MLOps

- Feature Store

MLOps Feature Store

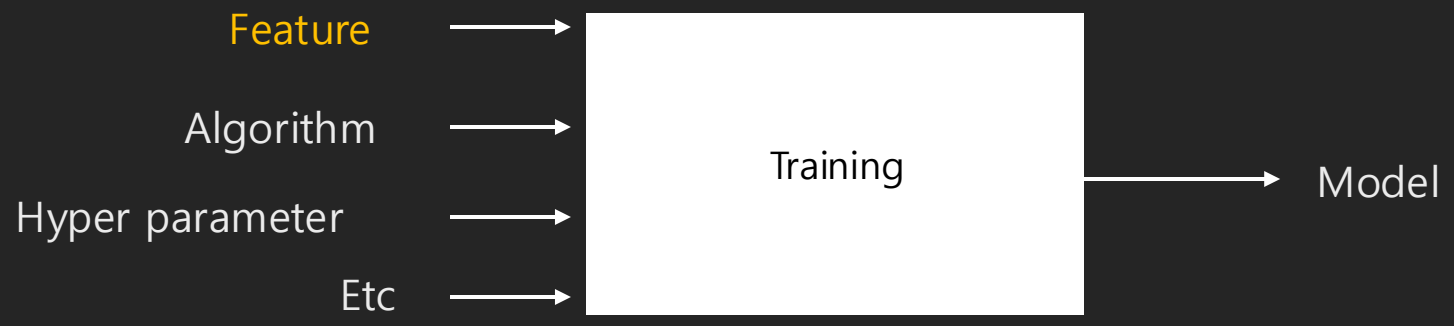
- 목차
  1. Feature 정의
  2. Feature Store 정의와 쓰는 이유

MLOps Feature Store

- Feature Store: Feature을 관리하는 시스템

MLOps Feature Store

- Feature란? AI 모델의 입력으로 사용되는 측정 가능한 데이터 속성



MLOps Feature Store

- 측정 가능한 데이터

```python
# 원시 텍스트 (측정 불가능)
raw_text = "이 제품은 정말 훌륭하고 추천합니다!"

# 측정 가능한 Feature로 변환
text_features = {
    'review_length': 23,              # 텍스트 길이 (측정 가능)
    'sentiment_score': 0.85,          # 감정 점수 (측정 가능)
    'positive_word_count': 2,         # 긍정 단어 개수 (측정 가능)
    'readability_score': 7.2          # 가독성 점수 (측정 가능)
}

# 텍스트 임베딩 벡터 (측정 가능)
text_embedding = [0.2, -0.1, 0.7, 0.3, ...]  # 512차원 벡터
```

MLOps Feature Store

- 데이터 속성

key

event_timestamp	dob_ssn	credit_card_due	mortgage_due	student_loan_due
2020-04-26T18:01:04.746Z	19530219_5179	8419	91803	22328
2020-04-26T18:01:04.746Z	19781116_7723	2944	741165	2515
2020-04-26T18:01:04.746Z	19931128_5771	833	976522	33000
2020-04-26T18:01:04.746Z	19500806_6783	5936	1553523	48955
2020-04-26T18:01:04.746Z	19620322_7692	1575	1067381	9501

value

MLOps Feature Store

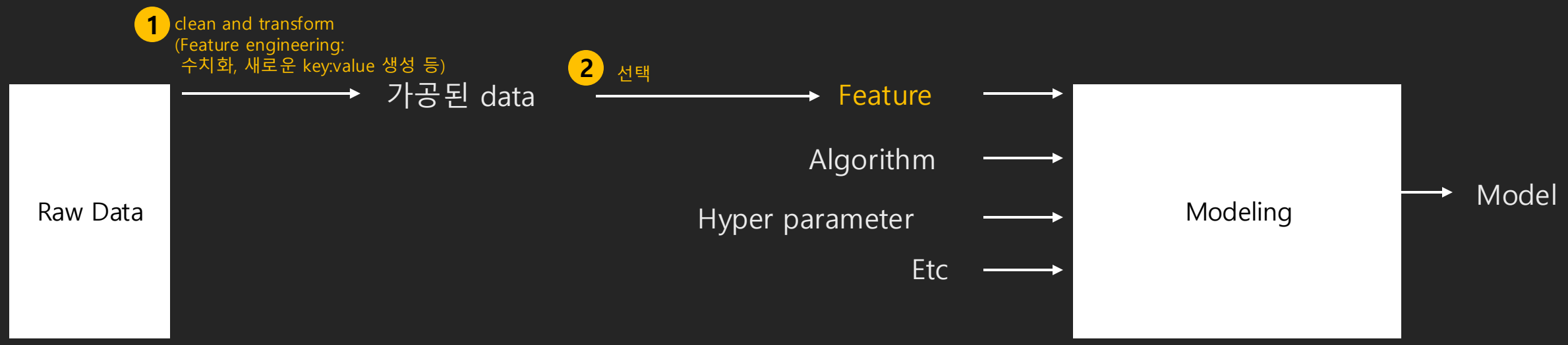
- 데이터 속성

```python
def predict(self, request):
    # Get online features from Feast
    feature_vector = self._get_online_features_from_feast(request)

    # Join features to request features
    features = request.copy()
    features.update(feature_vector)
    features_df = pd.DataFrame.from_dict(features)

    # Apply ordinal encoding to categorical features
    self._apply_ordinal_encoding(features_df)

    # Sort columns
    features_df = features_df.reindex(sorted(features_df.columns), axis=1)

    # Drop unnecessary columns
    features_df = features_df[features_df.columns.drop("zipcode").drop("dob_ssn")]

    # Make prediction
    features_df["prediction"] = self.classifier.predict(features_df)

    # return result of credit scoring
    return features_df["prediction"].iloc[0]
```

Dataset vs Feature

MLOps Feature Store

- Dataset vs Feature(학습에 사용되는 dataset)



Feature: 학습에 사용한 데이터셋

MLOps Feature Store

Feature store: Feature를 관리하는 시스템

storage

registry

serving

Feature Store를 왜 사용할까?

-> Feature를 왜 관리 해야할까?

-> 학습에 사용한 데이터셋을 왜 관리해야할까

MLOps Feature Store

- Feature Store 사용하는 첫번째 이유: Training-Serving Skew 문제 해결

Home > Products > Machine Learning > Guides > Rules of ML

Rules of Machine Learning:

Best Practices for ML Engineering

Training-Serving Skew

Training-serving skew is a difference between performance during training and performance during serving. This skew can be caused by:

- A discrepancy between how you handle data in the training and serving pipelines.
- A change in the data between when you train and when you serve.
- A feedback loop between your model and your algorithm.

참고자료: https://developers.google.com/machine-learning/guides/rules-of-ml#training-serving_skew

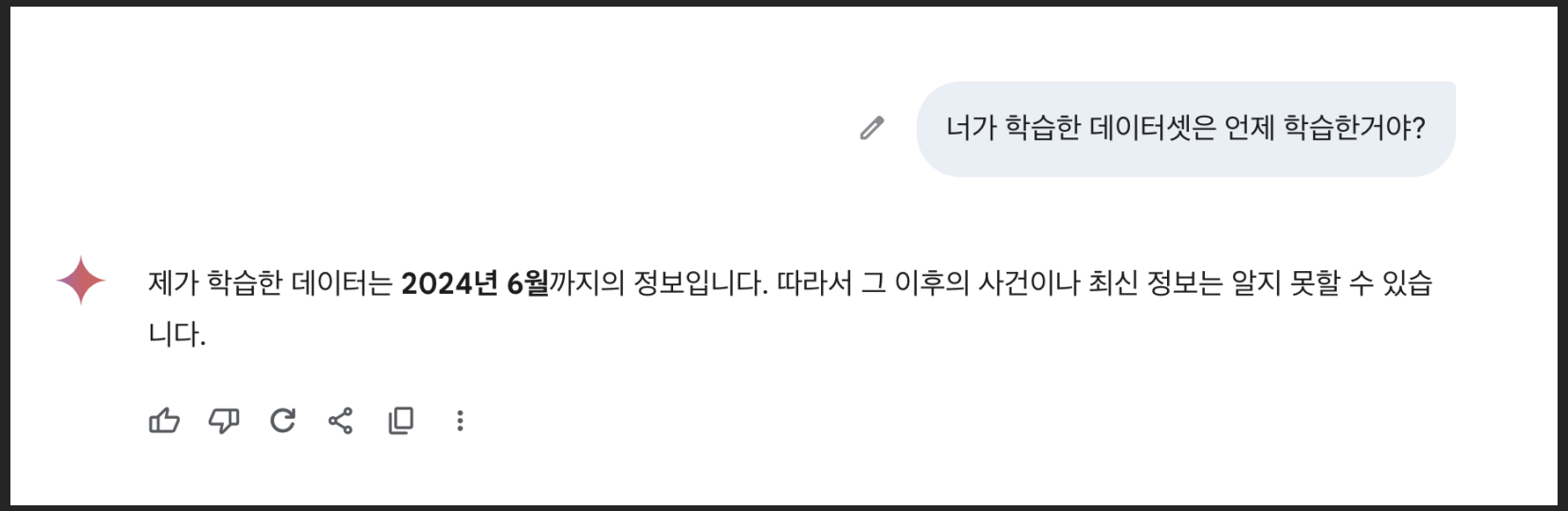
MLOps Feature Store

- Feature Store 사용하는 첫번째 이유: Training-Serving Skew 문제 해결

MLOps Feature Store

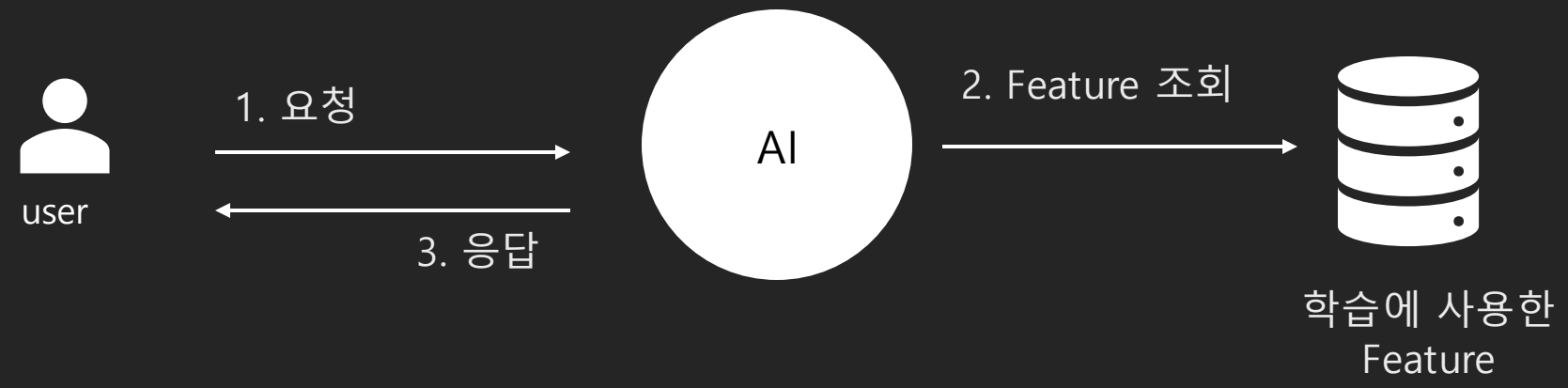
- Training-Serving Skew 문제 해결

[Gemini]



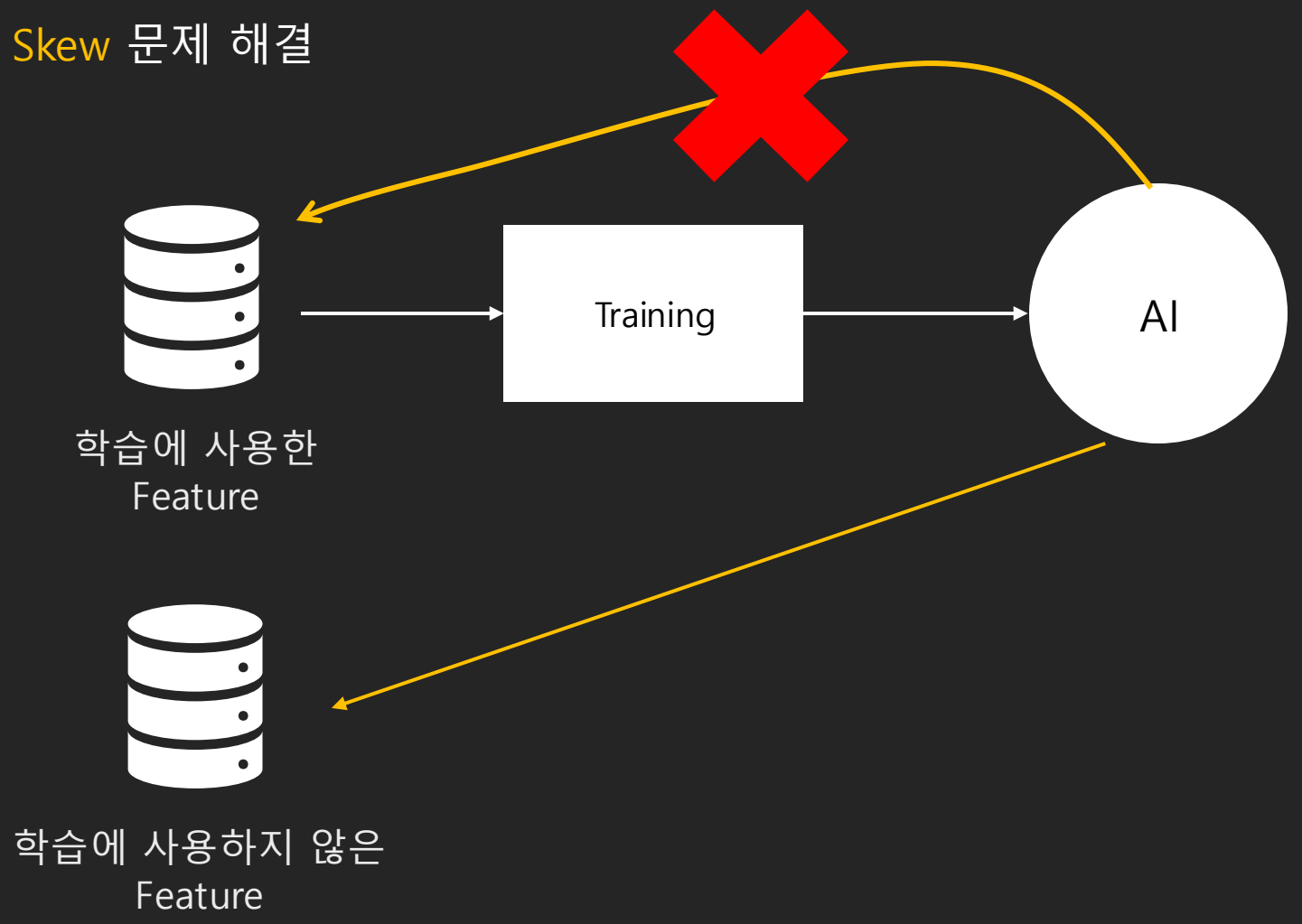
MLOps Feature Store

- Feature Store 사용하는 첫번째 이유: Training-Serving Skew 문제 해결



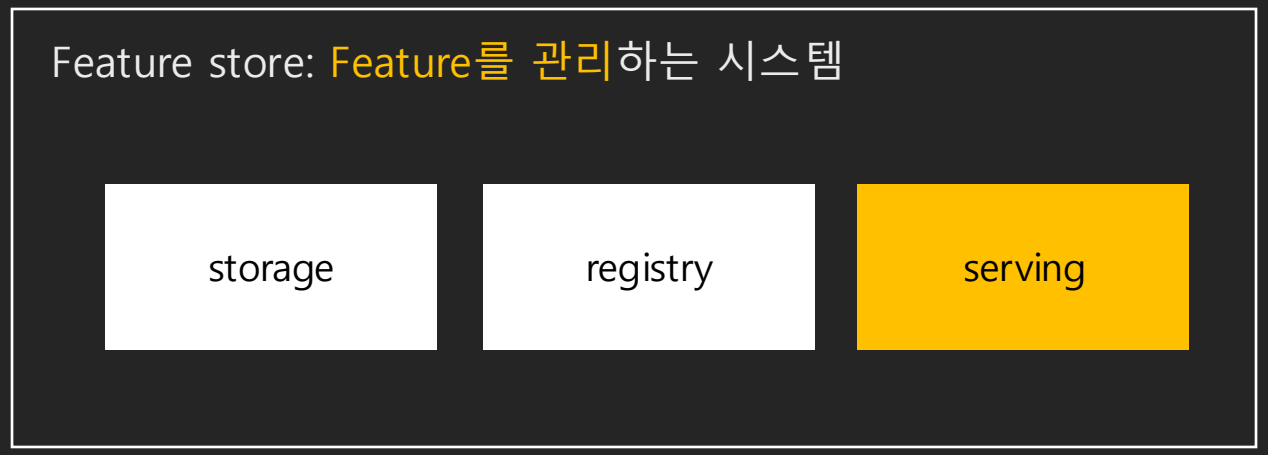
MLOps Feature Store

- Training-Serving Skew 문제 해결

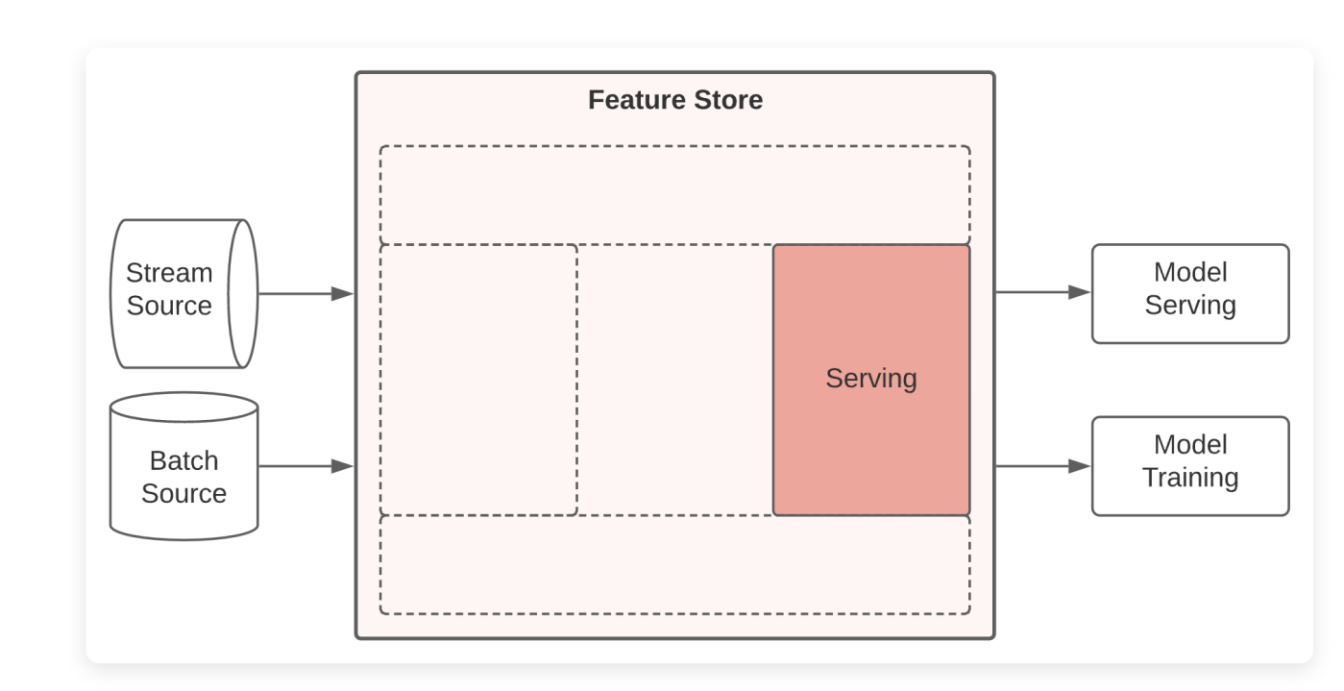


MLOps Feature Store

- Serving: 학습에 사용했던 Feature를 제공



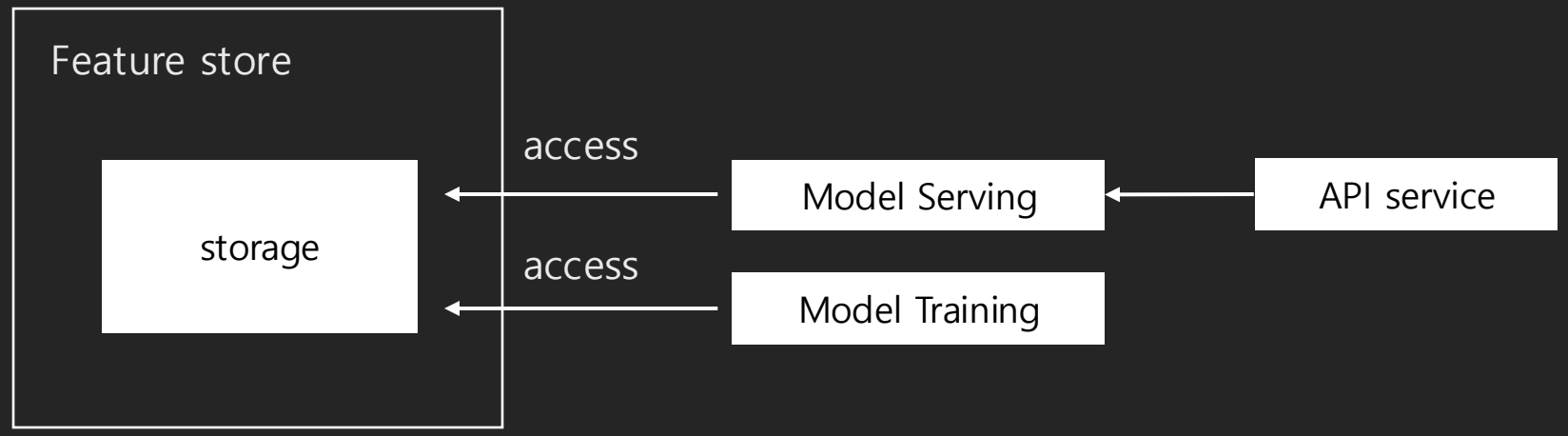
MLOps Feature Store



참고자료: https://feast.dev/blog/what-is-a-feature-store/

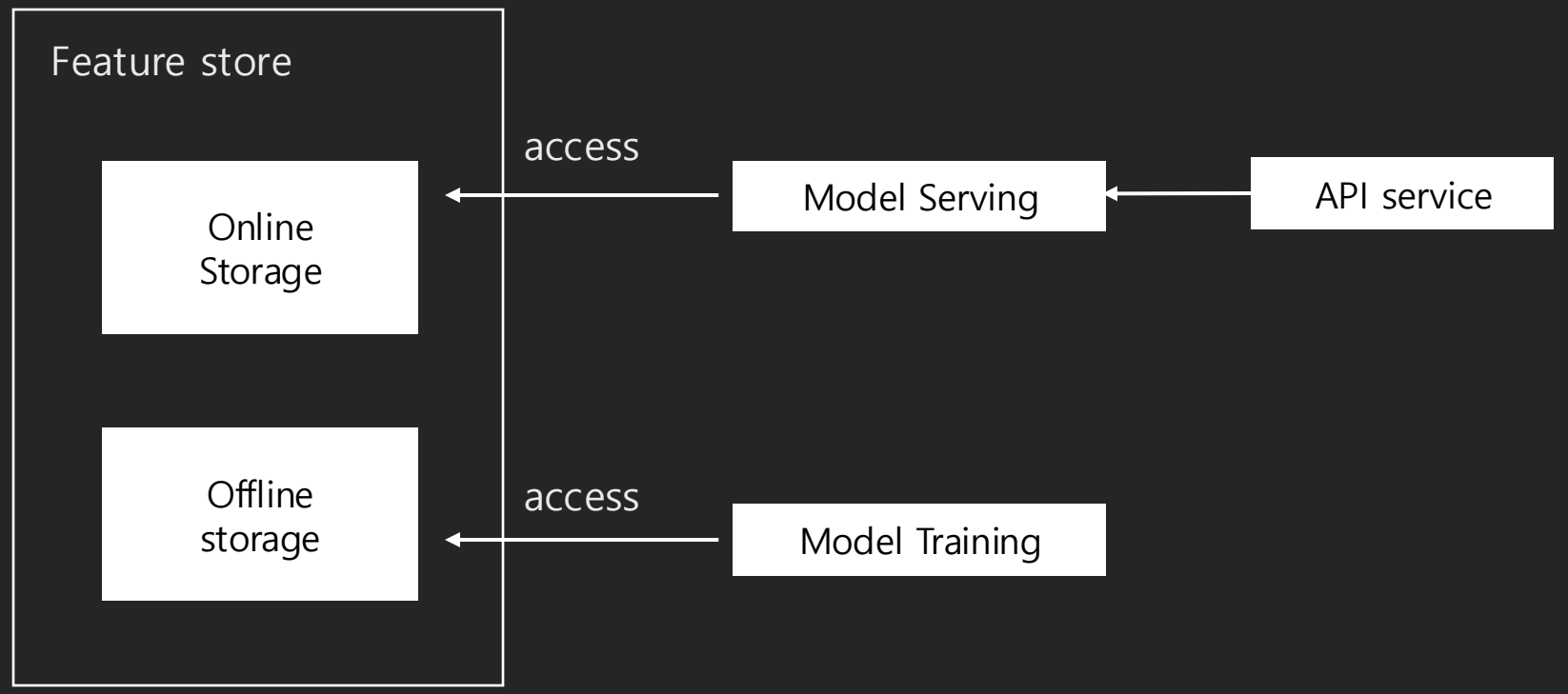
MLOps Feature Store

- AI, Training 모두 같은 storage를 사용한다면 성능이 저하



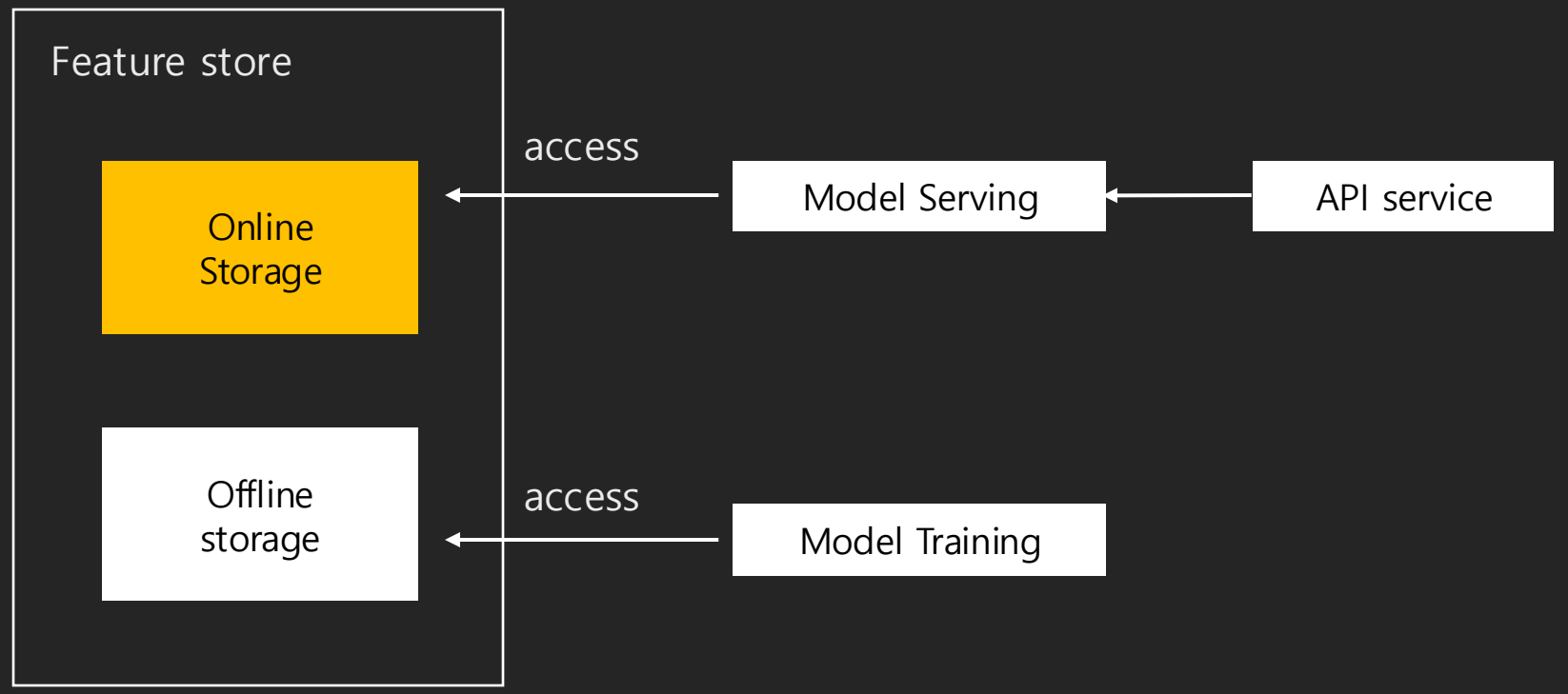
MLOps Feature Store

- 성능 개선위해 storage를 분리
  - Online Storage: Serving AI가 사용
  - Offline Storage: 학습에 사용



MLOps Feature Store

- Online Storage 후보:
  - AWS S3, AWS Dynamo, Redis
  - 특징: Latency가 빠름



MLOps Feature Store

- 설정 예:

```yaml
project: credit_scoring_aws
# use local proto registry
registry: registry.db
# if you are using a remote registry, you can specify it like this:
# registry:
#   path: s3://[YOUR BUCKET YOU CREATED]/registry.pb
#   cache_ttl_seconds: 60
provider: aws
entity_key_serialization_version: 3
online_store:
    type: dynamodb
    region: ap-northeast-2
offline_store:
    type: postgres
    host: localhost
    port: 5432
    database: feast
    db_schema: public
    user: feast
    password: password
```

MLOps Feature Store

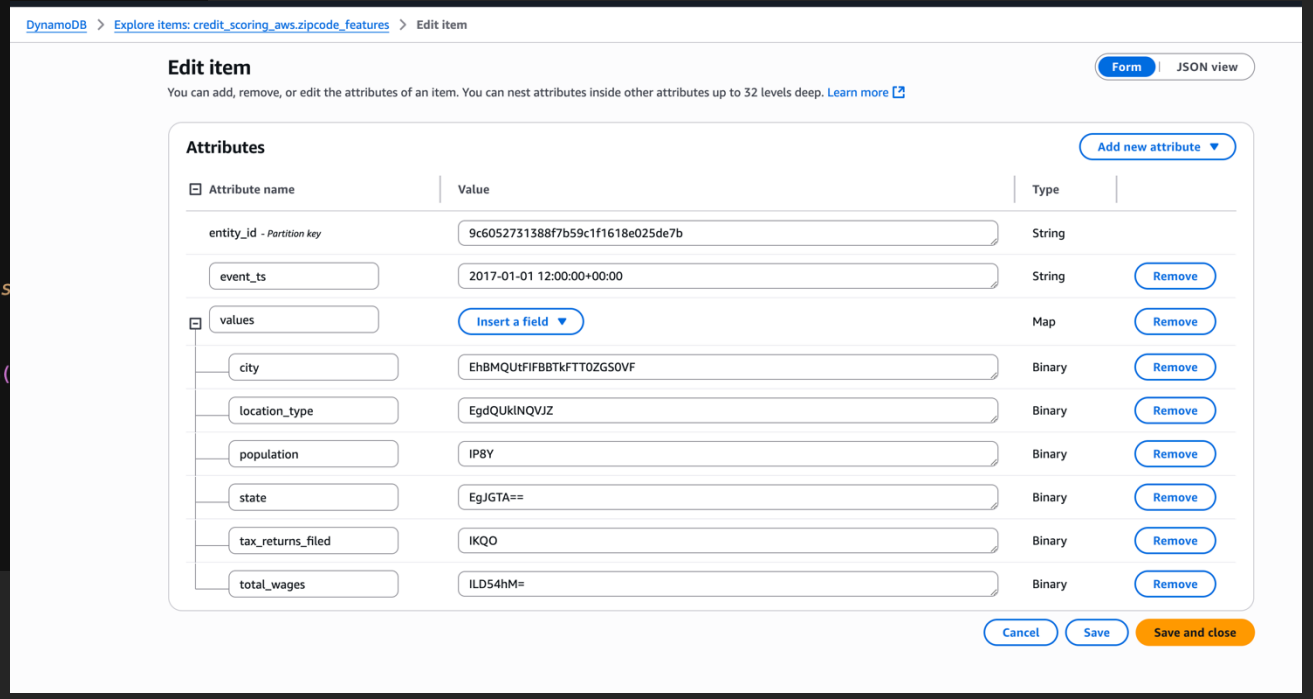
- 예제: 신용정보를 바탕으로 대출 가능/불가 판단

```python
# Make online prediction (using DynamoDB for retrieving online features)
loan_request = {
    "zipcode": [76104],
    "dob_ssn": ["19630621_4278"],
    "person_age": [133],
    "person_income": [59000],
    "person_home_ownership": ["RENT"],
    "person_emp_length": [123.0],
    "loan_intent": ["PERSONAL"],
    "loan_amnt": [35000],
    "loan_int_rate": [16.02],
}

result = model.predict(loan_request)

if result == 0:
    print("Loan approved!")
elif result == 1:
    print("Loan rejected!")
```

MLOps Feature Store

- 예제

```python
def predict(self, request):
    # Get online features from Feast
    feature_vector = self._get_online_features_from_feast(request)

    # Join features to request features
    features = request.copy()
    features.update(feature_vector)
    features_df = pd.DataFrame.from_dict(features)

    # Apply ordinal encoding to categorical features
    self._apply_ordinal_encoding(features_df)

    # Sort columns
    features_df = features_df.reindex(sorted(features_df.columns), axis

    # Drop unnecessary columns
    features_df = features_df[features_df.columns.drop("zipcode").drop(

    # Make prediction
    features_df["prediction"] = self.classifier.predict(features_df)

    # return result of credit scoring
    return features_df["prediction"].iloc[0]
```

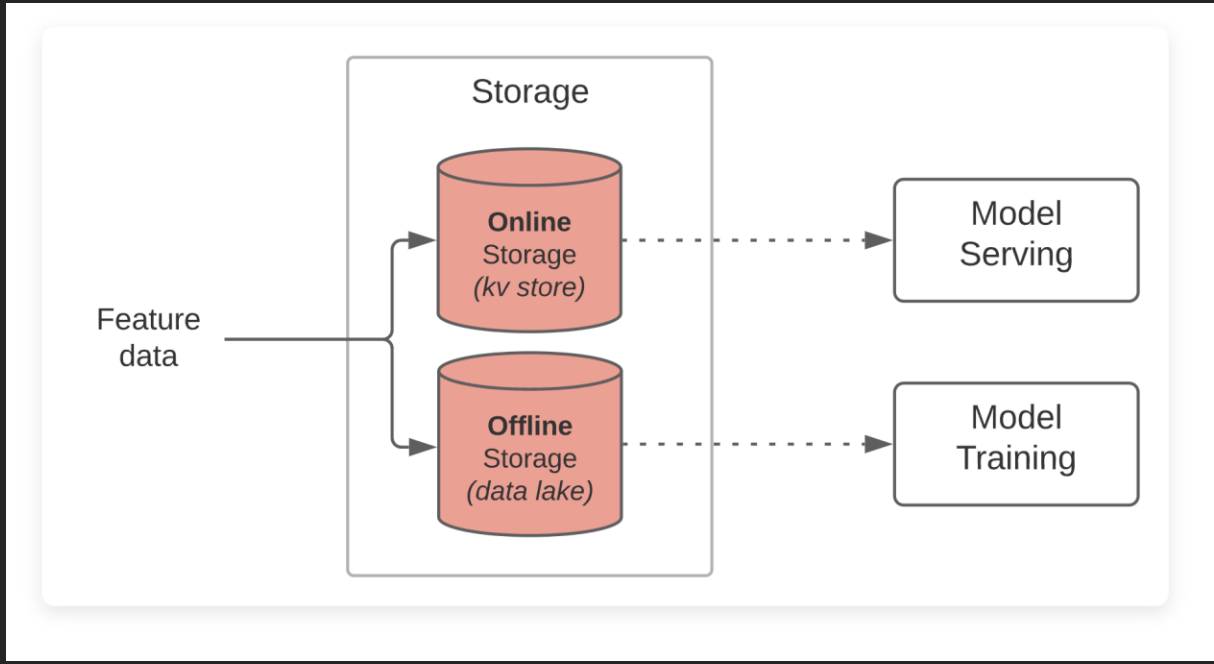


MLOps Feature Store

- 만약 online store가 이상있다면?

```
$ python predict.py
Traceback (most recent call last):
  File "/Users/choisungwook/git/portfolio/mlops/feast-aws-credit-scoring-tutorial/predict.py", line 26, in <module>
    result = model.predict(loan_request)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/choisungwook/git/portfolio/mlops/feast-aws-credit-scoring-tutorial/credit_model.py", line 100, in predict
    feature_vector = self._get_online_features_from_feast(request)
                     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/choisungwook/git/portfolio/mlops/feast-aws-credit-scoring-tutorial/credit_model.py", line 126, in _get_online_features_from_feast
    return self.fs.get_online_features(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

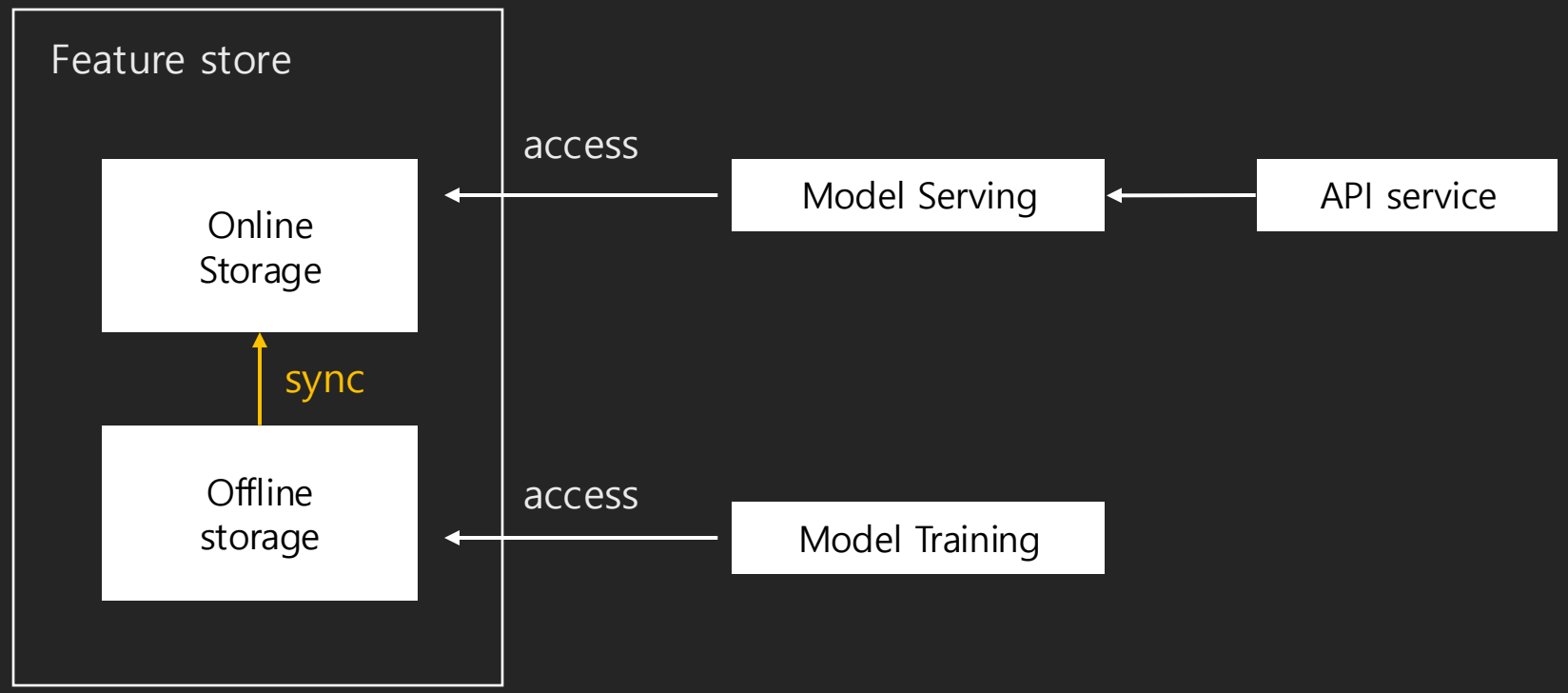
MLOps Feature Store



참고자료: https://feast.dev/blog/what-is-a-feature-store/

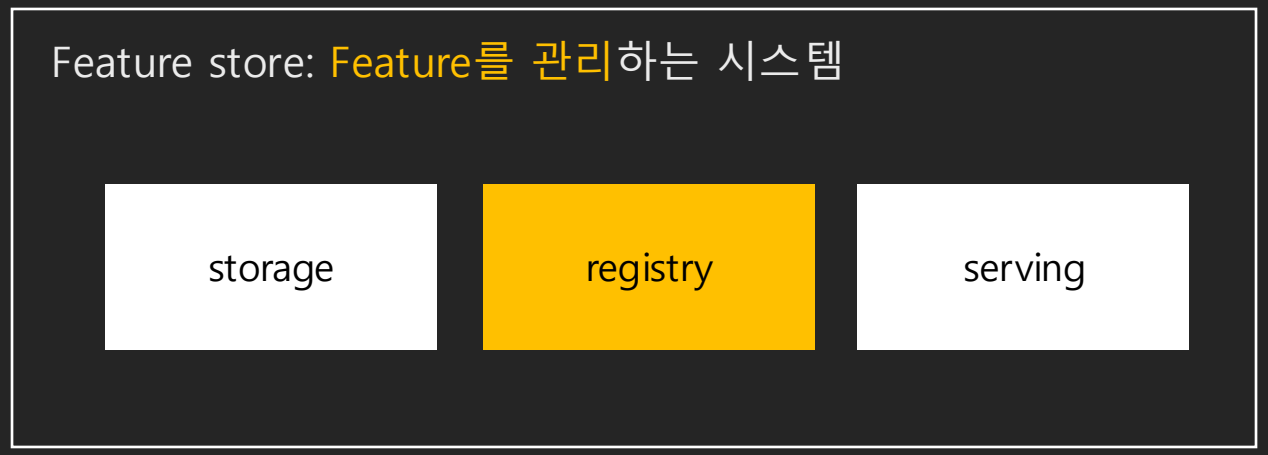
MLOps Feature Store

- sync작업이 필요



MLOps Feature Store

- Registry: Feature를 저장하기 위한 메타데이터와 구조를 저장



MLOps Feature Store

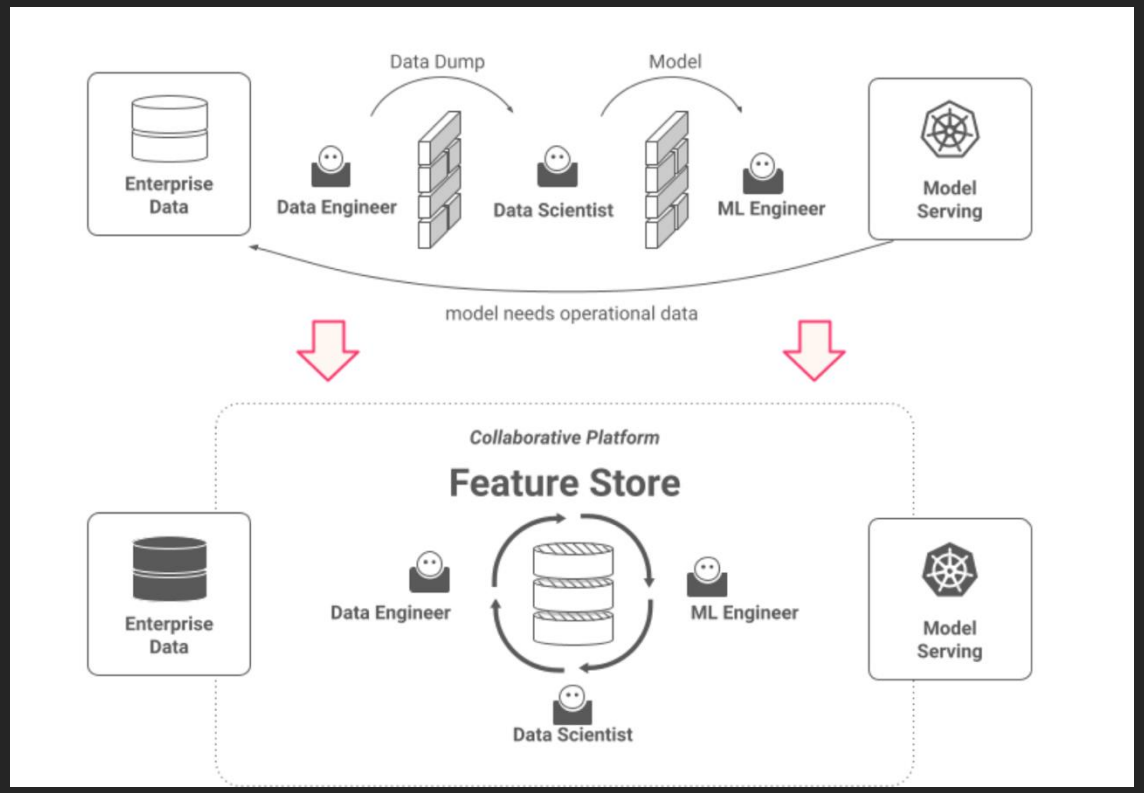
- Registry: Feature를 저장하기 위한 메타데이터와 구조를 저장

zipcode	city	state	location_type	tax_returns_filed	population	total_wages
7675	WESTWOOD	NJ	PRIMARY	13245	24083	1089095041
7677	WOODCLIFF LAKE	NJ	PRIMARY	2945	5471	325436960
7885	WHARTON	NJ	PRIMARY	5273	8999	240827990

```python
zipcode = Entity(name="zipcode", value_type=ValueType.INT64)

zipcode_source = PostgreSQLSource(
    name="zipcode_features",
    query="SELECT * FROM zipcode_features",
    timestamp_field="event_timestamp",
    created_timestamp_column="created_timestamp",
)

zipcode_features = FeatureView(
    name="zipcode_features",
    entities=[zipcode],
    ttl=timedelta(days=3650),
    schema=[
        Field(name="city", dtype=String),
        Field(name="state", dtype=String),
        Field(name="location_type", dtype=String),
        Field(name="tax_returns_filed", dtype=Int64),
        Field(name="population", dtype=Int64),
        Field(name="total_wages", dtype=Int64),
    ],
    source=zipcode_source,
)
```

MLOps Feature Store

- Feature Store 사용하는 두번째 이유: 협력
  - Feature 중앙관리
  - SDK, API 등으로 Feature 접근
  - Feature 일관성 유지
  - Feature 재사용
  - Feature 버저닝
  - Feature 추적
  - 기타 등등



참고자료: https://www.featurestore.org/what-is-a-feature-store

MLOps Feature Store

- Feature Store 사용하는 세번째 이유: 재현성(유사: Immutable)
  - 같은 코드, 같은 AI모델이어도 데이터가 다르면 재현 불가

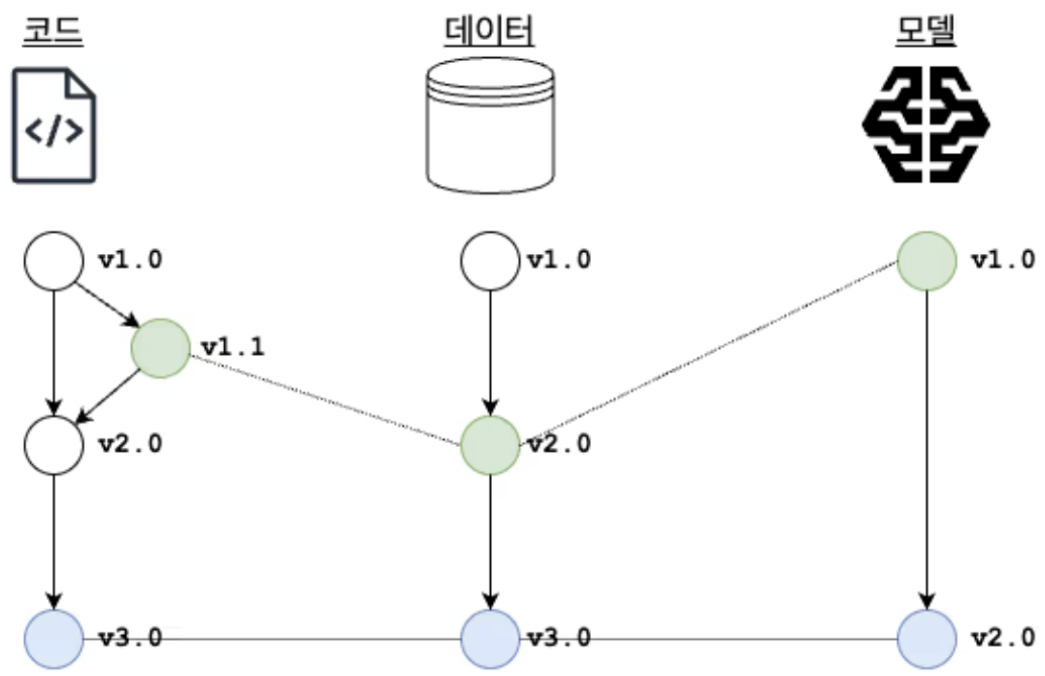


그림 4-10. 코드, 데이터, 모델의 버전 관리

참고자료: https://blog.taehun.dev/from-zero-to-hero-mlops-tools-4

MLOps Feature Store

- Feature store 솔루션 또는 오픈소스
  - AWS: SageMaker
  - SaaS: Databricks
  - On-premise: Feast

MLOps Feature Store

- Lesson Learn
  - Feature != Dataset
  - online store, offline store

MLOps Feature Store

- 더 공부할 내용
  - RAG, 인터넷 검색, MCP등 학습에 없던 Feature는 Training-Serving Skew를 발생시키지 않는가?
  - Observability