December 10, 2025

▣ **강의명:** CSCI E-103: 재현 가능한 머신러닝

▣ **주차:** Lecture 10

▣ **교수명:** Anindita Mahapatra

Eric Gieseke

▣ **목적:** Lecture 10의 핵심 개념 학습

□ 핵심 요약

이 문서는 10번째 강의인 '데이터 및 머신러닝 문제 다루기'의 핵심 내용을 요약 및 보완하여 정리한 노트입니다. 데이터 문제가 모델 문제로 이어지는 과정을 이해하고, 특히 **모델 편향(Model Bias)**, **데이터 불균형(Class Imbalance)** 문제를 중점적으로 다룹니다.
또한, **AutoML**의 개념과 'Glassbox ML' 접근법을 살펴보고, 마지막으로 이 모든 개념이 통합된 실제 대규모 **실시간 사기 탐지(Fraud Detection) 시스템** 아키텍처 사례를 심층적으로 분석합니다.

Contents

1 개요: 데이터에서 모델까지

이번 강의의 핵심 주제는 "데이터 문제가 곧 모델 문제"라는 것입니다. 머신러닝 모델의 성능은 결국 우리가 제공하는 데이터의 품질에 달려있습니다.

강의 핵심 요약

- **모델 오류의 이해:** 모델의 총 오류는 편향(Bias), 분산(Variance), 그리고 줄일 수 없는 오류(Irreducible Error)로 구성됩니다. 이 중 편향과 분산은 트레이드오프 관계에 있습니다.
- **모델 편향 문제:** 모델이 특정 집단에게 불공정한 예측을 하는 현상입니다. 이는 주로 훈련 데이터의 부족(Under-representation)이나 인간의 인지 편향이 데이터에 반영되어 발생합니다.
- **데이터 불균형 문제:** 사기 탐지나 질병 진단처럼, 한 클래스가 극도로 드문(예: 99% vs 1%) 상황입니다. 이때 '정확도(Accuracy)'는 무의미하며, F1-Score, 재현율(Recall) 등 다른 평가지표와 SMOTE 같은 리샘플링 기법이 필요합니다.
- **AutoML과 Glassbox:** AutoML은 모델 개발을 자동화하지만, 내부를 볼 수 없는 'Blackbox'는 한계가 있습니다. Databricks 등이 지향하는 'Glassbox ML'은 EDA, 하이퍼파라미터 튜닝 등 모든 과정을 노트북으로 투명하게 제공합니다.
- **실전 사례 (사기 탐지):** 15ms의 응답 속도와 초당 5만 건의 트래픽을 처리하는 실시간 사기 탐지 시스템을 구축하기 위해, **람다(Lambda) 아키텍처**, **순환 버퍼(Circular Buffer)**, **Cassandra** 저장소 등 고성능 엔지니어링 기술이 어떻게 적용되었는지 살펴봅니다.

2 주요 용어 정리

강의에서 다룬 핵심 용어들을 미리 정리합니다.

용어	설명 (초심자용 풀이)	원어 / 관련 용어
AutoML	사용자가 데이터를 입력하면, 모델 선택, 피처 엔지니어링, 하이퍼파라미터 튜닝 등을 자동으로 수행하여 최적의 모델을 제안하는 기술입니다.	Automated ML
Glassbox ML	AutoML의 한 종류로, 모델이 만들어지는 전 과정을 투명하게 공개하는 접근법입니다. (예: Databricks AutoML) 내부를 볼 수 없는 'Blackbox'와 대비됩니다.	Glassbox ML vs Blackbox ML
편향 (Bias)	모델이 실제 데이터의 복잡한 관계를 제대로 파악하지 못해, 예측값이 정답과 체계적으로 벗어나는 경향입니다. **과소적합(Underfitting)**의 주요 원인입니다.	Bias
분산 (Variance)	모델이 훈련 데이터의 노이즈까지 너무 복잡하게 학습하여, 새로운 데이터(테스트 데이터)에 대한 예측이 불안정하게 흔들리는 정도입니다. **과대적합(Overfitting)**의 주요 원인입니다.	Variance
편향-분산 트레이드오프	편향을 낮추려 하면(모델이 복잡해지면) 분산이 높아지고, 분산을 낮추려 하면(모델이 단순해지면) 편향이 높아지는, 두 오류가 반비례하는 관계입니다.	Bias-Variance Trade-off
데이터 불균형	훈련 데이터에서 클래스(정답) 간의 비율이 크게 차이 나는 상황입니다. (예: 정상 99%, 사기 1%)	Class Imbalance / Imbalanced Data
정밀도 (Precision)	모델이 "Yes"라고 예측한 것 중, 실제로 "Yes"인 것의 비율입니다. (TP / (TP+FP)) "스팸"이라고 예측한 메일 중 실제 스팸 비율.	Precision
재현율 (Recall)	실제 "Yes"인 것들 중, 모델이 "Yes"라고 예측해낸 비율입니다. (TP / (TP+FN)) 실제 "암 환자" 중 몇 명을 "암"이라고 맞혔는가. (민감도, Sensitivity)	Recall / Sensitivity / True Positive Rate
F1 Score	정밀도와 재현율의 조화 평균입니다. 불균형 데이터에서 정확도(Accuracy) 대신 사용하는 핵심 평가지표입니다.	F1 Score
리샘플링 (Resampling)	불균형 데이터를 처리하기 위해 데이터셋을 조작하는 기법입니다. 소수 클래스를 늘리거나(Oversampling) 다수 클래스를 줄입니다(Undersampling).	Resampling

용어	설명 (초심자용 풀이)	원어 / 관련 용어
SMOTE	대표적인 오버샘플링 기법. 소수 클래스 데이터를 단순히 복제하는 것이 아니라, 기존 데이터들 사이에 '합성(Synthetic)' 데이터를 생성하여 추가합니다.	Synthetic Minority Oversampling Technique
PII	개인 식별 정보. 이름, 전화번호, 주민등록번호, 주소 등 개인을 특정할 수 있는 민감 정보입니다. 모델 학습 전에 반드시 제거하거나 익명화해야 합니다.	Personally Identifiable Information
람다 아키텍처	대용량 데이터를 처리하기 위한 하이브리드 아키텍처. 모든 데이터를 처리하는 **배치(Batch) 레이어**와 실시간 데이터를 처리하는 **스피드(Speed) 레이어**로 구성됩니다.	Lambda Architecture
CEP	실시간 스트리밍 환경에서 발생하는 이벤트들의 패턴을 즉각적으로 감지하고 대응하는 기술입니다. (예: 1초 이내 5회 이상 결제 시도 감지)	Complex Event Processing
순환 버퍼	고정된 크기의 메모리 공간을 순환하면서 사용하는 데이터 구조. 사기 탐지 시스템에서는 '최근 7일간의 거래 합계' 등을 매우 빠르고 효율적으로 계산하기 위해 사용되었습니다.	Circular Buffer / Ring Buffer
Cassandra	대용량의 비정형 데이터를 처리하기 위한 NoSQL 데이터베이스. 특히 쓰기(Write) 성능이 매우 빠르며, 사기 탐지 시스템에서 피처 스토어(Feature Store)로 사용되었습니다.	Apache Cassandra

3 AutoML: 모델 개발의 자동화

머신러닝 프로젝트의 성공은 좋은 모델을 선택하고, 데이터를 적절히 전처리하며, 최적의 하이퍼파라미터를 찾는 데 달려있습니다. 이 과정은 시간과 전문 지식이 많이 필요합니다.

AutoML(Automated Machine Learning)은 이 복잡한 과정을 자동화하는 기술입니다.

3.1 Blackbox vs Glassbox

- **Blackbox AutoML (블랙박스):**
  - 데이터만 넣으면 클릭 몇 번으로 최적의 모델을 '결과물'로만 제공합니다.
  - (예: Data Robot)
  - **단점:** 왜 이 모델이 선택되었는지, 내부적으로 어떤 피처가 중요했는지 알기 어렵습니다. 기업 환경에서 문제가 발생했을 때 "후드 아래(under the hood)"를 볼 수 없어 대응이 불가능합니다.
- **Glassbox AutoML (글래스박스):**
  - Databricks 등이 추구하는 접근법입니다.
  - 모델 생성에 사용된 **모든 과정을 투명하게 노트북(Notebook)으로 제공**합니다.
  - **제공되는 산출물:**
    1. 데이터 탐색(EDA) 노트북
    2. 하이퍼파라미터 튜닝 과정이 담긴 노트북
    3. 최종 선정된 '베스트 모델' 노트북 (예: LightGBM)
    4. 모델의 예측 근거를 설명하는 SHAP 등 해석 가능성 자료
  - **장점:** MLflow와 통합되어 모든 실험이 추적되며, '시민 데이터 과학자(Citizen Data Scientist)'가 생성한 모델을 전문가가 이어받아 수정하고 배포할 수 있습니다.

3.2 AutoML 지원 모델 유형

AutoML은 다양한 유형의 문제에 대해 여러 모델을 동시에 병렬로 실행하고 비교합니다. 다음 표는 Databricks AutoML에서 지원하는 모델의 예시입니다.

Table 2: *AutoML* 지원 모델 예시 (분류, 회귀, 예측)

분류 (Classification)	회귀 (Regression)	예측 (Forecasting)	예측 (서버리스)
결정 트리 (Decision trees)	결정 트리 (Decision trees)	Prophet	Prophet
랜덤 포레스트 (Random forests)	랜덤 포레스트 (Random forests)	Auto-ARIMA	Auto-ARIMA
로지스틱 회귀 (Logistic regression)	선형 회귀 (Linear regression)		DeepAR
XGBoost	XGBoost		
LightGBM	LightGBM		

참고: 분류는 '고객이 이탈할까?(Yes/No)', '암인가?(Yes/No)'처럼 정해진 범주로 나누는 문제입니다. **회귀**는 '내일의 금값은?', '부동산 가격은?'처럼 연속된 숫자 값을 예측하는 문제입니다. **예측**은 시간 순서가 있는 데이터(시계열)를 기반으로 미래를 예측하는 것입니다. (예: Prophet, ARIMA)

3.3 AutoML 활용 시나리오 (데모 요약)

강의에서는 고객 이탈(Churn) 데이터셋을 사용한 AutoML 데모를 시연했습니다.

1. **실험 설정:** 'churn' 데이터를 선택하고, 예측할 컬럼('Churn' 여부)을 지정합니다. 평가 지표로 'F1 Score'를 선택하고, 최대 15분의 타임아웃을 설정합니다.
2. **데이터 탐색 (EDA):** AutoML이 자동으로 생성한 '데이터 탐색 노트북'을 확인합니다. 각 변수의 분포, 결측치, 고유값 등을 시각화하여 보여주므로, 사용자가 직접 코드를 짤 필요가 없습니다.
3. **모델 확인:** 15분 후, 여러 모델(XGBoost, LightGBM, 로지스틱 회귀 등)이 F1 Score 기준으로 정렬됩니다. 이 중 'LightGBM'이 베스트 모델로 선정되었습니다.
4. **코드 검토 (Glassbox):** '베스트 모델 노트북'을 열면, 데이터 전처리(Boolean, Numeric, Categorical 컬럼 분리), 하이퍼파라미터 튜닝(Hyperopt 사용), 파이프라인 구성 등 **모든 소스 코드**가 공개됩니다. 사용자는 이 코드를 그대로 사용하거나 수정할 수 있습니다.
5. **모델 해석:** SHAP 라이브러리를 사용해 어떤 피처(Feature)가 이탈 예측에 중요하게 작용했는지(Feature Importance) 확인할 수 있습니다.
6. **모델 등록:** 클릭 한 번으로 이 베스트 모델을 MLflow 모델 레지스트리에 등록하여, 추후 배치 또는 실시간 추론에 사용할 수 있습니다.

4 머신러닝의 오류: 편향과 분산

머신러닝 모델이 완벽한 예측을 하지 못하는 이유는 '오류(Error)' 때문입니다. 모델의 총 예측 오류는 세 가지 구성 요소로 나뉩니다.

모델의 총 오류 (Prediction Error) **총 오류 = 편향(Bias) 오류 + 분산(Variance) 오류 + 줄일 수 없는 오류 (Irreducible Error)**

- **줄일 수 없는 오류 (Irreducible Error):** 데이터 자체에 포함된 무작위성(노이즈)으로, 모델이 제어할 수 없는 한계입니다.

우리가 집중해야 할 것은 **편향**과 **분산**입니다.

4.1 편향 (Bias) = 과소적합 (Underfitting)

- **정의:** 모델이 너무 단순해서, 데이터에 내재된 실제 관계(신호)를 제대로 잡아내지 못할 때 발생하는 오류입니다.
- **비유:** 복잡한 곡선 형태의 데이터를 단순한 '직선'으로만 예측하려는 것과 같습니다.
- **결과:** 훈련(Train) 데이터와 테스트(Test) 데이터 모두에서 성능이 낮게 나옵니다.
- **특징: 높은 편향 (High Bias), 낮은 분산 (Low Variance)**을 보입니다. (모델이 단순해서 예측이 안정적이지만, 정답과는 거리가 멈)
- **슬라이드 예시:** 그래프 A (데이터 분포를 제대로 따르지 못하는 단순한 선형 모델)

4.2 분산 (Variance) = 과대적합 (Overfitting)

- **정의:** 모델이 너무 복잡해서, 데이터의 실제 신호(signal)뿐만 아니라 노이즈(noise)까지 과도하게 학습했을 때 발생하는 오류입니다.
- **비유:** 훈련 데이터의 모든 점을 통과하기 위해 극도로 구불구불한 선을 그리는 것과 같습니다.
- **결과:** 훈련(Train) 데이터에서는 성능이 매우 높지만, 새로운 데이터(Test)에서는 성능이 급격히 떨어집니다. 모델이 '일반화'에 실패한 것입니다.
- **특징: 낮은 편향 (Low Bias), 높은 분산 (High Variance)**을 보입니다. (훈련 데이터에 너무 맞춰져 있어, 데이터가 조금만 바뀌어도 예측이 크게 흔들림)
- **슬라이드 예시:** 그래프 D (훈련 데이터 포인트를 모두 연결하는 복잡하고 구불구불한 모델)

4.3 편향-분산 트레이드오프 (Bias-Variance Trade-off)

가장 중요한 개념: 트레이드오프 편향과 분산은 **트레이드오프(Trade-off)** 관계에 있습니다.

- 모델을 **단순하게** 만들면 (예: 덜 훈련시킴) → 분산은 낮아지지만, **편향이 높아집니다.** (과소적합)
- 모델을 **복잡하게** 만들면 (예: 너무 많이 훈련시킴) → 편향은 낮아지지만, **분산이 높아집니다.** (과대적합)

좋은 머신러닝 모델이란, 이 두 오류가 적절히 균형을 이루는 '스위트 스팟(Sweet Spot)'을 찾는 것입니다.

5 모델 편향(Model Bias) 심층 분석

모델이 특정 집단에게만 불리한 예측을 하거나, 사회적 편견을 그대로 학습하는 문제를 '모델 편향'이라고 합니다. 이는 앞서 설명한 통계적 편향(과소적합)과는 다른, '공정성(Fairness)'의 문제입니다.

5.1 모델 편향은 왜 발생하는가?

5.1.1 1. 인간의 인지 편향 (Human Factor)

모델은 데이터를 통해 학습합니다. 만약 데이터 자체가 인간의 편견을 담고 있다면, 모델은 그 편견을 그대로, 심지어 증폭시켜 학습합니다.

- **사례 1: 안면 인식 기술의 정확도 차별 (슬라이드 6 그래프)**
  - **현상:** Microsoft, IBM 등 여러 기업의 안면 인식 기술이 '밝은 피부의 남성'에게는 90% 이상의 높은 정확도를 보인 반면, '어두운 피부의 여성'에게는 6070%대의 현저히 낮은 정확도를 보였습니다.
  - **원인:** 이는 알고리즘 자체의 문제가 아니라, **훈련 데이터의 편향성** 때문입니다. 모델을 훈련시킬 때 사용한 데이터셋에 '어두운 피부의 여성'의 안면 데이터가 절대적으로 부족(Under-represented)했기 때문입니다. 모델은 충분히 배우지 못한 것에 대해 나쁜 성능을 보일 수밖에 없습니다.
- **사례 2: 검색 엔진의 편견**
  - **현상:** 과거 일부 검색 엔진에서 'evil(악)'을 검색하면 특정 인종의 이미지가, 'good(선)'을 검색하면 백인 아기 이미지가 주로 노출되었습니다.
  - **원인:** 이는 인터넷상의 데이터(텍스트, 이미지 태그)에 존재하는 인간의 사회적, 문화적 편견을 모델이 그대로 학습했기 때문입니다.
- **사례 3: Microsoft 'Tay' 챗봇**
  - **현상:** 사용자와의 대화를 통해 학습하도록 설계된 챗봇 'Tay'가, 일부 악의적인 사용자들로부터 인종차별적, 혐오적 발언("홀로코스트는 조작된 것이다")을 학습하여 공개적으로 해당 발언을 하기 시작했습니다.
  - **원인:** 나쁜 품질의 입력 데이터를 필터링 없이 실시간으로 학습한 결과입니다.

5.1.2 2. 낮은 품질의 훈련 데이터 (Poor Quality Data)

데이터의 양이 많더라도 품질이 낮으면 편향이 발생할 수 있습니다.

- **데이터 해상도 문제:** 슬라이드 6의 '개(Volpino Italiano) vs 북극 여우' 사진처럼, 이미지가 너무 저해상도이거나 품질이 나쁘면, 인간도 구분하기 어렵습니다. 모델도 마찬가지입니다.
- **데이터 라벨링 오류:** (지도학습의 경우) 데이터를 분류하는 '라벨' 자체가 잘못되었을 수 있습니다. 인간 라벨러가 편견을 가지고 '악당' 캐릭터 이미지를 '어두운' 색상과 연관지어 라벨링했다면, 모델은 그 '잘못된' 관계를 학습합니다.

5.2 모델 편향을 줄이는 방법 (슬라이드 7)

편향 오류 줄이기

- **모델 변경:** 데이터의 특성(예: 비선형 관계)에 맞지 않는 단순한 모델(예: 선형 회귀)을 사용하고 있다면, 더 복잡하거나 적절한 모델(예: 트리 기반 알고리즘)로 변경합니다.

- **데이터의 대표성 확보 (가장 중요):** 훈련 데이터가 모든 그룹(인종, 성별, 연령 등)을 공정하게 대표하고 있는지 확인하고, 부족한 그룹의 데이터를 추가로 수집해야 합니다.
- **가중치 또는 페널티 모델 사용:** 데이터 불균형이 심한 경우(다음 섹션 참고), 소수 그룹에 더 높은 가중치(weight)를 부여하는 모델을 사용합니다.
- **앙상블 학습 (Ensemble Learning):** (편향보다는 분산을 줄이는 데 더 효과적이지만) 여러 모델(약한 학습기, 강한 학습기)을 결합하여 전반적인 모델의 견고함과 정확도를 높입니다.
- **더 많은 데이터로 훈련 (분산 줄이기):** 데이터가 많아지면 데이터 대 노이즈 비율이 증가하여, 모델이 특정 노이즈에 과대적합(Overfitting)되는 것을 방지하고 일반화 성능(분산 감소)을 높이는 데 도움이 됩니다.

6 데이터 불균형 문제 (Class Imbalance)

모델 편향의 특수한 경우로, 훈련 데이터의 클래스(정답) 분포가 한쪽으로 크게 치우친 상황을 말합니다.

- **예시:** 신용카드 사기 탐지 (정상 99.9%, 사기 0.1%), 암 진단 (정상 98%, 암 2%)
- **산업 분야:** 사기 탐지(Fraud), 클레임(Claim), 이상 징후(Anomaly), 침입 탐지(Intrusion) 등

6.1 불균형 데이터의 진짜 문제: '정확도'의 함정

대부분의 머신러닝 알고리즘은 **'정확도(Accuracy)'** (전체 중 몇 개를 맞혔는가)를 최대화하도록 설계되었습니다.

정확도(Accuracy)의 함정 사기 거래 비율이 0.1%인 데이터셋이 있다고 가정해 봅시다. 만약 어떤 모델이 모든 거래에 대해 **"정상(Not Fraud)"**이라고만 예측한다면?

이 모델의 정확도는 **99.9%**입니다. 정확도 지표상으로는 완벽해 보이지만, 우리가 잡고 싶은 '사기 거래'는 단 한 건도 잡아내지 못하는, 완전히 쓸모없는 모델입니다.

6.2 해결책 1: 평가 지표 변경 (Accuracy 대신 F1 Score)

불균형 데이터에서는 정확도 대신, **정밀도(Precision)**, **재현율(Recall)**, 그리고 이 둘을 조합한 **F1 Score**를 사용해야 합니다.

정밀도(Precision) vs 재현율(Recall)

- **정밀도 (Precision):** 모델이 "사기"라고 예측한 것 중에, 실제 "사기"인 비율. (TP / (TP+FP))
  - **중요한 경우:** 스팸 메일 분류 (정상 메일을 스팸으로 잘못 분류하면 안 됨). 모델의 예측을 신뢰해야 할 때.
- **재현율 (Recall):** 실제 "사기"인 것 중에, 모델이 "사기"라고 맞힌 비율. (TP / (TP+FN))
  - **중요한 경우: 암 진단, 사기 탐지** (실제 암 환자나 사기 거래를 '정상'으로 놓치면 치명적임).
- **F1 Score:** 정밀도와 재현율의 조화 평균. $2 \times \frac{Precision \times Recall}{Precision + Recall}$
  - 두 지표가 모두 중요할 때 사용하는, 불균형 데이터의 **핵심 평가지표**입니다.

6.3 해결책 2: 리샘플링 (Resampling)

데이터의 분포를 인위적으로 조절하여 균형을 맞추는 기법입니다.

- **과소추출 (Undersampling):** 다수 클래스(예: 정상 거래) 데이터를 대거 삭제하여 소수 클래스(사기 거래)와 비율을 맞춥니다.
  - **단점:** 유용한 정보를 잃어버릴 위험이 큽니다.
- **과대추출 (Oversampling):** 소수 클래스 데이터를 복제하여 수를 늘립니다.
  - **단점:** 단순히 복제만 하면 모델이 특정 샘플에 과대적합(Overfitting)될 수 있습니다.

6.4 해결책 3: SMOTE (합성 소수 오버샘플링 기법)

단순 과대추출(Oversampling)의 과대적합 문제를 해결하기 위해 등장한 기법입니다.

SMOTE (Synthetic Minority Oversampling Technique) 작동 방식 SMOTE는 데이터를 '복제'하는 것이 아니라 '생성'합니다. (슬라이드 12 그림 참고)

1. 소수 클래스(예: 검은색 원)에서 임의의 샘플(x_1)을 선택합니다.
2. 그 샘플(x_1)과 가장 가까운 이웃 샘플들(x_2, x_3 등)을 찾습니다.
3. 두 샘플을 잇는 **직선 상에** 무작위로 점(a, b, e)을 찍어, 이를 **'새로운 합성 데이터'**로 간주합니다.

이 방식을 통해 모델은 소수 클래스의 '특징 공간'을 더 넓고 견고하게 학습할 수 있습니다.

리샘플링 시 절대 주의사항 데이터 리샘플링(SMOTE 등)은 데이터를 훈련(Train) / 테스트(Test) 세트로 **분리한 후, 훈련(Train) 세트에만 적용**해야 합니다.

만약 전체 데이터에 SMOTE를 적용한 후 분리하면, 원본과 합성 데이터가 훈련/테스트 셋에 섞여 들어가기 때문에, 테스트 성능이 비정상적으로 높게 측정되는 '데이터 누수(Data Leakage)'가 발생합니다.

7 실전 사례 연구: 실시간 사기 탐지 시스템

강의 후반부는 이 모든 개념이 적용된, Eric Gieseke가 직접 참여했던 대규모 신용카드 사기 탐지 시스템 (Fraud Detection System) 구축 사례입니다.

7.1 비즈니스 요구사항

이 시스템의 성공 여부를 결정짓는 핵심 요구사항은 다음과 같습니다.

- **높은 신뢰도:** 실제 사기(Fraud)는 반드시 잡아내고(High Recall), 정상 거래를 사기로 오인(False Positive)해서는 안 됩니다. (정상 거래를 막으면 고객이 카드를 잘라버릴 수 있음)
- **초저지연 응답 속도 (Latency): → 15 밀리초 (ms)**
  - 고객이 카드를 긁고 POS 기기에서 응답을 받기까지의 총 시간(약 1 2초) 중, 사기 탐지 시스템에 할당된 시간입니다. 15ms는 일반적인 디스크(HDD)에서 데이터를 읽는 시간보다도 짧은, 극도로 도전적인 목표입니다.
- **높은 처리량 (Throughput): → 초당 50,000 건 (TPS)**
- **유연성:** 데이터 사이언티스트가 새로운 피처(Feature)나 룰(Rule)을 쉽게 추가하고 배포할 수 있어야 합니다.

7.2 피처(Feature) 정의

사기를 탐지하기 위해 원본 거래(Transaction) 데이터로부터 다양한 '파생 변수(피처)'를 계산합니다.

- **거래(Transaction) 기반 피처:**
  - 고객의 집 주소로부터의 거리 (예: 평소엔 서울인데 갑자기 이스탄불에서 결제)
  - 고객의 평균/최대/최소 거래 금액과의 차이 (예: 평소 1만원 쓰는데 갑자기 2,000만원 결제)
  - 당일 거래 횟수 (예: 평소 하루 2건인데 1시간 내 100건 결제)
- **차원(Dimension) 기반 피처:**
  - 판매자(*Merchant*)의 평균 거래 금액
  - 소비자(*Consumer*)의 거래 빈도

7.3 시스템 아키텍처: 람다 아키텍처 (Lambda Architecture)

15ms 응답 속도와 대용량 배치를 모두 만족시키기 위해, **람다(Lambda) 아키텍처**를 채택했습니다. (슬라이드 19, 21 참고)

람다(Lambda) 아키텍처 대용량 데이터 파이프라인의 표준 중 하나로, 전체 시스템을 3개의 레이어로 분리합니다.

- **배치 레이어 (Batch Layer) (Hadoop/Spark)**
  - 모든 원본 데이터(Immutable)를 저장하고, 주기적으로(예: 매일 밤) 전체 데이터를 계산하여 '마스터 데이터셋'(피처 값)을 만듭니다. (느리지만 정확함)
  - 역할: 새로운 피처를 추가할 때, 과거 1년치 데이터에 대해 이 피처 값을 계산(Backfill)합니다.
- **스피드 레이어 (Speed Layer) (CEP / Realtime)**
  - 실시간으로 들어오는 데이터(이벤트)를 즉시 처리합니다. (빠르지만 근사치일 수 있음)

  - 역할: '지금' 들어온 거래가 사기인지 아닌지 15ms 안에 판단합니다.
- **서빙 레이어 (Serving Layer) (Cassandra / Feature Store)**
  - 배치 레이어와 스피드 레이어의 결과를 결합하여 저장하고, 실시간 쿼리에 응답합니다.

7.4 핵심 기술 1: 메타데이터 기반 코드 생성 (Slide 19)

- **문제:** 수백 개의 피처를 배치(SQL)와 스피드(EPL) 환경을 위해 두 번씩 수동으로 코딩해야 했습니다. 이는 오류가 발생하기 쉽고 유지보수가 어렵습니다.
- **해결:** 피처를 '코드'로 정의하지 않고, **'메타데이터(Metadata)'**로 정의했습니다.
  - (예) 메타데이터: '$\text{Feature}_A = AVG(Transaction_Amount, 7days)$'
- **코드 생성기(Code Generator)**가 이 메타데이터를 읽어, 배치용 **SQL 코드**와 스트리밍용 **EPL 코드**를 **자동으로 생성**했습니다.
- **효과:** 피처 관리가 용이해지고, 인간의 실수(Human Error)가 줄어들었습니다.

7.5 핵심 기술 2: 순환 버퍼 (Circular Buffer) (Slide 20)

- **문제:** "특정 고객의 최근 7일간 평균 거래액"을 15ms 안에 계산해야 합니다. 수백만 명의 고객에 대해 최근 7일치 거래 내역 전체를 DB에서 읽어오는 것은 불가능합니다.
- **해결: 순환 버퍼(Circular Buffer)**라는 독자적인(Proprietary) 데이터 구조를 고안했습니다.
- **작동 원리:**
  - 고객별로 7일치(7칸) 버퍼를 만듭니다.
  - 각 칸(날짜)에 **모든 거래 내역**을 저장하는 대신, 오직 **'거래 횟수(Count)'**와 **'거래 총액(Sum)'** 두 값만 저장합니다.
  - (예: [월: (3건, 30$), 화: (5건, 50$), ... 일: (5건, 250$)])
  - 새 날(월요일)이 되면, 8일 전의 데이터(지난주 월요일)를 버리고 새 데이터를 덮어씁니다.
- **효과:** 이 컴팩트한 구조 덕분에, (Count의 합) / (Sum의 합) 연산만으로 7일간의 평균, 합계, 최대/최소값 등을 **메모리상에서 매우 빠르게** 계산할 수 있었습니다.

7.6 핵심 기술 3: 서빙 레이어 (Cassandra) (Slide 22)

- **문제:** 순환 버퍼 외에도, 고객/판매자/터미널별 피처를 저장하고 15ms 내에 '읽어야' 할 저장소가 필요했습니다.
- **해결: Cassandra (NoSQL)** 데이터베이스를 피처 스토어(Feature Store)로 채택했습니다.
- **이유:** Cassandra는 쓰기(Write)도 빠르지만, 이 시스템에서는 **극도로 빠른 읽기(Read) 속도** (약 2.3ms 달성) 때문에 선택되었습니다.
- **데이터 모델링 (Columnar):**
  - RDBMS처럼 정규화된 테이블(고객 테이블, 판매자 테이블...)을 사용하지 않았습니다.
  - 단 2개의 거대한 테이블: **이벤트 테이블(Fact)**과 **차원 테이블(Dimension)**을 사용했습니다.
  - Cassandra의 '컬럼(Column)' 기반 특성을 활용하여, 새로운 피처나 차원이 생길 때마다 **단순히 새 컬럼을 추가**하는 **유연한 스키마(Flexible Schema)**를 구현했습니다.

7.7 종합 아키텍처 (Slide 21)

Streaming Fraud Detection 시스템 흐름

1. **(분석가)** 사기 분석가가 GUI/CLI를 통해 '메타데이터 서비스'에 새로운 피처나 룰을 정의합니다. (예: 'Feature$_B$')
1. **(배치)** '배치 프로세싱(Spark)'이 이 메타데이터를 기반으로 코드를 생성, 과거 1년치 데이터에 대해 'Feature$_B$' ′ ($Cassandra$)′ .
1. **(스트리밍)** '스트림 분석(Stream Analytics)'이 이 메타데이터를 기반으로 실시간 코드를 업데이트합니다.
2. **(고객)** 고객이 '결제 서비스(Payment Service)'를 통해 카드 거래를 요청합니다.
3. **(탐지)** '사기 탐지 서비스(Fraud Detection Service)'가 거래 정보를 '스트림 분석'에 전달합니다.
4. **(판단 15ms)** '스트림 분석'은 **15ms 이내**에 다음을 수행합니다.
   - (a) 실시간 피처 계산 (순환 버퍼 등)
   - (b) 과거 피처 조회 (Cassandra에서 'Feature$_B$')
   - (c) ML 모델 및 룰 실행
   - (d) '사기/정상' 응답(Response) 반환
5. **(승인)** '사기 탐지 서비스'가 '정상' 응답을 '결제 서비스'에 전달하고, 고객의 거래가 승인됩니다.

8 학습 점검을 위한 체크리스트

이번 강의 내용을 스스로 점검해 봅시다.

핵심 개념 체크리스트

모델 오류의 세 가지 구성요소(편향, 분산, 줄일 수 없는 오류)를 설명할 수 있는가?

편향-분산 트레이드오프(Trade-off)가 무엇인지, 왜 중요한지 이해하는가?

과소적합(Underfitting)과 과대적합(Overfitting)이 각각 편향, 분산과 어떻게 연결되는지 아는가?

안면 인식 사례에서 '어두운 피부의 여성'에 대한 정확도가 낮았던 근본적인 원인을 설명할 수 있는가?

데이터 불균형 문제에서 '정확도(Accuracy)'를 평가지표로 쓰면 안 되는 이유를 설명할 수 있는가?

'정밀도(Precision)'와 '재현율(Recall)'의 차이를 암 진단(Cancer) 예시로 설명할 수 있는가?

SMOTE가 무작위 오버샘플링(Random Oversampling)과 어떻게 다른지 설명할 수 있는가?

람다(Lambda) 아키텍처가 왜 배치(Batch) 레이어와 스피드(Speed) 레이어를 둘 다 사용하는지 아는가?

사기 탐지 시스템이 15ms의 응답 속도를 맞추기 위해 사용한 핵심 기술 2가지(순환 버퍼, Cassandra)가 어떤 문제를 해결했는지 아는가?

9 FAQ (자주 묻는 질문)

초심자가 혼동할 수 있는 내용들을 Q&A 형식으로 정리합니다.

Q: AutoML을 쓰면 데이터 사이언티스트가 필요 없나요? **A:** 아닙니다. AutoML은 '시민 데이터 과학자'가 좋은 출발점(Baseline)을 얻거나, 데이터의 유용성을 빠르게 검증하는 데 매우 유용한 도구입니다.

하지만 'Blackbox' AutoML은 내부 수정이 어렵고, 'Glassbox' AutoML이라 할지라도, 결국 비즈니스 문제를 정의하고, 어떤 데이터를 사용할지 결정하며, 모델의 결과를 해석하여 비즈니스에 적용하는 것은 여전히 데이터 사이언티스트와 도메인 전문가의 핵심 역량입니다.

Q: 모델 편향은 알고리즘만 수정하면 해결되나요? **A:** 아닙니다. 편향의 근본 원인은 알고리즘 자체가 아니라 **'데이터'**에 있는 경우가 압도적으로 많습니다. (예: 특정 그룹 데이터 부족, 데이터 라벨링의 편견)

알고리즘 수정(예: 가중치 부여)도 한 가지 방법이지만, 가장 중요한 해결책은 훈련 데이터 자체가 현실 세계를 공정하고 다양하게 대표할 수 있도록 **데이터 수집 및 정제 과정**에 투자하는 것입니다.

Q: 15ms (0.015초) 안에 어떻게 그 많은 계산을 하나요? **A:** '모든' 계산을 15ms 안에 하는 것이 아닙니다. 핵심은 **'미리 계산(Pre-computation)'**과 **'빠른 조회(Fast Retrieval)'**입니다.

- (1) **미리 계산:** '고객의 1년치 평균 거래액'처럼 오래 걸리는 피처는 **배치 레이어(Spark)**가 매일 밤 미리 계산해서 **Cassandra**에 저장해 둡니다.
- (2) **빠른 조회:** 실시간 요청이 오면, 이 미리 계산된 값을 Cassandra에서 2.3ms 만에 **조회**합니다.
- (3) **실시간 계산:** '최근 10분간 거래 횟수'처럼 실시간성이 강한 피처만 **순환 버퍼(Circular Buffer)** 같은 메모리 기반 구조를 이용해 즉석에서 계산합니다.

결국 15ms는 이 '조회 + 최소한의 실시간 계산 + 모델 추론'을 합친 시간입니다.

10 빠르게 훑어보기 (1-Page 요약)

강의 10 핵심 요약 카드

오류 1: 편향(Bias) vs 분산(Variance)

편향 (과소적합): 모델이 너무 단순함. (훈련/테스트 모두 실패) **분산 (과대적합):** 모델이 너무 복잡함. (훈련만 성공, 테스트 실패) **Trade-off:** 둘은 반비례 관계. 둘의 균형점을 찾는 것이 목표.

오류 2: 모델 편향(Model Bias)

원인 1: 인간의 편견: 데이터 자체가 편견을 가짐 (예: 검색 결과) **원인 2: 데이터 부족:** 특정 그룹(예: 어두운 피부 여성) 데이터가 부족(Under-represented)하여 모델이 제대로 학습하지 못함. **해결:** 데이터의 **대표성**과 **다양성** 확보가 최우선.

오류 3: 데이터 불균형(Imbalance)

문제: 사기(1%) vs 정상(99%). '정확도(Accuracy)'는 99%짜리 가짜 모델을 만듦. **해결 1 (평가): F1 Score** (정밀도+재현율) 사용. (특히 재현율이 중요!) **해결 2 (데이터): SMOTE** 기법으로 소수 클래스의 '합성' 데이터를 생성하여 밸런스 맞추기. (단, Train 셋에만 적용!)

실전: 사기 탐지 시스템 (15ms 응답)

아키텍처: 람다(Lambda) (배치 + 스피드) **핵심 1 (계산): 순환 버퍼(Circular Buffer)** 사용. ('Count'와 'Sum'만 저장하여 메모리에서 초고속 집계) **핵심 2 (저장): Cassandra** (NoSQL) 사용. (2.3ms의 초고속 읽기 속도로 피처 조회) **핵심 3 (유지보수): 코드 생성기** (메타데이터로 피처 정의)