Feature 저장소 기술 동향

A Survey on Feature Store

허성진 (S.J. Hur, sjheo@etri.re.kr) 스마트데이터연구실 책임연구원
김지용 (J.Y. Kim, kjy@etri.re.kr) 스마트데이터연구실 책임연구원/실장

ABSTRACT

In this paper, we discussed the necessity and importance of introducing feature stores to establish a collaborative environment between data engineering work and data science work. We examined the technology trends of feature stores by analyzing the status of some major feature stores. Moreover, by introducing a feature store, we can reduce the cost of performing artificial intelligence (AI) projects and improve the performance and reliability of AI models and the convenience of model operation. The future task is to establish technical requirements for establishing a collaborative environment between data engineering work and data science work and develop a solution for providing a collaborative environment based on this.

KEYWORDS 데이터 플랫폼, 데이터 공학, AI 플랫폼, 데이터 과학, 협업 환경, Feature 저장소

Ⅰ. 서론

AI 기반 응용 서비스를 개발하는 AI 프로젝트는 일반적으로 주어진 데이터를 통해 AI 모델을 학습하고, 학습된 모델을 이용하여 예측 및 추론을 시행한다. 이를 위해 데이터 플랫폼은 다양한 데이터 소스로부터 데이터를 취득하고, 이를 저장하는 데이터 저장소 구축, ETL을 포함한 데이터 플로우 관리 그리고 Feature 엔지니어링 등의 작업 수행을 통해 최종적으로 AI 모델 개발에 필요한 훈련 데이터를 생성하는 데이터 공학 작업을 수행한다. 다음으로, AI 플랫폼은 데이터 플랫폼에서 생성한 훈련 데이터를 활용하여 모델 학습, 모델 선택 및 검증, 하이퍼파라미터 최적화, 서로 다른 실행 환경을 고려한 모델 배포 기술을 통해 AI 응용 서비스에 개발된 모델을 적용하고, 이후 운영 중인 모델의 성능을 지속적으로 관찰하면서 모델 개선 필요성 여부를 판단하는 데이터 과학 작업을 수행한다[1].

두 플랫폼이 담당하는 서로 다른 성격의 업무 처리 효율성을 위하여, 데이터 플랫폼은 통상적으로 데이터 집중적인 작업 처리에 적합한 하둡(Ha–

* DOI: https://doi.org/10.22648/ETRI.2021.J.360207
* 본 연구는 한국전자통신연구원 연구운영지원사업의 일환으로 수행되었음[21HB1330, 시공간복합 인공지능 녹조 예측 기술 개발].

doop) 계열의 대규모 분산처리 시스템을 기반으로 구축하는 반면에, AI 플랫폼은 데이터 및 계산 집중적인 작업 처리에 적합한 GPU와 같은 전용처리 장치 또는 고성능 서버 그리고 최근에는 클라우드 컴퓨팅 기술 등에 기반한 시스템을 통해 구축하고 있다.

AI 프로젝트는 기존의 전통적 SW 프로젝트가 주로 소프트웨어 코딩 작업 위주로 진행되는 것과는 달리, 대규모 데이터에 내재하는 의미를 도출하기 위한 실험적(Experimental), 반복적(Iterative), 탐색적(Exploratory) 작업을 수행하는 특성으로 인하여 데이터 플랫폼 작업과 AI 플랫폼 작업을 순환 반복하는 경우가 빈번하게 발생한다. 즉, 데이터 플랫폼의 작업 결과인 특정 훈련 데이터를 기반으로 AI 플랫폼에서 모델을 개발하다가 원하는 성능 목표에 도달하지 못하거나 모델이 제대로 학습되지 않는 경우 다시 데이터 플랫폼 작업으로 돌아가서 훈련 데이터를 수정한 후 AI 플랫폼에서 모델을 개발하는 방식으로 진행되는데, 이는 AI 프로젝트 수행에서 데이터 플랫폼과 AI 플랫폼을 연계하는 협업의 효율성이 AI 프로젝트 소요 시간 및 개발 비용에 중대한 영향을 끼칠 수 있음을 시사한다[2].

현재, AI 프로젝트 수행에서 TensorFlow, Keras, PyTorch, Caffe 등의 다양한 ML 프레임워크와 더불어 Airflow, Argo, Luigi, MLflow, Kubeflow, TensorFlowExtended 등의 ML 워크플로우 및 파이프라인 관리도구들이 널리 활용되고 있지만, 이러한 도구들은 주로 AI 모델의 개발, 배포, 적용 등 AI 플랫폼 관련 작업 또는 AI 프로젝트 수행에 필요한 종단 간 워크플로우 구축 및 관리를 지원하는 기능에 집중하고 있으며, 데이터 플랫폼과 AI 플랫폼이 유기적으로 연계·통합하여 AI 프로젝트의 수행 효율성을 제공하기 위한 도구 및 기술 개발은 미흡한 실정이다[3].

본 고에서는 AI 프로젝트 진행 과정에서 데이터 플랫폼에 기반한 데이터 공학 작업과 AI 플랫폼에 기반한 데이터 과학 작업 사이의 원활한 연계·협업 도구로서 Feature 저장소 도입의 필요성과 중요성을 강조하기 위하여 Feature 저장소의 개념과 특성 및 주요 기능에 대한 설명에 이어 현재 널리 활용되는 주요 Feature 저장소의 현황 소개를 통해 Feature 저장소의 기술 동향을 알아보고자 한다.

Ⅱ. Feature 저장소

1. Feature 저장소 개념

Feature 저장소(Feature Store)는 2017년 Uber가 자사의 AI 응용 서비스 개발 편의성과 AI 모델 성능 및 신뢰성 향상을 위해 자체적으로 구축하여 운영하는 Michelangelo AI 개발 플랫폼을 통해 그 개념이 처음으로 소개되었다. Michelangelo 플랫폼에서는 데이터 플랫폼이 다양한 데이터 소스로부터 Feature 엔지니어링 과정을 통해 생성하는 모든 Feature 데이터를 Feature 저장소에 저장·등록하고, AI 플랫폼은 AI 모델 개발에 필요한 Feature 데이터를 Feature 저장소로부터 검색하여 해당 Feature 데이터를 AI 모델 개발에 활용하는 방식으로 Feature 저장소를 도입한다.

통상적으로 Feature는 계량할 수 있는 현상의 특성이나 성질을 의미하며, AI 모델 설정 또는 운영 중인 AI 모델에서의 추론·예측에 활용되고 있다. 이러한 Feature는 파일, 데이터베이스, 센서 등 다양한 데이터 소스로부터 직접 측정하거나 도출된 값 또는 통계값, 다수의 데이터 소스를 통하여 유도되는 값 등 다양한 방법을 거쳐서 생성되며 그림 1은 Feature 저장소의 개념도를 도식적으로 간략하게 나타내었다.

출처 게티이미지뱅크, 무단 전재 및 배포금지

그림 1 Feature 저장소 개념도

가. 데이터 웨어하우스 vs Feature 저장소

Feature 저장소는 간단히 생각해서 Feature 데이터 웨어하우스로 간주할 수 있는데, 기업의 전통적 데이터 웨어하우스와 기능상의 구체적 비교를 통해 Feature 저장소의 개념 및 특징을 설명하고자 한다.

데이터 웨어하우스는 기업의 모든 데이터를 통상적으로 중앙저장소에 집중하여 저장하고, Tableau나 Power BI와 같은 가시화 도구를 통한 분석 환경을 제공함으로써 기업의 구성원들이 데이터의 저장 위치나 질의 방법 결정 등에 대한 어려움 없이 기업의 구성원 누구나 쉽게 기업의 Historical Insight 도출이 가능한 환경을 제공한다[4].

Feature 저장소는 데이터 과학자의 예측 모델 생성을 위해 필요한 Feature 데이터를 위한 중앙 저장소로서 다양한 AI 모델 개발 시 Feature 저장소에 저장된 Feature 데이터 가운데 필요한 Feature 데이터를 검색하여 AI 모델을 설정하는 데 활용한다[5].

데이터 웨어하우스와 Feature 저장소는 모두 정제된 데이터의 중앙저장소이며, 다양한 데이터 소스로부터 정제된 데이터를 추출하기 위해 각각의 파이프라인(Data 파이프라인, Feature 파이프라인)을 통해 원천 데이터에 대한 적절한 처리 작업을 거친다는 공통점이 있는 반면, 두 중앙저장소에 저장된 데이터의 최종 사용자가 각각 데이터 분석가와 데이터 과학자라는 차이로 인해 두 중앙저장소에서 저장하는 정제 데이터를 추출하는 전처리 작업 및 두 중앙저장소 구현에 필요한 데이터베이스에 대한 요구사항이 서로 상이하다[6].

또한 두 중앙저장소에 데이터를 저장하기 전 Feature 저장소는 반드시 결측치, 이상치 등의 유무를 검증하는 반면, 데이터 웨어하우스는 데이터 검증 과정을 거치지 않는다. 저장된 데이터 접근을 위한 API 관점에서 데이터 웨어하우스는 주로 SQL에 의존하는 반면, Feature 저장소는 Python, Scala/Java, SQL, DSL 등을 활용한다[6].

나. Online vs Offline Feature 저장소

일반적으로 Feature 저장소는 기업의 전통적 데이터 웨어하우스 솔루션을 기반으로 해서 추가적인 기능을 탑재하는 방식으로 구현할 수 있는데, Feature 저장소 구현에 필요한 대표적인 부가 기능 중 하나는 운영 중인 AI 모델에 Serving Feature 데이터를 제공하여 추론·예측 값을 얻기 위한 Feature 데이터 저지연 접근 기능이다.

통상적으로 Hive, BigQuery 같은 관계형 데이터베이스 기반의 데이터 웨어하우스는 대규모 데이터를 대상으로 하는 분석 작업에 적합한 반면, Feature 저지연 접근과 같은 빠른 응답을 요구하는 기능의 제공에는 한계가 있으므로 MySQL 클러스터, Cassandra, Redis와 같은 비관계형 데이터베이스 시스템의 Row-Oriented 데이터 레이아웃 기반 Feature 저장·관리를 통해 Feature 저지연 접근 기능을 구현하고 있으며, 이를 대규모 훈련용 Feature 관리를 위한 일괄 처리용 Offline Feature 저장소와 구분하여 Online Feature 저장소로 지칭한다[7].

대규모 일괄 작업으로 가공되는 Feature는 먼저 Offline Feature 저장소에 보관하여 추후 AI 모델 Train/Test 데이터로 활용하는 반면에, CCTV, 센서 등과 같은 스트리밍 데이터로부터 실시간으로 추출되어 운영 AI 모델에 제공하여 예측·추론 값 도출에 활용되는 Feature는 저지연 접근을 제공하

는 Online Feature 저장소에서 저장·관리한다. 운영 중인 AI 모델의 성능을 보장하기 위해서는 훈련 데이터와 예측·추론을 위해 제공하는 Serving 데이터 사이의 일관성 유지가 매우 중요한데, 이는 곧 Offline Feature 저장소와 Online Feature 저장소 사이의 일관성을 유지하기 위한 기능이 필요함을 의미한다. 이를 위하여 Offline Feature 저장소에 일괄 작업으로 갱신되는 Feature 데이터는 주기적으로 Online Feature 저장소에 전송하고, Online Feature 저장소에 먼저 입력되는 실시간 Feature 데이터는 Offline Feature 저장소에 복사하는 방법 등을 통해 Online/Offline Feature 저장소의 일관성을 유지함으로써 운영 AI 모델의 성능 저하를 방지한다.

다. Feature 저장소 참조 구조

최근 AI 기반 서비스 성공 사례 증가로 인한 AI 기반 서비스 도입 확산과 이에 따른 운영 AI 모델 증가와 더불어 AI 모델 기반 비즈니스의 신규 참여를 위해 새로운 AI 모델 개발을 계획하는 기업들이 많아지는 가운데 Feature 저장소 도입의 중요성 및 필요성에 대한 인식이 점차 부각되고 있다. 하지만 각 기업들의 기존 데이터 인프라 구축 상황이나 구상하는 AI 비즈니스 형태 또는 지향하는 목적 등에 따라 Feature 저장소에 대한 구체적인 요구사항들이 달라지면서 각기 다른 특징과 기능을 제공하는 다양한 Feature 저장소들이 등장하게 되었다.

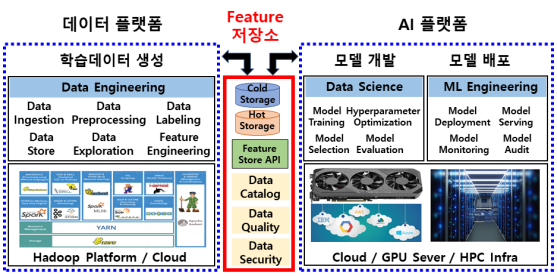
본 절에서는 현재까지 발표된 다양한 Feature 저장소들의 특징과 장단점을 객관적으로 비교하고 평가하기 위한 Feature 저장소의 참조 구조의 주요 특징을 살펴보고, 이를 통해 서로 다른 Feature 저장소의 특징과 기능을 파악할 수 있도록 한다[2].

참조 구조가 제시하는 Feature 저장소 구조의 대표적인 특징은 Feature 저장소 도입 시 이미 설치되어 운영 중인 기존의 데이터 인프라 구조와 배치되지 않아야 한다는 점이다. 이는 기업이 Feature 저장소 도입을 위해 기존의 데이터 인프라와 분리된 별도의 데이터 사일로를 추가로 설치하는 것이 아니라 기존의 데이터 인프라와 통합하여 운영함으로써 기업의 데이터 인프라 단순화와 더불어 일관되고 체계적인 데이터 관리 환경을 유지하는 효과를 위함이고, 실제 Feature 저장소를 직접 개발하거나 또는 외부로부터 도입하여 운영하고 있는 여러 AI 기업의 사례를 통해 이러한 특징의 중요성이 분명하게 드러나고 있다.

2. Feature 저장소 기반 협업

본 고는 AI 프로젝트를 진행하는 과정에서 데이터 플랫폼에 기반하여 데이터 소스로부터 Feature를 추출하는 데이터 공학 작업과 AI 플랫폼에 기반하여 AI 모델을 개발하는 데이터 과학 작업 그리고 AI 모델을 배포하는 ML 공학 작업 사이의 원활한 협업 환경을 제공하기 위한 도구로서 Feature 저장소 도입을 제안하고, 그림 2는 이러한 Feature 저장소 기반 협업 개념도를 도식적으로 나타내었다.

본 절에서는 Feature 저장소 기반 데이터 플랫폼과 AI 플랫폼 협업의 대표적인 효과들을 나열하고,



출처 게티이미지뱅크, 무단 전재 및 배포금지

그림 2 Feature 저장소 기반 협업 개념도

이를 통해 Feature 저장소 도입의 필요성 및 중요성을 설명하고자 한다[3].

가. Feature 공유

데이터 플랫폼에서 생성한 Feature를 Feature 저장소에 저장하고 등록하면, AI 플랫폼은 등록된 Feature로부터 특정 AI 모델 개발에 필요한 Feature를 검색하여 AI 모델 개발에 활용한다. Feature 저장소를 매개로 하여 데이터 플랫폼은 Feature를 생성하고 AI 플랫폼은 Feature를 활용하는 형태로 상호 동작함으로써 서로 다른 AI 모델에서 동일한 Feature가 필요한 경우 각 AI 모델을 위한 Feature를 중복으로 생성할 필요 없이 이미 생성된 Feature를 쉽게 공유할 수 있는 장점을 가진다. 통상적으로 Feature를 생성하는 난이도 및 비용이 아주 큰 것을 고려할 때 Feature 공유는 AI 모델 개발 비용 절감에 큰 효과를 가져 온다. 또한 데이터 플랫폼과 AI 플랫폼이 직접 접촉하는 대신 중간에 Feature 저장소를 경유함으로써 데이터 플랫폼과 AI 플랫폼 사이의 인터페이스 구조가 체계적이고 단순화되어 End-to-end ML 파이프라인 구축 및 관리 비용이 절감되는 효과도 가져올 수 있다.

나. Feature 관리

Feature 저장소라는 중앙 집중화된 장소에서 모든 Feature가 저장되고 활용됨으로써 전체 Feature에 대한 체계적이고 일관성 있는 관리가 가능하다. Feature의 버전 관리, Feature 생성에 관여한 데이터 소스 정보와 더불어 데이터 소스로부터 Feature를 생성하는 과정에서 적용된 변환 연산 정보 등 Feature 생성과 관련된 모든 기록, Feature를 생성한 주체, Feature를 활용한 주체 등 Feature와 관련된 다양한 정보의 체계적인 관리를 통하여 AI 모델 생성 과정 추적 및 재생성(Reproducibility)을 지원함으로써 AI 모델 개발의 효율성과 AI 모델의 신뢰성을 증대한다.

다. Feature 모니터링

Feature 저장소가 AI 플랫폼에 제공하는 Feature는 AI 모델 개발을 위한 훈련 데이터와 운영 중인 AI 모델로부터 예측·추론 값을 얻기 위한 Serving 데이터로 구분된다. 이때 훈련 데이터와 Serving 데이터의 통계적 특성이 상이한 경우 운영 중인 AI 모델은 학습하지 않은 데이터에 대한 예측·추론을 해야 하는 상황을 의미하고, 결국 운영 AI 모델의 성능이 저하되는 문제가 발생한다. 이는 AI 모델 기반의 응용 서비스를 제공하는 실제 현장에서 빈번하게 발생하는 현상으로서, AI 서비스 확산과 함께 운영하는 AI 모델이 증가하면서 문제의 심각성이 점점 주목받는 실정이다. 이러한 문제를 해결하기 위하여 Feature 저장소는 훈련 데이터를 저장하는 Offline Feature 저장소와 Serving 데이터를 저장하는 Online Feature 저장소의 일치성을 유지하는 것과 더불어, 각각의 저장 데이터의 통계적 특성에 대한 지속적인 모니터링을 통해 훈련 데이터와 Serving 데이터 사이의 통계적 일치성 유지 여부를 확인하고 허용 범위를 초과하는 불일치가 발생하는 경우, 이전 학습 이후로부터 변화된 상황을 반영한 갱신된 훈련 데이터를 이용하여 AI 모델을 재학습하여 AI 서비스 운영 환경에 배포함으로써 훈련 데이터와 Serving 데이터 사이의 통계적 특성 불일치에 따른 성능 저하문제를 해결한다.

라. Feature 접근 제어

AI 모델 개발을 위한 Feature 중에는 개인 신상정보 등 민감한 내용이 포함될 수 있다. 또한 개인정보뿐만 아니라 기업이나 비즈니스 측면에서 민감하게 취급해야 하는 Feature 등에 대해서는 엄격

한 접근 제어가 필요하다. 이를 위해 Feature 저장소는 개인 신상 정보 등 민감한 정보를 포함하는 Feature에 대한 체계적인 접근 제어를 위한 정책 수립과 이에 따른 접근 제어 기능 구현과 더불어 이러한 Feature의 사용을 요청하는 주체에 대한 엄격한 인증 및 보안 관리 기능을 제공함으로써 민감한 Feature 정보를 효과적으로 관리할 수 있다.

Ⅲ. Feature 저장소 기술 동향

현재까지 널리 알려진 Feature 저장소는 대략 10여 개 이상이 있으며, 각 Feature 저장소가 적용된 플랫폼과 주요 특징은 다음과 같다[8].

- Hopswokrs: 공개 SW, AGPL V3
- Feast: 공개 SW, Apache V2
- Michelangelo: Uber 전용 플랫폼
- Conde Nast: 상용 솔루션
- Zipline: Airbnb 전용 플랫폼
- Comcast: Comcast 전용 플랫폼
- Metaflow: Netflix 전용 플랫폼
- Twitter: Twitter 전용 플랫폼
- FBLearner: Facebook 전용 플랫폼
- Galaxy: PInterest 전용 플랫폼
- Iguazio: 상용 솔루션
- Tecton: 상용 솔루션

Hopsworks와 Feast는 공개 SW 프로젝트로 개발이 진행되었고, Conde Nast, Iguazio, Tecton 등은 상용 판매를 위한 범용 Feature 저장소 솔루션으로 개발되었으며, 나머지는 AI 모델에 기반하여 다양한 서비스를 제공하는 AI 기업들이 자사의 AI 기반 서비스 개발을 위해 운영하는 전용 플랫폼에 적용되었다.

이 장에서는 Feature 저장소의 전체적인 기술 동향을 개략적으로 설명하기 위하여, 널리 알려진 Feature 저장소 가운데 특별히 주목할 만한 의미가 있는 3개를 선정하여 각각의 주요 현황을 소개하는데, 먼저 최초의 Feature 저장소로서 이후 기술적으로 다른 Feature 저장소 개발에 큰 영향을 끼친 Michelangelo 플랫폼의 Feature 저장소를 살펴보고 두 번째로 최초의 Feature 저장소 공개 SW 프로젝트이면서 다른 Feature 저장소 대비 기술적인 완성도가 높은 Hopsworks Feature 저장소를 설명하며 마지막으로 또 다른 공개 SW 프로젝트로 개발이 진행되는 가운데 최근에 많은 주목을 받고 있은 Feast의 주요 현황을 소개한다[1].

1. Michelangelo

2017년 9월 5일에 최초 공개된 Uber의 Michelangelo는 Uber가 수많은 AI 모델을 개발·배포·운영하는 과정에서 직면했던 문제들을 해결하기 위한 AI 서비스 개발 플랫폼으로서, End-to-end AI 워크플로우 설정을 통해 AI 응용 서비스를 개발하는 방식으로 설계되었으며, 현재 Uber 내부의 모든 AI 모델 훈련과 서비스 배포·운영에 활용되고 있다.

Michelangelo는 기본적으로 다양한 공개 SW를 기반으로 구축하였는데, 공개 SW를 통해 Uber의 요구사항을 충족할 수 없는 일부 기능에 대해서는 Built in-house 컴포넌트들을 직접 개발하여 공개 SW와 조합하는 방식으로 구성되었으며, Michelangelo를 구성하는 주요 공개 SW 컴포넌트는 HDFS, Spark, Samza, Cassandra, MLLib, XGBoost, Tensorflow 등이 있다.

분류, 회귀, 시계열 예측을 포함하는 Uber의 모든 AI 모델 개발 과정은 예외 없이 1) 데이터 관

리, 2) 모델 훈련, 3) 모델 검증, 4) 모델 배포, 5) 예측, 6) 성능 모니터링 등 6단계의 워크플로우를 따르며, 이러한 워크플로우 기반 AI 서비스 개발 방식은 구체적인 구현 기술과 독립적이고, 새로운 알고리즘 타입 및 ML 프레임워크 도입에 따른 기능 확장이 쉬우며, Online, Offline 등 서로 다른 배포 모드에도 동일하게 적용이 가능하다는 장점과 함께 AI 서비스의 확장성, 안정성, 재생산성, 사용 편의성 및 자동화된 기능 제공 등의 효과를 가지게 된다.

특히, Uber는 1) 데이터 관리 단계에서 좋은 Feature를 찾는 것과 더불어 데이터 소스로부터 해당 Feature를 안정적으로 생성하고 제공하는 Feature 파이프라인을 구축하고 관리하는 것이 전체 AI 프로젝트 진행에서 가장 어렵고 또 큰 비용이 요구되는 작업이라는 사실에 주목하고, Uber의 기존 데이터 인프라에 Feature 저장소 개념을 구현한 솔루션을 최초로 적용하였다.

Uber는 이전까지 수없이 많은 AI 모델 개발의 경험을 통해 서로 다른 다양한 AI 모델 개발에서 동일하거나 비슷한 Feature들이 활용되는 사례가 빈번한 사실에 주목하여, 특정 AI 모델 개발을 위하여 생성하는 모든 Feature들을 Feature 저장소에 저장하고 다른 AI 모델 개발 과정에서 저장된 Feature를 공유할 수 있도록 하였으며, 모델 개발에 적용된 Feature와 Serving에 제공되는 Feature가 동일한 변환 연산 절차를 거쳐 생성되는 것과 더불어 두 Feature 데이터 사이의 통계적 특성의 일치성을 유지하기 위하여 Online/Offline Feature 파이프라인을 구축하고 이를 통해 생성된 Feature들을 각각 Online(Cassandra), Offline(Hive) Feature 저장소에 저장·관리한다.

Feature 저장소에 저장된 Feature는 정규 이름을 참조하는 방식으로 접근하는데, 이를 통해 모델 학습이나 배치 예측에 필요한 정확한 HDFS 데이터 세트들을 구성하고, Online 예측을 위해서는 Cassandra로부터 정확한 Feature를 가져올 수 있다. 또한, Feature 저장소에 저장된 Feature들은 주기적으로 자동 갱신되며, 새롭게 생성된 Feature는 지속적으로 추가하면서 Online/Offline 저장소의 통계적 일치성 유지를 지원한다.

Feature 파이프라인을 통해 생성되거나, 클라이언트 서비스로부터 수신한 Feature들은 모델 학습에 적합하지 않은 형식이거나 결측값이 있을 수 있고, 때로는 모델이 주어진 Feature 들의 일부만을 사용하는 경우가 있는데, 이러한 문제를 해결하기 위하여 모델을 학습하고 예측하는 시점에 모델에 제공되는 Feature들을 적절하게 설정할 수 있는 DSL(Domain Specific Language)을 정의하였다.

Uber가 Michelangelo를 개발하던 초기에는 확장성 있는 모델 학습과 배포된 모델의 운영 문제 그리고 Feature 공유 문제 해결에 집중하였던 반면에, 최근에는 개발자의 AI 모델 생산성(초기의 아이디어 단계에서 상용 모델을 운영하기까지의 소요 시간) 향상에 주력하고 있다. 또한 주어진 예측 문제를 해결하기 위해 Feature 저장소를 검색하여 가장 유용하고 중요한 Feature들을 자동으로 식별할 수 있는 기능 구현을 진행하고 있다.

Uber의 Feature 저장소는 오랜 AI 서비스 개발 경험에서 마주했던 수많은 실제적인 문제들을 해결하는 아이디어의 결과물이었기에 추후 Netflix, Twitter, Airbnb 등의 주요 AI 기업들이 각자의 Feature 저장소를 개발하고 자사의 AI 서비스 개발에 적용하는 과정에도 큰 영향을 미치게 되었다. 또한 Michelangelo Feature 저장소 개발팀이 주축이 된 기술 기업 Tecton을 통해 2020년 12월초 범용 Feature 저장소 솔루션을 개발하여 판매하기 시작했는데, 이는 이전까지 대부분의 Feature 저장소가 AI 서비

스 개발 플랫폼의 한 컴포넌트로 자리매김했던 것과는 달리 Feature 저장소가 하나의 독립된 패키지 상품이 되어 새로운 시장을 형성하는 주요한 사례가 되고 있다[9-11].

2. Hopsworks

Hopsworks는 Logical Clock 주도로 개발된 ML 모델 개발 및 운영을 위한 공개 SW 플랫폼으로서 Feature 저장소와 더불어 분산 POSIX-like 파일시스템 HopsFS, Pip, Conda, Tensorflow, Scikit-learn, Keras, Jupyter Notebooks, Tensorboard 등의 분산 ML&DL 도구, 모델 배포를 위한 Kubernetes, Kafka, Spark Streaming 모델 모니터링 도구 그리고 워크플로우 관리 도구로서 Airflow 등 다양한 공개 SW로 구성되어 있다.

이중 Hopsworks Feature 저장소가 가장 중요한 컴포넌트인데, 2018년 12월 말 첫 버전 발표 이후 2020년 11월 12일 현재 Github를 통해 Hopsworks Feature 저장소 기반으로 신규 Feature 및 Feature Group, 훈련 데이터 생성 편의성 등을 제공하는 라이브러리로서 HSFS(Hopsworks Feature Store)를 발표하였다. HSFS 설계 시 가장 중요한 고려 사항 중 하나는 데이터 소스로부터 Feature를 생성하고 Feature 저장소에 저장하는 과정의 범용성을 확보하는 문제였다. Hopsworks가 Feature 저장소를 구현할 때 Uber의 Michelangelo를 참조하였는데, Michelangelo는 DSL을 통해 특정 모델에 적합한 Feature를 설정하는 방법을 제공하는 반면, Hopsworks는 특정 프레임워크나 언어로부터 독립적인 Feature Group(DataFrames in Spark or Pandas) 개념을 도입하여 Feature를 생성하였다[12].

Feature는 외부 DB 질의 등을 통해 원시 데이터를 획득하고, DataFrame API를 통해 생성한 후 Feature 저장소에 저장하기 전 반드시 데이터 검증을 거친다. DB의 접근 제어를 위해 하나 이상의 DB 생성이 가능한 것과 같은 개념으로 하나 이상의 Feature 저장소를 생성할 수 있는데, 이를 통해 전체 조직에서 공유하기 부담스러운 민감한 Feature들에 대한 별도 관리가 가능하다. 또한 Train/Test Feature와 함께 예측을 위한 Serving Feature를 제공하기 위한 내부 저장소로서 각각 Hive로 구현된 Offline Feature 저장소와 MySQL 클러스터로 구성된 Online Feature 저장소가 있으며, 시간을 거슬러 과거 시점의 Feature 제공 기능을 통해 시간이 경과된 이후 이전 시점을 기준으로 훈련 데이터를 생성하는 기능을 제공하고 있다.

Hopsworks Feature 저장소는 데이터 공학자와 데이터 과학자의 역할을 엄격히 구분하여 하둡 기반 데이터 레이크 또는 데이터 웨어하우스를 관리하는 데이터 공학자에게만 Feature 생성 권한을 부여하였으며, 데이터 과학자가 새로운 Feature가 필요한 경우에는 반드시 데이터 공학자와의 협업을 거치도록 규정하였다. 이는 Feature 생성에 관련되는 업무 절차와 Feature 활용에 필요한 업무 절차 사이에 현격한 차이가 존재하는 현실에서 End-to-end AI 파이프라인을 Feature 저장소를 중심으로 Feature 생성 파이프라인과 Feature 활용 파이프라인으로 구분하여 각각의 업무 특성을 서로 독립적으로 반영한 파이프라인을 구현할 수 있는 기반을 제공한다.

Hopsworks Feature 저장소는 최초의 공개 SW이면서 Feature 저장소의 전체 기능이 공개되어 상용 환경에서 활용이 가능한 유일한 Feature 저장소 솔루션이다. 또한 Spark 기반 플랫폼과 밀접하게 연관되어 동작하는데, 이는 Feature 생성을 위한 데이터 엔지니어링 작업들이 대부분 Spark 솔루션에 기반하고 있는 현실이 반영된 결과이다[12-14].

3. Feast

Feast는 인도네시아의 온라인 주문 및 결재 대행 업체인 Gojek이 자사 비즈니스 모델을 위해 Michelangelo AI 플랫폼과 유사한 플랫폼을 개발하기 위해 구글 클라우드 등과 협력하여 개발한 공개 SW Feature 저장소 솔루션으로서 2018년 개발 시작과 함께 2009년 초 최초 버전 발표 후 2020년 12월 4일 기준 최신 버전 v0.8.2를 발표하였으며, 2020년 11월 말부터 Linux Foundation의 LF AI & Data Foundation의 Incubation 프로젝트가 되면서 기술적으로 특정 기업 의존성이 배제되는 방향으로 개발이 진행되면서 Logical Clock 주도로 진행되는 Hopsworks Feature 저장소와는 차이를 보인다.

Feast는 Feature를 생성하기 위한 데이터 파이프라인을 이미 갖추고 있는 기관/조직에서 Feature를 저장하고, 운영 중인 AI 모델에 저장된 Feature를 제공하는 기능이 필요한 경우에 적합한 솔루션이다. 즉, Feast는 Feature를 생성하기 위해 데이터 소스로부터의 변환 연산 작업은 포함하지 않는다는 점에서 Michelangelo, Hopsworks Feature 저장소 등과는 차이가 있는데, Feast의 핵심 개념은 Feature 엔지니어링 작업과 Feature 사용 작업을 분리함으로써 새로운 AI 모델 개발을 위해 동일한 Feature 엔지니어링 작업을 반복하지 않고, Feature 카탈로그를 참조하여 Feature 저장소로부터 필요한 Feature를 선택하도록 함으로써 AI 프로젝트 비용을 절감하는 것이다.

현재 Feast는 Feature 저장 시 데이터 검증 기능, Feature 탐색 기능 등이 제공되지 않는 등 완성도가 다소 미흡한 측면이 있지만, LF AI & Data Foundation의 Incubating 프로젝트로서 향후 현재의 미흡한 기능 보완과 동시에 최적의 Feature 저장소 구현을 위한 개발자 커뮤니티 확산에 유리한 환경을 갖추고 있다는 측면에서 발전성이 유망하고, 특정 기업의 기술에 종속되지 않은 독립된 솔루션이라는 점에서 많은 기업이 AI 서비스 개발 시 활용할 것으로 예상되는 주목할 만한 Feature 저장소이다 [15,16].

Ⅳ. 결론 및 향후 과제

본 고에서는 데이터 공학 작업과 데이터 과학 작업 사이의 협업 환경 구축을 위한 Feature 저장소 도입의 필요성과 중요성을 언급하고 주요 Feature 저장소의 현황 소개를 통해 Feature 저장소의 기술 동향을 알아보았다.

데이터 플랫폼과 AI 플랫폼 사이의 협업 환경으로서 Feature 저장소는 서로 다른 AI 모델에서의 Feature 공유, Feature를 매개로 하는 데이터 플랫폼과 AI 플랫폼 사이의 체계적인 인터페이스 수립, End-to-end AI 워크플로우가 Feature 저장소를 중심으로 Feature를 생성하는 워크플로우와 Feature를 활용하는 워크플로우의 분리 등을 통해 AI 프로젝트 비용 절감, AI 모델의 성능과 신뢰성 및 모델 운영 편의성 향상 등의 효과를 가져 온다.

AI 프로젝트 수행에 있어서 데이터 플랫폼과 AI 플랫폼 사이의 협업은 그 필요성과 중요성에도 불구하고 상대적으로 다른 AI 관련 이슈 대비 큰 주목을 받지 못하는 것은 협업이 다소 추상적이고 선언적 성격의 개념이어서 기술적으로 구체적이고 명확한 정의가 수립되지 못한 원인과 함께 협업의 당사자로서 주로 데이터 관련 작업을 하는 데이터 공학자와 AI 모델 관련 작업을 하는 데이터 과학자 사이의 업무 목표, 각자의 배경지식, 업무 처리 방식 등의 차이로 인한 상호 이해 부족 등에 기인하고 있다.

향후 과제로는 데이터 공학 작업과 데이터 과학

작업 사이의 협업 환경 구축에 필요한 요구사항 및 사양을 기술적으로 명확하게 구체화하고, 이를 바탕으로 Feature 저장소를 비롯한 기존 기술 활용 및 신규 기술 개발을 통한 협업 환경 제공 솔루션을 개발하여 AI 모델 개발 비용 절감과 AI 모델 성능 및 신뢰성 향상에 기여하는 것이다.

참고문헌

[1] M. Taifi, "ML feature stores: A casual tour 1/3," Apr. 13, 2020, https://medium.com/@farmi/ml-feature-stores-a-casual-tour-fc45a25b446a

[2] G. Osin, "Feature store as a foundation for machine learning," Dec. 10, 2020, https://towardsdatascience.com/feature-store-as-a-foundation-for-machine-learning-d010fc6eb2f3

[3] J. Dowling, "MLOps with a feature store," Mar. 5, 2020, https://towardsdatascience.com/mlops-with-a-feature-store-816cfa5966e9

[4] AltexSoft, "What is data engineering: Explaining the data pipeline, data warehouse, and data engineer role," June 2019, https://www.altexsoft.com/blog/datascience/what-is-data-engineering-explaining-data-pipeline-data-warehouse-and-data-engineer-role/

[5] 정창헌, "Data warehouse vs. data lake," 2017. 7. 12. https://blog.b2en.com/253

[6] J. Dowling, "Feature store vs data warehouse," Oct. 10, 2020, https://medium.com/data-for-ai/feature-store-vs-data-warehouse-306d1567c100

[7] J. K. Park and H. S. Hwang, "Feature store란?," Sept. 24, 2020, https://medium.com/daria-blog/feature-store-%EB%9E%80-4c53d18c1149

[8] http://featurestore.org/

[9] A. Holler and M. Mui, "Evolving michelangelo model representation for flexibility at scale," Oct. 16, 2019, https://eng.uber.com/michelangelo-machine-learning-model-representation/

[10] J. Hermann and M. Del Balso, "Meet michelangelo: Uber's machine learning platform," Sept. 5, 2017, https://eng.uber.com/michelangelo/

[11] M. Del Balso, "Tecton: The data platform for machine learning" Apr. 28, 2020, https://www.tecton.ai/blog/data-platform-ml/

[12] https://github.com/logicalclocks/hopsworks

[13] A. A. Ormenis et al., "Horizontally scalable ml pipelines with a feature store," Mar. 20, 2019, https://mlsys.org/Conferences/2019/doc/2019/demo_7.pdf

[14] https://hopsworks.readthedocs.io/en/latest/featurestore/guides/featurestore.html

[15] https://github.com/feast-dev/feast

[16] Feast, "What is feast," https://docs.feast.dev/