다는 것은 거의 항상 진실입니다. 고객(도움을 요청하는 사업부)이나 동료를 논의(개발 관련 선택에 관한 아주 사소한 부분이라 하더라도)에서 빠뜨린다면 기대하지 않았거나 원하지 않았던 결과물을 개발할 가능성이 커집니다.

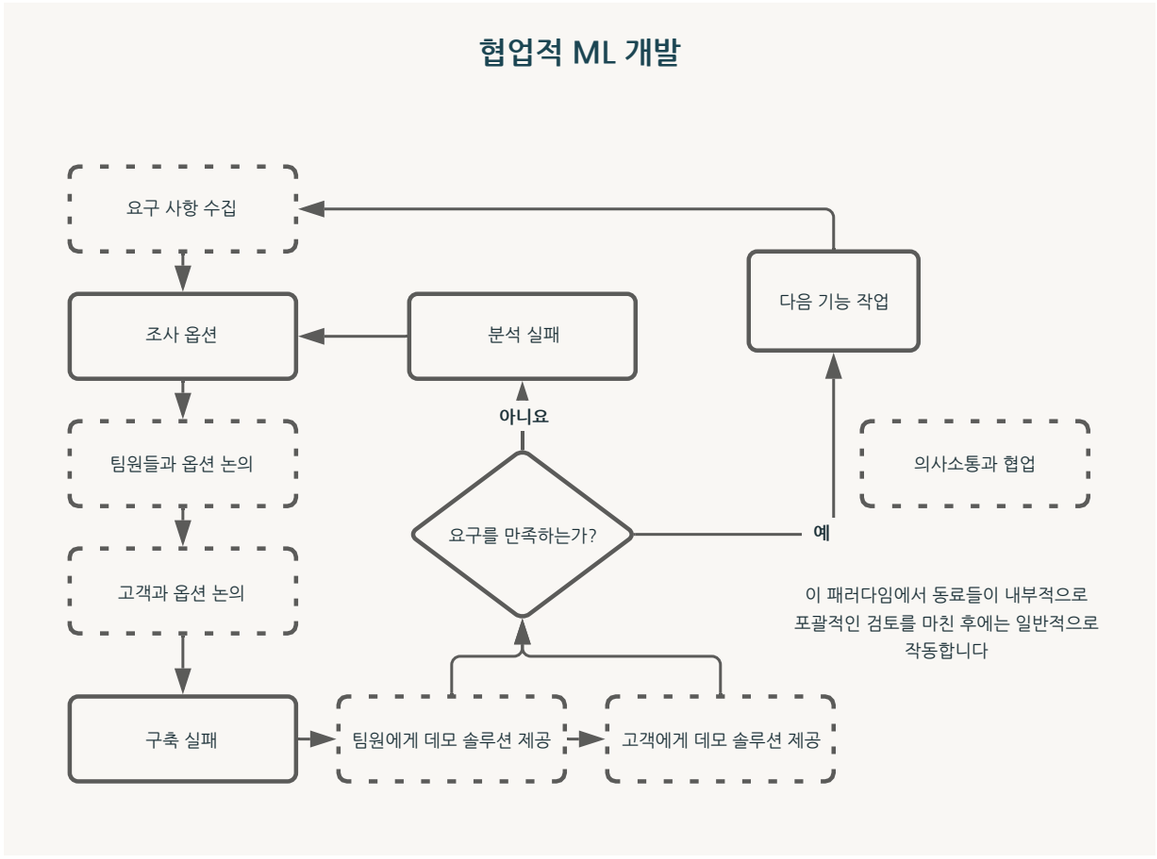


그림 2.5
지속적인 의사소통, 개방적 피드백, 협업에 집중한 애자일 기반 ML 프로젝트 개발. 이런 프로젝트 개발 방법은 매우 성공적인 것으로 입증되었습니다.

변화를 받아들이고 예상하기

변화에 대비하고 실험과 프로젝트 방향성뿐만 아니라 개발에서도 변화는 불가피하다고 생각하는 것이 무엇보다도 중요합니다. 거의 모든 ML 프로젝트에서 처음에 설정한 목표는 프로젝트가 끝날 무렵의 결과물과 일치하는 법이 없습니다.

이는 특정 기술, 개발 언어, 알고리즘에서부터 데이터에 대한 가정이나 기대, 심지어는 ML을 활용한 최초의 문제 해결(예: 간단한 집계 대시보드로 더욱 효율적으로 문제 해결)에 이르기까지 모든 것에 적용됩니다.

불가피한 변화에 대비한다면 데이터 사이언스 작업에서 *가장 중요한 것, 바로* ***문제 해결****에 집중할 수 있습니다*. 또한, 중요하지 않은 요소(화려한 알고리즘, 멋진 새로운 기술, 놀라울 정도로 강력한 솔루션 개발 프레임워크)에 한눈을 팔지 않을 수 있습니다.

변화를 예상하거나 허용하지 않는다면 프로젝트 구현에 대해 그때까지 했던 모든 작업을 완전히 새로 작성하지 않고는 수정이 매우 어려운(혹은 불가능한) 결정을 내려야 할 수도 있습니다. 프로젝트 방향성이 어떻게 바뀔지 생각한다면 기능이 느슨하게 결합된 모듈 방식을 선택할 수밖에 없고, 방향성을 바꾸더라도 이미 완료한 다른 작업에 미치는 영향이 줄어들게 됩니다.

애자일은 이렇게 느슨하게 결합된 설계의 개념을 도입하고 빠른 반복을 통해 새로운 기능을 구축하는 데 집중하므로 요구 사항이 동적으로 변화하더라도 계속 작동할 수 있습니다. 이 패러다임을 ML 개발에 적용하면 뒤늦게 갑작스러운 변화가 생기더라도 비교적 간단하게 대응할 수 있습니다(물론, 타당한 수준에서 말입니다. 2주 만에 트리 기반 알고리즘에서 딥러닝 알고리즘으로 전환하기는 불가능한 일입니다). 과정이 *단순화* 되었기는 하지만, 그렇다고 해서 **단순성이 보장되는 것은 아닙니다**. 하지만 변화를 예상하고 신속한 반복과 수정을 지원하는 프로젝트 아키텍처를 구축하면 개발 프로세스가 훨씬 수월해집니다.

섹션 2: 데이터 사이언스에서 엔지니어링의 활용

2.4 ML 엔지니어링의 기반

지금까지 애자일 원칙을 ML에 도입하여 데이터 사이언스의 기반이 되는 원칙을 알아보았습니다. 이제 이 프로젝트 개발 시스템의 에코시스템 전체를 간단히 살펴보겠습니다. 저도 이를 통해 업계 내에서 탄력적이고 유용한 문제 해결 솔루션을 개발하는 데 성공한 사례를 많이 보았습니다.

앞서 언급했듯이 ML 엔지니어링(MLOps)을 패러다임으로 생각하는 개념은 소프트웨어 개발에서 DevOps의 개념에 대해 유사한 원칙을 적용한 것에서 비롯되었습니다. 그림 2.6은 DevOps의 핵심적 기능을 보여줍니다.

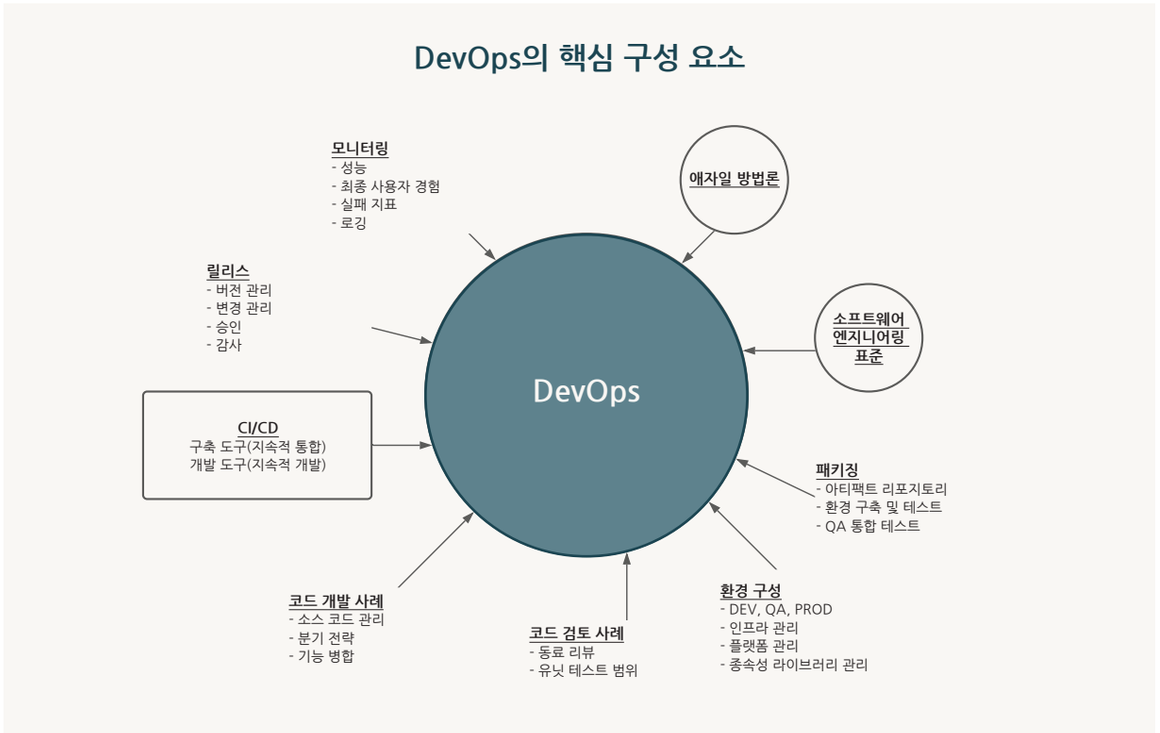


그림 2.6
DevOps의 구성 요소

섹션 2.3에서 애자일에 대한 핵심 원칙을 비교했듯이, 그림 2.7은 DevOps의 "데이터 사이언스 버전" 즉 MLOps를 보여줍니다. 이러한 각각의 요소들을 병합하고 통합하면 데이터 사이언스 작업에서 최악의 결과라고 할 수 있는 솔루션 실패, 취소나 채택 거부와 같은 불상사는 완전히 방지할 수 있습니다.

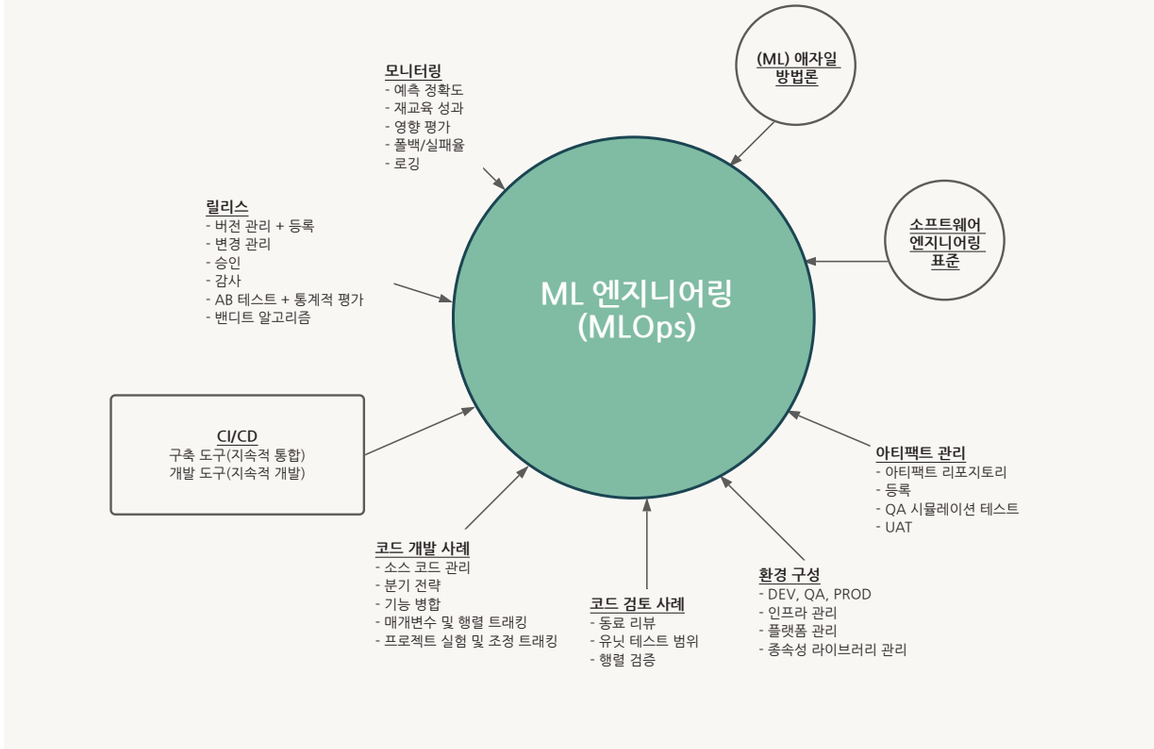


그림 2.7
DevOps 원칙을 ML 프로젝트 개발(MLOps)에 도입

섹션 2: 데이터 사이언스에서 엔지니어링의 활용

2.5 요약

표준 프로세스, 도구, 방법론을 데이터 사이언스 기술 역량을 보강하는 데 적용하면 프로젝트 성공률을 높이는 데 도움이 됩니다.

데이터 사이언스 프로젝트의 목표를 특정 툴링이나 알고리즘을 쓰는 데 두어서는 안 됩니다. 그보다는, 모든 데이터 사이언스 프로젝트의 근본적인 목표는 주어진 문제를 해결하기 위한 가장 단순한 방법(분석 시각화보다 조금 나은 수준에 그친다 하더라도)을 찾기 위해 애쓰는 데 있어야 합니다.

애자일 개발에서 수정한 원칙에 초점을 맞추면 성공이 보장된 데이터 사이언스 개발 패턴을 수립하고, 더욱 수준 높고 유지 관리가 수월한 솔루션을 개발할 수 있습니다.

ML 엔지니어링(MLOps)의 에코시스템은 DevOps의 수많은 프로세스와 표준을 수정하고, 특정한 툴링과 영역별 요소를 더해 탄력적이고 관리 가능한 프로덕션급 데이터 사이언스 솔루션을 구현할 수 있도록 구성되었습니다.

심화

eBook에서 설명한 각각의 원칙을 자세히 살펴보고 싶다면 Manning 웹사이트에서 “Machine Learning Engineering in Action” 서적을 구매하세요. 여기서 다룬 주제들에 각각 한 챕터를 할애해 심층적으로 다루며, 각 요소가 중요한 이유뿐만 아니라 유용한 예시와 실제로 따라 하면서 자신의 실무 능력을 키울 수 있는 실제 구현도 제시합니다. 이 책의 목표는 여러분이 머신 러닝을 성공적으로 활용하도록 돕는 데 있습니다.

이제 프로덕션 등급 머신 러닝 프로젝트를 개발하는 비결을 알았으니 그곳에 도달할 수 있는 플랫폼을 확인해보세요.

Databricks Machine Learning 은 모든 수명 주기를 아우르는 데이터 네이티브 협업 ML 플랫폼으로 데이터 엔지니어링을 도구로 사용하여 확장 및 복제가 가능한 모델을 구축하는 데 집중할 수 있습니다.

세계에서 가장 많이 사용하는 오픈 소스 프로젝트인 MLflow 및 Delta Lake를 기본으로 제공하는 Databricks Machine Learning은 피처 엔지니어링(featurization)부터 훈련, 튜닝, 제공, 모니터링에 이르기까지 모든 과정에서 머신 러닝의 속도를 높여 줍니다.

데모를 확인하고, 기능을 자세히 살펴보고, Databricks Machine Learning에서 제공하는 최신 솔루션 액셀러레이터를 확인해보세요.

본격적으로 진짜 머신 러닝을 시작할 준비가 되셨나요?
Databricks 솔루션 액셀러레이터페이지에서 업종,
사용 사례별로 바로 배포 가능한 코드를 확인하세요.

무료로 시작하기

databricks

Databricks 소개

Databricks는 데이터 및 AI 회사입니다. Comcast, Condé Nast, H&M 및 Fortune 선정 500대 기업의 40% 이상을 포함하여 전 세계적으로 5,000개 이상의 기업이 Databricks 레이크하우스 플랫폼을 사용하여 데이터, 분석 및 AI를 통합합니다. Databricks는 샌프란시스코에 본사가 있으며 전 세계에 지사를 두고 있습니다. Apache Spark™, Delta Lake 및 MLflow의 원제작자가 설립한 Databricks는 데이터팀이 세계의 어려운 문제들을 해결할 수 있도록 지원하는 사명을 가지고 있습니다. Databricks에 대해 더 알고 싶으시다면 Twitter, LinkedIn, Facebook을 팔로우해 주세요.