December 10, 2025

▣ **강의명:** CSCI E-103: 재현 가능한 머신러닝

▣ **주차:** Lecture 09

▣ **교수명:** Anindita Mahapatra
Eric Gieseke

▣ **목적:** Lecture 09의 핵심 개념 학습

□ 핵심 요약

이 문서는 머신러닝(ML) 모델을 개발하고 운영하는 전 과정을 다루는 **MLOps (Machine Learning Operations)**의 핵심 개념을 정리합니다. MLOps는 단순히 모델을 만드는 것을 넘어, 신뢰할 수 있고 반복 가능한 방식으로 모델을 배포하고 모니터링하는 프로세스입니다.
이 노트는 MLOps에 참여하는 **다양한 역할**(데이터 엔지니어, 과학자, ML 엔지니어 등)을 정의하고, 고객 이탈 방지 사례를 통해 **3대 핵심 파이프라인(학습, 추론, 모니터링)**을 살펴봅니다. 또한, 모델의 성능을 결정하는 **'데이터 중심 AI'**의 중요성과 실습을 통한 E2E 파이프라인 구축 과정을 설명합니다.

Contents

1 공지사항 및 주요 일정

- **과제 4 (Assignment 4):** 곧 공개될 예정이며, MLflow를 기반으로 합니다. 목요일 섹션에서 리뷰할 예정이며, 제출 기한은 약 3주입니다.
- **케이스 스터디 2 (Case Study 2):** 그룹 프로젝트로 진행되며, 추후 공개될 예정입니다. 제출 기한은 약 3주입니다.
  - 케이스 스터디 1과는 다른 새로운 그룹이 배정될 것입니다.
  - 이는 가능한 한 많은 동료 학생들과 교류하고 네트워크를 구축할 기회를 제공하기 위함입니다.
- **기말 프로젝트 (Final Project):** 그룹 프로젝트로 진행되며, 역시 새로운 그룹이 배정됩니다.
- **퀴즈 2 (Quiz 2):** 11월 말경에 공개될 예정입니다.
- **성적 공지:** 퀴즈 1 성적은 마감일로부터 약 2주 이내에 공개될 예정입니다.

2 지난 강의 핵심 복습

MLOps를 본격적으로 배우기 전, 지난 강의에서 다룬 핵심 용어들을 복습합니다. 이 용어들은 MLOps의 기반이 됩니다.

Table 1: *MLOps* 관련 핵심 용어 복습

용어	설명 (초심자 가이드)
모델 드리프트 (Model Drift)	시간이 지남에 따라 모델의 성능이 저하되는 현상입니다. (예: 계절이 바뀌어 작년의 판매 예측 모델이 안 맞게 됨)
MLflow	모델 개발의 전 과정을 관리(추적, 등록, 배포)하는 오픈소스 프레임워크입니다. 모델의 '이력서'를 관리해 준다고 생각할 수 있습니다.
MLflow Tracking	모델 학습 시 사용된 모든 정보(파라미터, 성능 지표, 코드, 데이터 버전 등)를 기록하는 기능입니다.
MLflow Registry	완성된 모델을 등록하고 버전을 관리하는 '모델 카탈로그'입니다. (예: '개발 중', '스테이징', '운영 배포' 상태 관리)
피처 엔지니어링 (Feature Eng.)	원본 데이터(Raw Data)를 모델이 더 잘 학습할 수 있는 유의미한 '특성(Feature)'으로 가공하는 과정입니다. (예: '생년월일'을 '나이'로 변경)
모델 서빙 유형 (Model Serving)	모델을 배포하여 예측값을 제공하는 방식입니다. (1) **배치**: 한 번에 대량 처리. (2) **스트리밍**: 실시간 데이터 흐름 처리. (3) **실시간**: 즉각적인 요청/응답.
과적합 (Overfitting)	모델이 학습 데이터에만 너무 치중하여, 새로운(실제) 데이터에 대한 예측 성능이 떨어지는 현상입니다. (예: 족보만 외워서 응용 문제를 못 푸는 학생)
피처 스토어 (Feature Store)	자주 사용되는 피처(특성)들을 저장하고 재사용할 수 있게 만든 중앙 저장소입니다.
AutoML	모델 생성 과정을 자동화하는 기술입니다. 데이터만 입력하면 AI가 알아서 최적의 모델을 탐색하고 생성해 줍니다.

3 핵심 개념: MLOps란 무엇인가?

3.1 MLOps의 정의와 필요성

전통적인 소프트웨어 개발은 'DevOps(개발+운영)'를 통해 안정적인 배포와 운영을 자동화했습니다. 하지만 머신러닝 시스템은 기존 소프트웨어와 근본적인 차이가 있습니다.

- **전통적 소프트웨어 = 코드 (Code)**
- **AI 시스템 = 코드 (Model/Algorithm) + 데이터 (Data)**

AI 시스템은 코드뿐만 아니라 데이터의 변화에도 민감하게 반응합니다. 따라서 모델을 한 번 만들고 끝나는 것이 아니라, 지속적으로 관리하고 업데이트해야 합니다.

MLOps (Machine Learning Operations)는 머신러닝(ML), 개발(Dev), 운영(Ops)의 합성어입니다. 이는 머신러닝 모델의 개발, 배포, 운영 전 과정을 체계적이고, 신뢰할 수 있으며, 반복 가능하게 만드는 문화와 기술적 관행을 의미합니다.

□ 핵심 요약

MLOps의 목표: 기존 DevOps의 원칙(CI/CD, 자동화, 모니터링)을 머신러닝 파이프라인에 적용하여, 모델이 실제 비즈니스 환경에서 안정적이고 지속적으로 가치를 창출하도록 보장하는 것입니다.

3.2 비즈니스 가치 (예: 고객 이탈 방지)

모든 ML 프로젝트는 비즈니스 문제 해결에서 시작됩니다. MLOps가 어떻게 가치를 창출하는지 **'고객 이탈(Churn) 방지'** 사례를 통해 알아보겠습니다.

- **상황:** 한 마케팅 분석 팀이 고객의 인구통계 및 과거 서비스 데이터를 대시보드로 보고 있습니다.
- **기존 방식 (BI):** "지난 달에 어떤 고객들이 이탈했는가?" (과거 분석)
- **비즈니스 요구 (ML):** "앞으로 어떤 고객이 이탈할 **것 같은가**?" (미래 예측)

왜 예측이 중요한가? 신규 고객을 유치하는 비용보다 기존 고객을 유지하는 비용이 훨씬 저렴합니다. 고객이 떠나기 *전에* 예측할 수 있다면, 할인 쿠폰이나 인센티브를 제공하는 등 선제적인 조치를 취해 고객 이탈을 막고 수익을 보존할 수 있습니다.

MLOps는 이 '이탈 예측 모델'을 개발하고, 실제 운영 환경에 배포하며, 모델 성능이 떨어지지 않도록 지속적으로 모니터링하는 전 과정을 관리합니다.

4 MLOps의 핵심 구성원 (The "People")

성공적인 MLOps는 다양한 역할의 전문가들이 협업하는 '프로세스'입니다. 각 구성원의 역할을 이해하는 것이 중요합니다.

4.1 비즈니스 이해관계자 (Business Stakeholder)

- **역할:** ML 솔루션의 비즈니스 가치를 책임집니다.
- **핵심 임무:**
  - **문제 정의:** 모호한 비즈니스 목표를 명확한 데이터 문제로 번역합니다.
    - (예: "고객 이탈"이란 정확히 무엇인가? → "90일 내 재구매가 없거나, 1개월 내 구독을 취소한 고객")
  - **KPI 설정:** 모델의 성공을 측정할 핵심 성과 지표(KPI)를 결정합니다.
    - (예: 이탈률 감소, 유지율 증가, 개입(할인) 대비 투자 수익률(ROI))
  - **결과 검증:** 배포된 모델이 실제로 비즈니스 목표 달성에 기여하는지(예: 이탈률 감소) 확인하고 분석합니다.

4.2 데이터 엔지니어 (Data Engineer)

- **역할:** 데이터 파이프라인을 구축합니다.
- **핵심 임무:**
  - **데이터 수집:** 다양한 소스(CRM, 웹 로그, 결제 시스템 등)에서 데이터를 수집합니다.
  - **데이터 처리 (ETL/ELT):** 데이터를 추출(Extract), 변환(Transform), 적재(Load)하여 분석 가능한 형태로 가공합니다.
  - **품질 보증:** 누락된 값, 중복 데이터, 불일치 등을 처리하여 데이터의 신뢰성을 보장합니다.
  - **인프라 관리:** 데이터가 저장되고 접근될 수 있는 확장 가능한 인프라(예: AWS, Azure)를 관리합니다.
- **최종 산출물:** 분석 및 모델링에 즉시 사용 가능한 **"신뢰할 수 있는 프로덕션 등급 데이터셋"**

4.3 데이터 사이언티스트 (Data Scientist)

- **역할:** 데이터를 예측 가능한 '인사이트'로 변환합니다.
- **핵심 임무:**
  - **탐색적 데이터 분석 (EDA):** 데이터를 탐색하며 패턴을 파악합니다.
  - **피처 엔지니어링:** 이탈에 영향을 미치는 변수(피처)를 식별하고 생성합니다. (예: 사용 빈도, 마지막 구매일, 불만 접수 횟수)
  - **모델 선택 및 학습:** 다양한 알고리즘(예: 로지스틱 회귀, 랜덤 포레스트, XGBoost)을 실험하고 최적의 모델을 학습시킵니다.
  - **성능 평가:** 정확도, 정밀도 등 적절한 지표로 모델 성능을 검증합니다.
  - **결과 해석:** 어떤 요인이 이탈에 가장 큰 영향을 미치는지 해석하고 비즈니스팀과 협업합니다.

4.4 머신러닝 엔지니어 (ML Engineer)

- **역할:** 데이터 사이언티스트가 만든 모델(프로토타입)을 실제 '제품(Production)'으로 만들고 유지보수합니다.
- **핵심 임무:**
  - **모델 배포:** 학습된 모델을 API 엔드포인트나 배치 스코어링 파이프라인으로 배포합니다.
  - **CI/CD 구축:** 모델의 재학습, 테스트, 버전 관리, 배포 과정을 자동화합니다.
  - **모니터링:** 모델 성능(정확도), 데이터 드리프트, 지연 시간(latency), 편향성(bias) 등을 지속적으로 추적합니다.
  - **확장성 보장:** 트래픽이 증가해도 안정적으로 예측 서비스를 제공할 수 있도록 인프라를 최적화합니다.

4.5 데이터 거버넌스 책임자 (Data Governance Officer)

- **역할:** 데이터 거버넌스 및 규정 준수(Compliance)를 책임집니다.
- **핵심 임무:**
  - 전체 파이프라인에 걸쳐 데이터 보안, 개인정보 보호(익명화), 접근 제어가 올바르게 적용되는지 감독합니다.

□ 핵심 요약

역할별 ML 파이프라인 참여도 요약

프로세스의 각 단계는 여러 역할의 협업으로 이루어집니다.

- **1. 데이터 준비 (Data Preparation):**
  - 주도: **데이터 엔지니어**
  - 감독: 데이터 거버넌스
- **2. 탐색적 데이터 분석 (EDA):**
  - 주도: **데이터 사이언티스트**
  - 감독: 데이터 거버넌스
- **3. 피처 엔지니어링 (Feature Engineering):**
  - 주도: **데이터 사이언티스트**
  - 감독: 데이터 거버넌스
- **4. 모델 학습 (Model Training):**
  - 주도: **데이터 사이언티스트**
  - 감독: 데이터 거버넌스
- **5. 모델 검증 (Model Validation):**
  - 협업: **데이터 사이언티스트, ML 엔지니어, 비즈니스 이해관계자**
  - 감독: 데이터 거버넌스
- **6. 배포 (Deployment):**
  - 주도: **ML 엔지니어**
  - 협업: 비즈니스 이해관계자
  - 감독: 데이터 거버넌스

- **7. 모니터링 (Monitoring):**
  - 주도: **ML 엔지니어**
  - 협업: 비즈니스 이해관계자
  - 감독: 데이터 거버넌스

5 MLOps의 3대 파이프라인 (The 3 Pipelines)

MLOps는 크게 세 가지의 상호 연결된 파이프라인으로 구성됩니다.

5.1 1. 모델 학습 파이프라인 (Model Training Pipeline)

- **목표:** 원본 데이터(Raw Data)로부터 잘 일반화된(과적합되지 않은) 모델 아티팩트(파일)를 생성합니다.
- **주요 단계:**
  1. **데이터 준비:** 데이터를 학습(Train) / 검증(Validation) / 테스트(Test) 세트로 분할합니다.
  2. **피처 엔지니어링:** 원본 변수를 모델에 유의미한 입력값으로 변환합니다. (피처 스토어에서 가져오거나 새로 생성)
  3. **모델 학습:** 적절한 알고리즘을 선택하고 학습 데이터로 모델을 훈련시킵니다.
  4. **파라미터 튜닝:** 모델의 정확도를 높이고 과적합을 줄이기 위해 하이퍼파라미터를 최적화합니다.
  5. **평가:** 검증 세트를 사용해 정확도, 정밀도(Precision), 재현율(Recall), F1 점수 등 다양한 지표로 모델 성능을 평가합니다.
  6. **아티팩트 저장:** 성능이 좋은 모델을 재사용 및 배포할 수 있도록 파일(아티팩트)로 저장하고 MLflow 같은 도구에 기록합니다.

5.2 2. 모델 추론 파이프라인 (Model Inferencing Pipeline)

- **목표:** 학습된 모델을 사용하여 새로운 데이터에 대한 예측(결정)을 생성하고, 비즈니스 의사결정을 지원합니다.
- **주요 배포 방식:**
  - **배치 추론 (Batch Inferencing):**
    - **작동 방식:** 주기적으로(예: 매일 밤) 대량의 데이터를 한 번에 처리합니다.
    - **예시:** "다음 주에 이탈할 가능성이 있는 고객" 목록을 미리 생성하여 마케팅 팀에 전달합니다.
  - **실시간 추론 (Real-time Inferencing):**
    - **작동 방식:** 사용자의 요청이 올 때마다 즉각적으로 예측을 반환합니다. (보통 API 형태)
    - **예시:** 고객이 웹사이트에서 '구독 취소' 버튼 근처에 마우스를 가져갈 때, 실시간으로 이탈 점수를 계산하여 할인 쿠폰 팝업을 띄웁니다.
- **주요 고려사항:** 성능(지연 시간), 처리량(Throughput), 확장성, 보안.

5.3 3. 모델 모니터링 파이프라인 (Model Monitoring Pipeline)

- **목표:** 운영 환경(Production)에서 모델의 동작과 성능을 지속적으로 추적하여 문제(성능 저하)를 감지합니다.
- **주요 모니터링 대상 (드리프트):**
  - **데이터 드리프트 (Data Drift):**
    - **정의:** 모델에 입력되는 데이터의 통계적 분포가 시간이 지남에 따라 변하는 현상입니다.
    - **예시:** 새로운 마케팅 캠페인으로 인해 이전과 다른 연령대의 신규 고객이 대거 유입되는 경우.
  - **개념 드리프트 (Concept Drift):**

    - ∗ **정의:** 입력 데이터와 목표 변수(예측 대상) 간의 관계 자체가 변하는 현상입니다. 데이터 분포는 그대로일 수 있습니다.
    - ∗ **예시:** (1) 새로운 경쟁사 출시로 인해 고객이 '가격'보다 '서비스 품질'을 이탈의 주요 원인으로 삼기 시작하는 경우. (2) COVID-19로 인해 사람들의 소비 패턴이 근본적으로 바뀐 경우.
  - **모델 품질 저하:** 실제 결과(Ground Truth)와 모델의 예측값을 비교하여 정확도, F1 점수 등이 설정한 임계값 이하로 떨어지는지 확인합니다.
- **조치:** 드리프트나 성능 저하가 감지되면, **경고(Alert)**를 보내거나 **자동으로 모델 재학습(Retrain) 파이프라인을 실행**합니다.

□ 예제:

Q: 데이터 드리프트와 모델 드리프트(성능 저하)는 어떻게 자동으로 감지하나요?

A: 주로 자동화된 통계적 품질 검사를 통해 감지합니다.

- **데이터 드리프트 감지:** 모델을 학습시킬 때 사용했던 데이터(베이스라인)의 통계적 특성(평균, 분산, 분포 등)을 저장해 둡니다. 그리고 실제 운영 환경에 새로 들어오는 데이터의 통계적 특성을 이 베이스라인과 주기적으로 비교합니다. 두 분포 간의 차이가 특정 임계값을 넘으면 드리프트로 간주합니다.
- **모델 드리프트 (성능 저하) 감지:** 모델의 예측 정확도를 모니터링하는 것이 가장 확실한 방법입니다. (예: 모델이 "이탈 안 함"으로 예측했는데, 실제로는 이탈한 고객의 비율이 점점 증가하는지 추적)
- **개념 드리프트 감지:** 가장 감지하기 어렵습니다. 데이터 드리프트나 모델 성능 저하가 뚜렷하지 않은데도 비즈니스 KPI(예: 할인 쿠폰 성공률)가 하락한다면, 비즈니스 맥락 자체가 변했을(개념 드리프트) 가능성을 의심하고 수동으로 분석해야 할 수 있습니다.

6 모델 배포 전략 (ML Deployment Patterns)

모델을 개발(dev) 환경에서 스테이징(staging), 운영(prod) 환경으로 승격시킬 때 두 가지 주요 전략이 있습니다.

6.1 1. 모델 배포 (Deploy Models)

- **방식:** 개발(dev) 환경에서 모델을 학습시키고, 생성된 모델 **아티팩트(파일)** 자체를 스테이징, 운영 환경으로 복사하여 배포합니다.
- **흐름:** [Dev에서 학습] → [모델 파일 생성] → [파일을 Staging으로 복사] → [파일을 Prod로 복사]

6.2 2. 코드 배포 (Deploy Code)

- **방식:** 개발(dev) 환경에서 모델 **학습 코드**를 개발하고, 이 코드를 스테이징, 운영 환경으로 복사(배포)합니다. 그 다음, **각 환경에서 해당 환경의 데이터로 모델을 다시 학습**시킵니다.
- **흐름:** [Dev에서 코드 개발] → [코드를 Staging으로 복사] → [Staging 데이터로 재학습] → [코드를 Prod로 복사] → [Prod 데이터로 재학습]

6.3 코드 배포의 장단점 (Benefits of Deploy Code)

'코드 배포' 방식은 더 복잡해 보이지만, MLOps 관점에서 많은 이점을 제공합니다.

Table 2: '코드 배포' 전략의 주요 장점

장점	설명
데이터 접근 제어	**가장 큰 장점.** 민감한 운영(Prod) 데이터를 비-운영(Dev, Staging) 환경으로 가져올 필요가 없습니다. 각 환경은 자신의 데이터에만 접근하여 모델을 학습합니다.
재현성 높은 모델	엔지니어링팀이 각 환경(Dev, Staging, Prod)의 학습 환경(라이브러리, 의존성 등)을 완벽하게 제어할 수 있어 모델의 재현성을 단순화하는 데 도움이 됩니다.
대규모 프로젝트 지원	이 패턴은 데이터 사이언스팀이 모듈화된 코드를 작성하고 반복적인 테스트를 수행하도록 강제합니다. 이는 대규모 프로젝트에서 협업과 개발을 용이하게 합니다.
실제 데이터 문제 처리	개발 환경의 샘플 데이터에서는 발생하지 않던 실제 운영 데이터의 편향(skew)이나 대용량으로 인한 문제를 운영 환경에서 직접 학습하며 처리할 수 있습니다. (예: Staging에서는 잘 동작했으나 Prod의 대용량 데이터 패턴으로 인해 실패하는 문제 방지)

주의사항

'코드 배포'의 단점

- **더 많은 컴퓨팅 비용:** 각 환경에서 모델을 재학습해야 하므로 컴퓨팅 자원이 더 많이 소모됩니다.
- **인프라 요구사항:** 일회성 모델이라도 단위 테스트, 통합 테스트를 위한 CI/CD 인프라가 필요합니다.
- **요구 역량:** 데이터 사이언티스트가 단순히 모델 파일만 전달하는 것이 아니라, 엔지니어링팀이 운영할 수 있는 모듈화된 코드를 작성하고 핸드오프(handoff)해야 합니다.

7 데이터 중심 AI (Data-Centric AI)

모델의 성능을 높이는 방법은 크게 두 가지입니다.

1. **모델 중심 AI (Model-Centric AI):** "더 좋은 모델(코드/알고리즘)을 만들어서 성능을 높일 수 없을까?"
2. **데이터 중심 AI (Data-Centric AI):** "데이터(입력 또는 레이블)를 체계적으로 개선해서 성능을 높일 수 없을까?"

과거에는 모델 중심 접근법이 주를 이루었지만, 최근 MLOps의 핵심 철학은 **데이터 중심 AI**입니다.

7.1 빅데이터 vs. 좋은 데이터 (Big Data vs. Good Data)

무조건 데이터가 많다고(Big Data) 좋은 모델이 나오는 것은 아닙니다. 품질이 낮은 대량의 데이터는 오히려 모델 성능에 해가 될 수 있습니다. 중요한 것은 **"좋은 데이터(Good Data)"**입니다.

좋은 데이터란?

- **명확한 레이블 (Unambiguous labels):** 정답(레이블)이 일관되고 명확해야 합니다.
- **중요 사용 사례 포괄 (Good cover):** 예측해야 할 중요한 케이스들을 충분히 포함해야 합니다. (예: 모든 주(State)를 예측해야 하는데, 매사추세츠주 데이터만 사용하면 안 됨)
- **시기적절한 피드백 (Timely feedback):** 운영 환경에서 발생한 최신 데이터를 빠르게 피드백 받을 수 있어야 합니다.
- **적절한 크기 (Sized appropriately):** 너무 적어도 안 되지만, 불필요하게 많을 필요도 없습니다.

7.2 왜 데이터 중심 AI가 중요한가?

AI에게 데이터는 '음식'과 같습니다.

□ **예제:**

비유: 데이터는 요리의 재료다
최고의 셰프(모델/알고리즘)라 할지라도, 상한 재료(나쁜 데이터)로는 맛있는 요리(좋은 예측)를 만들 수 없습니다.

- **모델 중심 접근:** 이미 좋은 재료가 있을 때, 레시피를 미세 조정하는 것 (예: 15분 끓일 것을 16분 끓이기). → 성능 향상에 한계가 있음.
- **데이터 중심 접근:** 레시피는 그대로 두고, 더 신선하고 고품질의 재료(데이터)를 공수해 오는 것. → 요리(모델)의 품질을 극적으로 향상시킬 수 있음.

실제 산업 사례(강철 결함 감지 등)에서도, 모델 코드를 수정하는 것(Model-centric)보다 데이터 품질을 개선하는 것(Data-centric)이 **월등히 높은(예: +16.9%) 성능 향상**을 보였습니다. MLOps는 이처럼 고품질의 데이터를 지속적으로 확보하고 개선하는 프로세스를 자동화하는 데 중점을 둡니다.

8 MLOps 성숙도 모델 (MLOps Progression)

모든 조직이 처음부터 완벽한 MLOps를 갖추는 것은 아닙니다. MLOps는 점진적으로 발전하는 '성숙도' 단계를 가집니다.

- **Level 1: Beginner (초급)**
  - 익숙한 도구를 사용하여 생산성을 높이고, 더 복잡한 작업을 계획하는 단계.
  - (예: 데이터 사이언티스트가 수동으로 모델을 학습하고 파일을 전달함)
- **Level 2: Intermediate (중급)**
  - 데이터 및 워크로드를 확장(Scale)합니다.
  - **자동화**와 **재현성**에 중점을 둡니다.
  - 흩어져 있던 데이터 팀을 통합하고 협업을 개선합니다.
- **Level 3: Advanced (고급)**
  - **더 빠른 생산 주기:** CI/CD가 완전히 자동화되어 한 달에 한 번이 아닌, 하루에도 여러 번 배포가 가능합니다.
  - **E2E 자동화 및 재현성:** 여러 워크로드에 걸쳐 엔드투엔드(End-to-End) 자동화가 이루어집니다.
  - **회복 탄력성:** 넷플릭스(Netflix)의 '카오스 몽키(Chaos Monkey)'처럼 의도적으로 프로덕션에 이슈를 발생시켜 시스템이 얼마나 빨리 스스로 복구하는지 테스트합니다.
  - **롤백(Rollback) 메커니즘:** 배포에 문제가 생겼을 때 즉시 이전 버전으로 되돌리는 기능이 완비되어 있습니다.

9 모델 튜닝 및 선택 (Model Tuning and Selection)

모델 학습 파이프라인에서 '좋은 모델'을 만들기 위한 핵심 기술입니다.

9.1 하이퍼파라미터 튜닝 (Hyperparameter Tuning)

- **목표:** 모델이 스스로 학습하지 않는 값, 즉 개발자가 직접 설정해야 하는 '하이퍼파라미터'(예: 트리의 깊이, 트리의 개수)를 최적화하여 과적합/과소적합을 방지하고 모델 성능을 극대화합니다.
- **주요 방식:**
  - **수동 (Grid Search):**
    - **작동 방식:** 개발자가 지정한 하이퍼파라미터 값의 모든 조합을 시도합니다. (예: [깊이: 3, 5], [트리 개수: 100, 200] → (3,100), (3,200), (5,100), (5,200) 총 4번 학습)
    - **특징:** 중첩된 for문과 같아서 파라미터가 많아지면 매우 비효율적이고 비쌉니다.
  - **자동 (Automated):**
    - **작동 방식:** `Optuna`, `Hyperopt` 같은 라이브러리를 사용합니다.
    - **종류:**
      - **Random Search:** 무작위로 조합을 탐색 (Grid Search보다 효율적일 때가 많음)
      - **Bayesian Search:** 이전 시도 결과를 바탕으로 다음 시도에 더 성능이 좋을 만한 영역을 지능적으로 탐색합니다.
      - **Genetic Search:** 좋은 모델과 나쁜 모델의 파라미터를 유전 알고리즘처럼 조합하여 더 나은 모델을 탐색합니다. (딥러닝에서 유용)

9.2 K-겹 교차 검증 (K-Fold Cross Validation)

- **목표:** 모델을 평가할 때 '테스트 세트'를 사용하면, 모델이 해당 테스트 세트에만 과적합될 위험이 있습니다. (일종의 '치팅') 이를 방지하기 위해 학습 데이터 내부에서만 모델을 검증하는 기법입니다.
- **작동 방식 (예: K=3인 경우):**
  1. 학습 데이터(Train + Validation)를 3개의 '폴드(Fold)'로 나눕니다. [Fold 1], [Fold 2], [Fold 3]
  2. **Pass 1:** [Fold 1, 2]로 학습 → [Fold 3]로 검증
  3. **Pass 2:** [Fold 1, 3]로 학습 → [Fold 2]로 검증
  4. **Pass 3:** [Fold 2, 3]로 학습 → [Fold 1]로 검증
  5. 3개 검증 점수의 **평균**을 내어 해당 하이퍼파라미터의 최종 성능으로 삼습니다.
- **장점:** 학습 데이터를 최대한 활용하면서도, 테스트 세트(Holdout set)를 오염시키지 않고 모델의 일반화 성능을 안정적으로 평가할 수 있습니다.

10 실습: E2E MLOps 파이프라인 (Lab Walkthrough)

Databricks 환경에서 고객 이탈 예측 모델을 E2E(End-to-End) MLOps 파이프라인으로 구축하는 과정을 요약합니다.

10.1 실습 목표 및 도구

- **목표:** 고객 이탈(Churn)을 예측하는 LightGBM 모델을 MLOps 수명 주기에 맞춰 개발, 등록, 검증, 배포합니다.
- **주요 도구:**
  - **Delta Lake:** 데이터의 저장, 버전 관리, 트랜잭션 보장
  - **MLflow:** 모델 수명 주기 관리 (Tracking, Registry)
  - **Unity Catalog (UC):** 중앙화된 데이터 및 모델 거버넌스 (카탈로그, 스키마, 모델 관리)
  - **LightGBM:** 머신러닝 알고리즘

10.2 1단계: 데이터 준비 및 피처 엔지니어링

1. 원본 데이터(`mlops_churn_bronze_customer`)를 읽어옵니다.
2. EDA를 수행합니다. (예: Pandas API on Spark를 사용해 인터넷 서비스 유형별 고객 수 계산)
3. 데이터를 정제하고 새로운 피처를 생성합니다.
   - (예: '온라인 보안', '온라인 백업' 등 부가 서비스 가입 여부를 합산하여 `num_optional_services` 피처 생성)
4. 데이터를 학습(Train) 및 추론(Inference)용으로 분할하여 Delta Table(`mlops_churn_training`)로 저장합니다.

10.3 2단계: 모델 학습 및 로깅 (MLflow Tracking)

1. MLflow Experiment(실험실)를 설정합니다.
2. **데이터 리니지(Lineage) 로깅 (핵심):**
   - 모델을 학습시키기 전에, 어떤 데이터 테이블의 **어떤 버전**을 사용했는지 MLflow에 명시적으로 기록합니다.
   - `mlflow.data.load_delta(...)` → `mlflow.log_input(dataset, ...)`
   - **이유:** 모델의 재현성을 위해 "어떤 코드가 어떤 데이터로 만들었는지" 추적하는 것은 필수입니다. (이때 데이터 복사본이 아닌 메타데이터만 저장됩니다.)
3. 데이터 전처리 파이프라인을 정의합니다. (Boolean, Numerical, Categorical 피처별 변환기)
4. LightGBM 분류기(Classifier)를 포함한 전체 학습 파이프라인을 정의합니다.
5. `mlflow.autolog()`를 활성화하여 파라미터, 지표, 아티팩트를 자동으로 로깅합니다.
6. 모델을 학습(`.fit()`)시키고 MLflow에 모델을 로깅(`mlflow.log_model()`)합니다.
7. **결과:** MLflow 실행(Run) 페이지에 모든 정보(파라미터, F1 점수 등)와 아티팩트(ROC 커브, 혼동 행렬, `requirements.txt`, 모델 파일 등)가 기록됩니다.

10.4 3단계: 모델 레지스트리 등록 (MLflow Registry)

1. 2단계에서 실행된 수많은 실험(Run) 중에서 **최고의 모델**(예: F1 점수가 가장 높은 모델)을 검색합니다.
2. `mlflow.register_model()`을 사용하여 이 모델을 Unity Catalog 내의 모델 레지스트리(예: `cscie103_catalog.m`에 등록합니다.
3. 등록된 모델은 **'version 1'**을 받습니다.
4. **모델 리니지 확인:** UC의 모델 레지스트리 UI에서, 이 모델 버전(v1)이 **어떤 데이터셋**과 **어떤 노트북(코드)**을 통해 생성되었는지 업스트림(Upstream) 리니지를 시각적으로 확인할 수 있습니다.
5. 모델과 버전에 설명(Description)과 태그(Tag)를 추가합니다. (예: '$F1_{score} = 0.85$')
5. 이 신규 등록 모델에 **'Challenger' (도전자)**라는 별칭(Alias)을 설정합니다.

10.5 4단계: 챌린저 모델 검증 (Champion vs. Challenger)

1. **자동화된 품질 검사:** 'Challenger' 별칭으로 모델을 로드한 뒤, 자동화된 테스트를 수행합니다. (예: 모델 설명이 비어있지 않은가? F1 점수가 최소 기준치(0.8) 이상인가?)
2. **챔피언-챌린저 대결:**
   - 현재 운영 환경에 배포된 모델(별칭: **'Champion'**)을 로드합니다. (try-except 구문 사용. 챔피언이 없으면(첫 배포) Challenger가 자동 승리)
   - 'Challenger' 모델의 F1 점수와 'Champion' 모델의 F1 점수를 비교합니다.
3. **승격 (Promotion):**
   - 만약 'Challenger'가 더 우수하다면, 'Challenger' 모델의 별칭을 **'Champion'**으로 설정(승격)합니다.
   - (기존 챔피언은 'Archived' 상태가 되거나 별칭이 제거됩니다.)

10.6 5단계: 배치 추론 (Batch Inference)

1. **'Champion'** 별칭으로 현재 운영 모델을 로드합니다.
2. 1단계에서 분리해둔 새로운 데이터(추론용 데이터)를 로드합니다.
3. `mlflow.pyfunc.spark_udf`를 사용하여 Spark 데이터프레임에 챔피언 모델을 적용합니다.
4. **결과:** 원본 데이터에 **'predictions' 열** (예: 1(이탈), 0(유지))이 추가된 최종 결과 테이블을 얻습니다. 이 테이블은 마케팅팀의 대시보드나 액션 시스템으로 전달됩니다.

11 핵심 용어 정리

이 강의에서 다룬 MLOps의 핵심 용어를 표로 정리합니다.

Table 3: *MLOps* 핵심 용어집

용어	설명 (초심자 가이드)
MLOps	머신러닝(ML), 개발(Dev), 운영(Ops)의 합성어. ML 시스템을 안정적이고 반복적으로 개발, 배포, 운영하기 위한 문화 및 기술.
데이터 드리프트 (Data Drift)	모델에 입력되는 실제 데이터의 통계적 분포가 학습 당시의 데이터 분포와 달라지는 현상. (예: 신규 고객 연령층 변화)
개념 드리프트 (Concept Drift)	데이터와 정답 간의 '관계' 자체가 변하는 현상. (예: 가격 때문에 이탈하던 고객들이 이제는 서비스 품질 때문에 이탈)
CI/CD	지속적 통합(Continuous Integration) / 지속적 배포(Continuous Deployment). 코드 변경사항을 자동으로 테스트하고 운영 환경에 배포하는 자동화 프로세스.
모델 아티팩트 (Model Artifact)	모델 학습의 결과물. 모델 가중치가 저장된 파일(예: `pickle` 파일), `conda.yaml`, `requirements.txt` 등을 포함하는 패키지.
모델 레지스트리 (Model Registry)	학습된 모델 아티팩트를 저장, 버전 관리, 상태(Staging, Prod) 추적하는 중앙 카탈로그. (예: MLflow Registry)
리니지 (Lineage)	'혈통' 또는 '계보'. 데이터나 모델이 어떤 원본(데이터, 코드)으로부터 어떻게 변환되어 생성되었는지 그 이력을 추적하는 것.
하이퍼파라미터 (Hyperparameter)	모델이 학습 과정에서 스스로 찾는 값(Parameter)이 아니라, 개발자가 사전에 직접 설정해야 하는 값. (예: 학습률, 트리 깊이)
Grid Search	하이퍼파라미터 튜닝 기법 중 하나. 개발자가 지정한 모든 파라미터 조합을 시도하는 '격자 탐색' 방식.
K-겹 교차 검증 (K-Fold CV)	학습 데이터를 K개로 나누어, K-1개로 학습하고 1개로 검증하는 과정을 K번 반복하여 모델 성능을 평균내는 검증 기법.
챔피언/챌린저 (Champion/Challenger)	모델 배포 전략. 현재 운영 중인 최고 성능 모델(**Champion**)과 새로 개발된 모델(**Challenger**)의 성능을 비교하여, 챌린저가 더 우수할 경우 챔피언을 교체하는 방식.
데이터 중심 AI (Data-Centric AI)	모델 알고리즘 개선보다 고품질의 데이터를 확보하고 개선하는 데 집중하여 AI 성능을 높이려는 접근 방식.

12 빠르게 훑어보기 (1-Page Summary)

☐ MLOps 핵심 요약 카드

1. MLOps란?

ML(머신러닝) + Dev(개발) + Ops(운영)의 합성어.
AI 시스템(Code + **Data**)을 **신뢰할 수 있고 반복 가능**하게 개발, 배포, 모니터링하는 전 과정.

2. MLOps 핵심 4대 역할

- **데이터 엔지니어(DE):** 데이터 파이프라인 구축 (신뢰 가능한 데이터 제공)
- **데이터 사이언티스트(DS):** 모델 개발 (데이터 → 인사이트)
- **ML 엔지니어(MLE):** 모델 배포 및 운영 (프로토타입 → 프로덕션)
- **비즈니스 이해관계자:** 문제 정의 및 KPI 설정

3. 3대 핵심 파이프라인

- **학습(Training):** 데이터 → 모델 아티팩트 생성
- **추론(Inferencing):** 모델 → 예측값 생성 (Batch vs. Real-time)
- **모니터링(Monitoring):** 모델 성능 추적 및 드리프트 감지

4. 드리프트(Drift) 3종 세트

- **데이터 드리프트:** 입력 데이터의 분포 변경
- **개념 드리프트:** 데이터-정답 간의 '관계' 변경
- **모델 드리프트:** 위 둘로 인한 모델 성능 저하

5. 데이터 중심 AI (Data-Centric AI)

AI 성능 향상은 '모델 코드' 개선보다 '데이터 품질' 개선이 더 효과적이다. (데이터는 AI의 '음식 재료'와 같다)
→ MLOps는 고품질 데이터를 지속 공급/관리하는 프로세스.

6. 배포 전략 (Code vs. Model)

- **모델 배포:** 학습된 '모델 파일'을 Prod 환경에 복사.
- **코드 배포 (권장):** '학습 코드'를 Prod 환경에 복사 후, Prod 데이터로 '재학습'. (데이터 접근 제어 및 재현성에 유리)