POLITECNICO
MILANO 1863

SYSTEMS AND METHODS FOR BIG AND UNSTRUCTURED DATA

Feature Store

Marco Brambilla

marco.brambilla@polimi.it

@marcobrambi

Agenda

POLITECNICO
MILANO 1863

Definition

ML-Ops

Feature store basics

Technologies

FEATURE STORE Overview

POLITECNICO
MILANO 1863

A feature store is a centralized repository used to store, manage, and serve features for machine learning (ML) tasks.

Features are the input variables or attributes that describe (data) objects used by machine learning models to make predictions, such as customer age, transaction history, or weather conditions.

The goal of a feature store is to streamline the process of creating, sharing, and using these features across various ML models, making it easier to develop, maintain, and deploy ML systems.

It’s a key asset in ML-Ops

POLITECNICO
MILANO 1863

ML-Ops

Marco Brambilla.

ML-Ops

POLITECNICO MILANO 1863

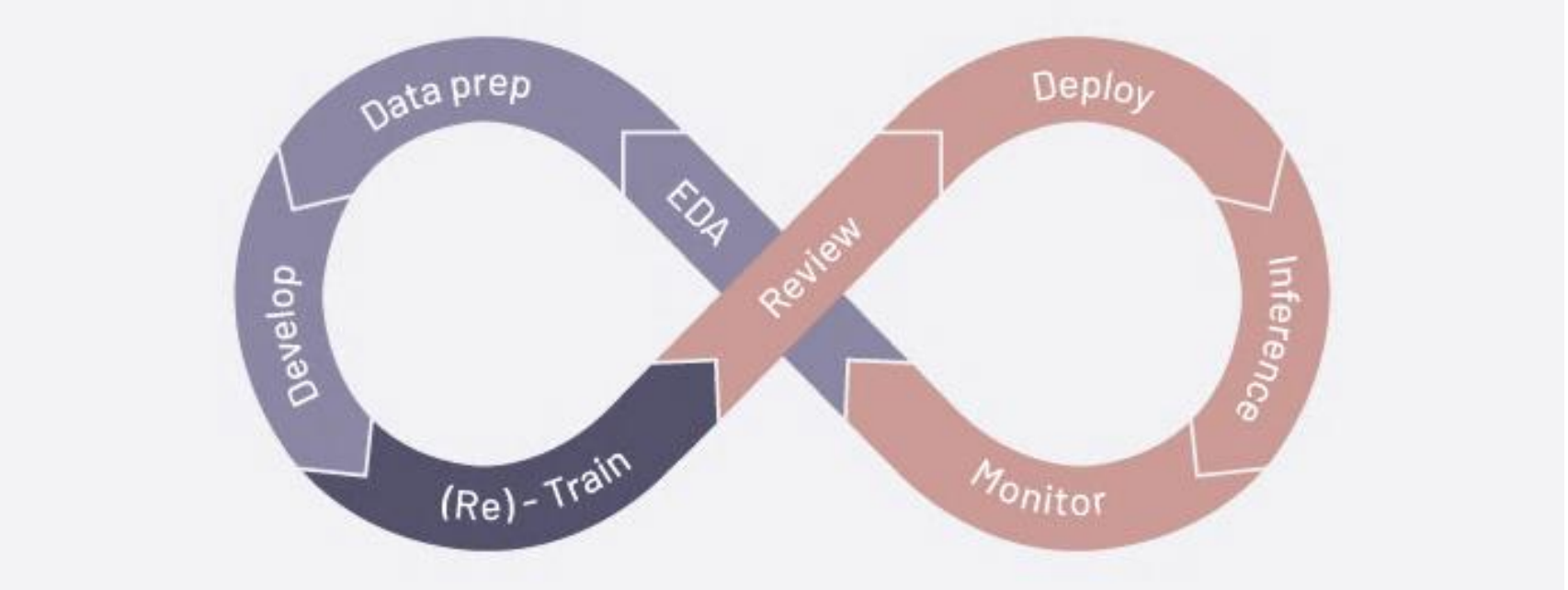
MLOps (Machine Learning Operations) is a set of practices and tools that aim to streamline the development, deployment, and maintenance of machine learning (ML) models in production environments.

It extends the principles of DevOps (Development and Operations) to the ML domain, focusing specifically on the unique challenges that arise in building, deploying, and monitoring ML systems at scale.

MLOps brings together data scientists, machine learning engineers, and operations teams to collaborate more effectively and ensure ML models can be reliably deployed and maintained.

ML-Ops (coarse) cycle

POLITECNICO
MILANO 1863

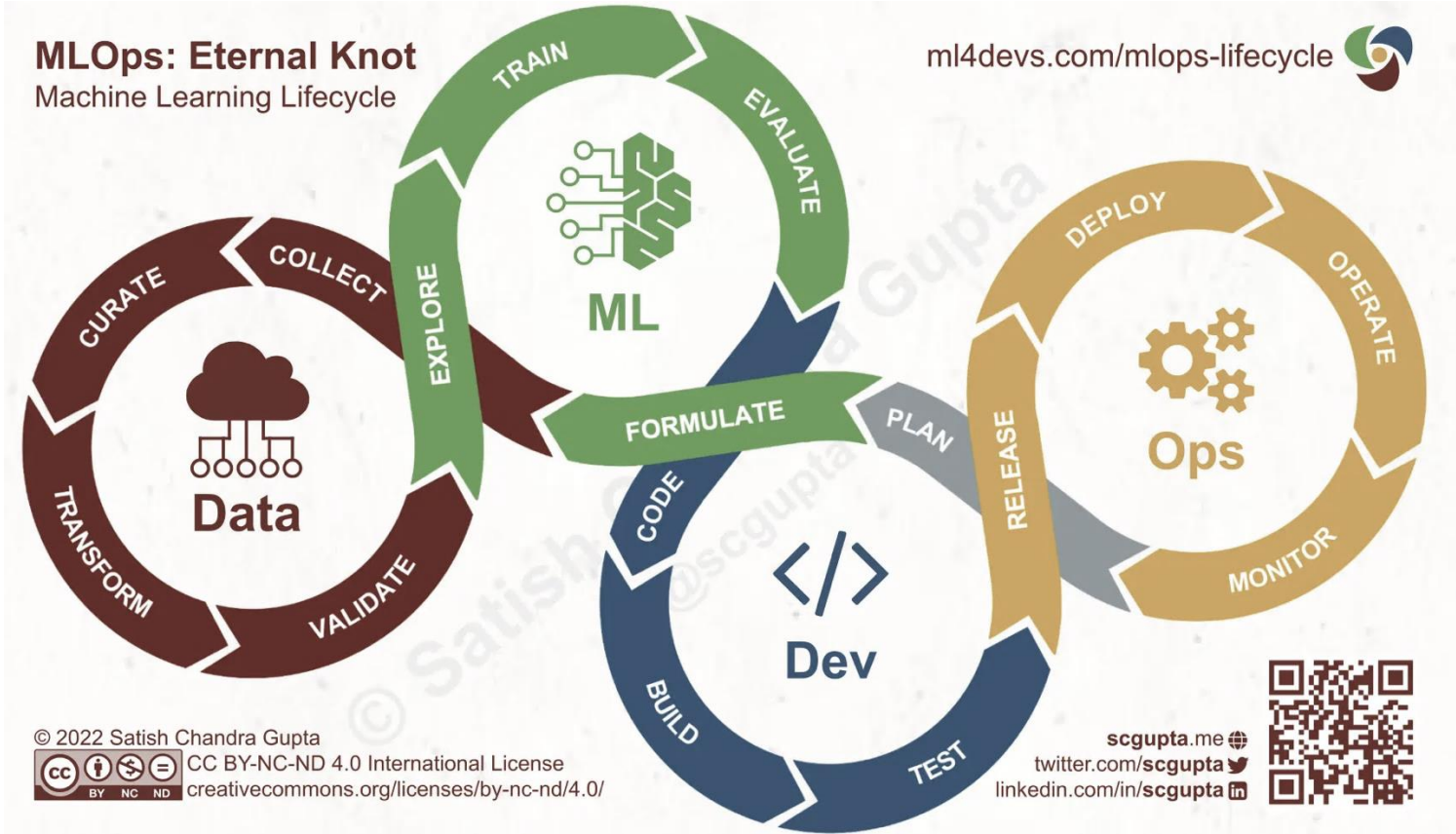


MLOps Lifecycle

POLITECNICO MILANO 1863

1. Development: Data scientists experiment with different models, algorithms, and feature sets. The code, models, and datasets are versioned, and automation is put in place to manage this process.
2. Training and Validation: The model is trained using data, and its performance is validated using a test dataset. MLOps helps automate these steps and ensures that the validation process can be reproduced.
3. Deployment: Once the model is validated, it is deployed into a production environment. MLOps uses CI/CD practices to ensure that this deployment process is automated and seamless.
4. Monitoring: After deployment, the model's performance is continuously monitored. MLOps tools provide real-time insights into how the model is performing, and whether data or concept drift is affecting its accuracy.
5. Retraining: When the model's performance drops or when new data becomes available, the model is retrained. This process is automated within MLOps pipelines to ensure that models are continuously updated as needed.

ML-Ops – focus on data and features



Components of ML-Ops

POLITECNICO MILANO 1863

1. **Version Control for Models and Data**:

Just like software, ML models and datasets need to be versioned to ensure reproducibility and traceability. MLOps helps track different versions of datasets, features, models, and hyperparameters. This ensures that a model can be reproduced with the exact same data and parameters used during training and testing.

2. **Continuous Integration/Continuous Deployment (CI/CD) for ML**:

Continuous Integration (CI): In MLOps, CI automates the process of testing and validating changes in models, data pipelines, and code. This ensures that models are trained on valid data and that any changes introduced don't break the existing system.

Continuous Deployment (CD): MLOps introduces automated pipelines that allow new models to be seamlessly deployed into production once they pass validation. This includes transitioning from a model training environment to real-world production systems.

3. **Model Training and Retraining**:

MLOps enables models to be automatically retrained when necessary (e.g., when there is a data drift or performance drop in the model). Automated pipelines can be set up to regularly retrain models with fresh data, keeping them up-to-date with current trends and behaviors in the production environment.

Components of ML-Ops

POLITECNICO MILANO 1863

4. **Monitoring and Model Performance Tracking**:

Once models are in production, MLOps involves continuous monitoring of their performance. This includes tracking key metrics such as accuracy, precision, recall, and business-specific KPIs. MLOps tools can detect data drift, concept drift, or model performance degradation and alert teams or trigger retraining workflows.

5. **Data and Feature Management**:

In ML projects, data and feature consistency are crucial for both training and production environments. MLOps integrates with feature stores to ensure that the same data transformations applied during training are applied in production, avoiding issues like “training-serving skew.” Feature versioning and validation are part of this process to ensure high data quality.

6. **Automation and Orchestration**:

MLOps automates many of the repetitive tasks involved in building, training, deploying, and managing models. These tasks include data preprocessing, model training, model evaluation, and deployment. Workflow orchestration tools like Kubernetes, Airflow, or Kubeflow help in managing complex pipelines that may involve multiple steps or dependencies.

Components of ML-Ops

POLITECNICO MILANO 1863

7. **Collaboration Between Teams**:

MLOps encourages collaboration between data scientists, machine learning engineers, and DevOps teams. By providing shared pipelines, documentation, and tools, MLOps bridges the gap between teams that may traditionally work in silos. This fosters better communication, faster iterations, and smoother handoffs from development to production.

8. **Model Governance and Compliance**:

MLOps ensures that models are compliant with regulatory and business standards. This includes auditing models, tracking changes to datasets and features, and enforcing policies related to data privacy, model explainability, and bias mitigation. Compliance features help organizations maintain transparency and accountability in their ML workflows.

Benefits of MLOps

POLITECNICO MILANO 1863

- **Faster Time to Market**: MLOps automates many steps in the ML lifecycle, allowing teams to deploy models faster and iterate more quickly.
- **Improved Collaboration**: By providing shared tools and workflows, MLOps fosters collaboration between data scientists, engineers, and operations teams.
- **Increased Model Reliability**: Continuous monitoring and automated retraining help ensure that models maintain high performance in production environments.
- **Reusability**: MLOps allows the reuse of code, pipelines, and features across different models and projects, reducing duplication of effort.
- **Better Model Governance**: With clear versioning, monitoring, and auditing processes, MLOps ensures that organizations can maintain regulatory compliance and manage the risk of deploying machine learning models.

MLOps wrap-up

POLITECNICO MILANO 1863

MLOps is essential for **scaling machine learning** from research and development into real-world production environments.

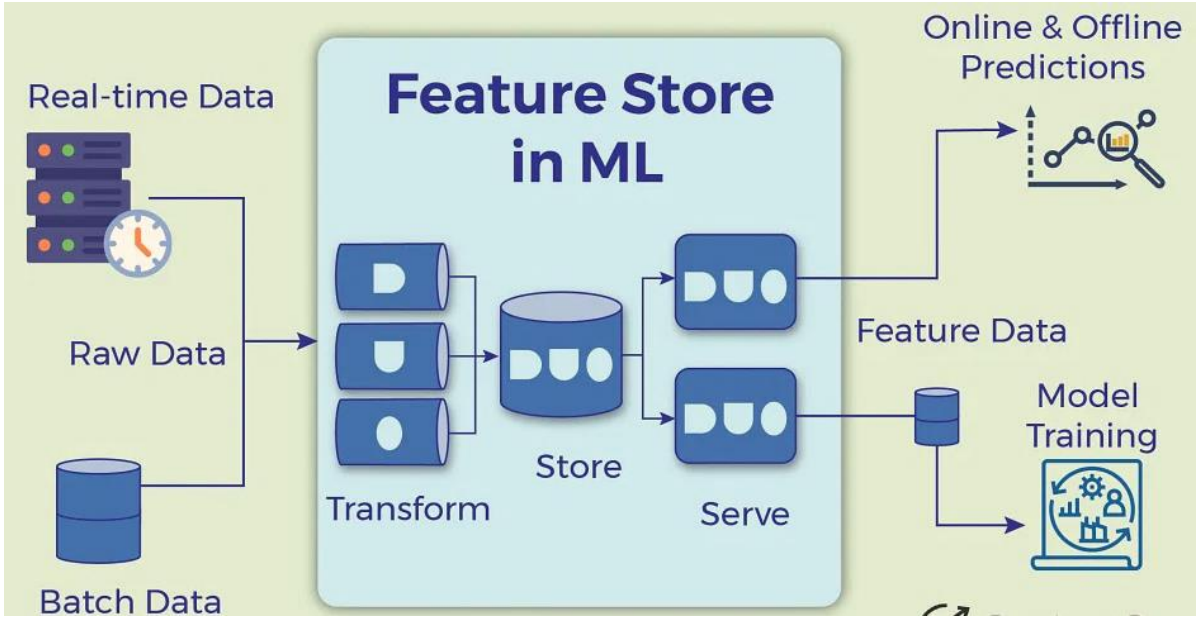
By **automating processes, ensuring reproducibility, and improving collaboration**, MLOps helps organizations successfully manage the complexities of operationalizing ML models at scale.

POLITECNICO
MILANO 1863

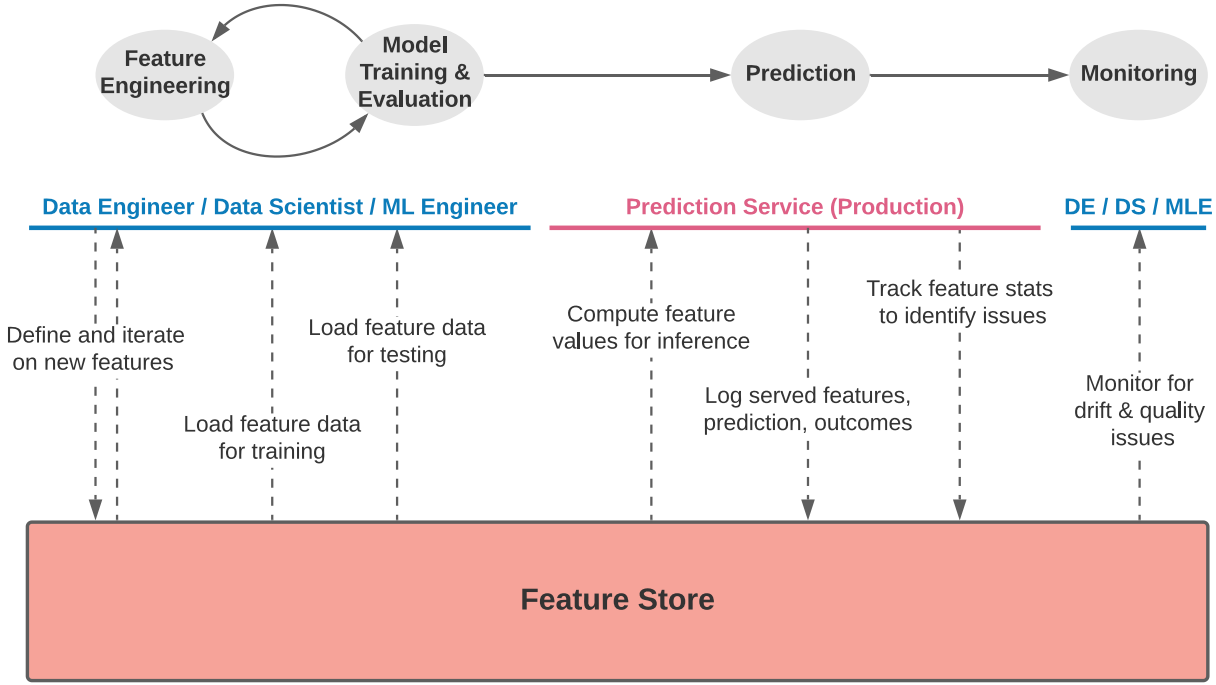
Feature Stores

Marco Brambilla.

Feature Store Overview



The feature store and the process



Key Components

POLITECNICO MILANO 1863

1. **Feature Storage**:

The feature store holds the data for features in a consistent format, often organized as structured datasets. It can store both historical features (used for training models) and real-time features (used for serving live predictions). Feature data can be stored in different databases, such as NoSQL databases for real-time features or distributed file systems for batch features.

2. **Feature Engineering Pipeline**:

A feature store often includes tooling or pipelines to transform raw data into usable features. These pipelines automate the process of feature extraction, transformation, and validation. The feature engineering steps can be complex, including aggregations, normalization, one-hot encoding, or any other data manipulation needed to convert raw inputs into usable features.

3. **Feature Versioning**:

A feature store tracks versions of features to ensure consistency between training and serving environments. This versioning allows teams to keep track of different iterations of a feature (e.g., changes to how a feature is calculated or updated). This is critical in avoiding “training-serving skew,” where a model performs poorly because the features used in production differ from the ones used during training.

Key Components

POLITECNICO MILANO 1863

4. **Online and Offline Stores**:

• **Offline Store**: Contains batch features used for model training. These are typically updated at regular intervals (e.g., daily or weekly) and stored in large-scale data storage systems like Hadoop or cloud-based data lakes.

• **Online Store**: Contains real-time features used for serving live predictions. These features need to be served with low latency for real-time inference and are typically stored in low-latency databases like Redis.

5. **Feature Serving**:

The feature store is responsible for serving features to different environments, like training environments (batch mode) and production environments (real-time). In production, the store must provide the most up-to-date features quickly, enabling models to generate predictions with minimal latency.

6. **Data Governance & Access Control**:

Feature stores often come with built-in mechanisms for controlling access to features, ensuring compliance with data governance policies. This allows organizations to enforce rules around data usage, such as who can create, access, or update certain features.

Key Components

POLITECNICO MILANO 1863

7. **Feature Discoverability and Reusability**:

One of the major benefits of a feature store is the ability to reuse features across multiple models. A feature store typically includes a catalog or search functionality that allows data scientists and ML engineers to discover existing features instead of creating new ones from scratch. This fosters collaboration and prevents duplication of effort.

8. **Monitoring and Validation**:

Feature stores monitor the health of features by validating data quality, ensuring that data drifts or anomalies are detected early. This helps maintain the quality of features used in both training and serving environments, preventing performance degradation of ML models due to corrupted or outdated data.

Benefits

POLITECNICO
MILANO 1863

Consistency: Ensures that the same feature transformations are applied in both training and production environments, reducing discrepancies between model training and deployment.

Efficiency: By centralizing features, teams can avoid redundant feature engineering work, speeding up the development process and encouraging collaboration.

Scalability: Handles large-scale feature computation and management for both batch and real-time features, allowing models to scale without sacrificing performance.

Version Control: Tracks changes in features, which ensures that models trained on older versions of features can still be reproduced, and new versions of features can be rolled out in a controlled way.

Usage Example

POLITECNICO MILANO 1863

In a credit scoring application, a feature store could store various features such as:

- Customer demographics (age, income, etc.)
- Historical credit behavior (loan repayments, defaults)
- Real-time transaction data (recent purchases, account balance changes)

The feature store will make it easier to ingest, integrate, preprocess, clean, version, the features

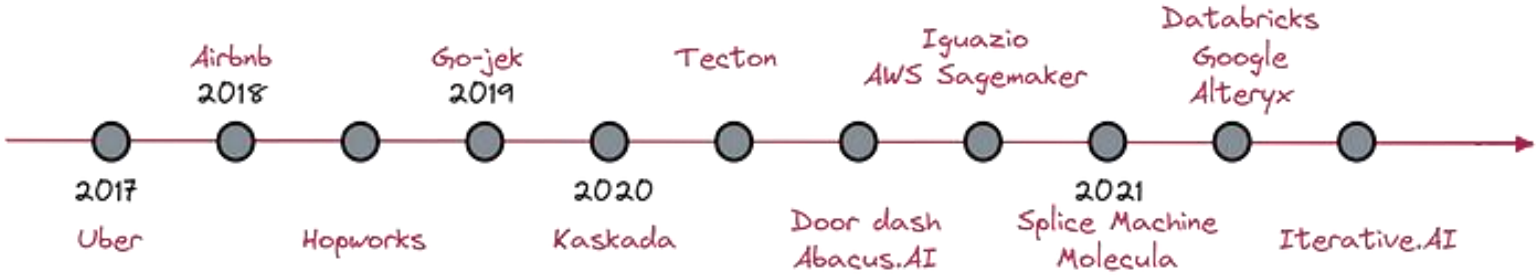
For training, the offline store would provide historical data, while for real-time predictions, the online store would supply the latest transaction data.

This way, the same features are **reused across all models consistently**, reducing engineering overhead and improving the efficiency of model deployment.

All in all, feature stores represent a strategic asset for ML-ops activities

History

First Experience in Uber, with Michelangelo platform.



Feature Management

POLITECNICO
MILANO 1863

Besides importing, preprocessing, and transferring data:

FEATURE ENGINEERING

Feature Engineering

POLITECNICO MILANO 1863

DEF: a machine learning approach that **extracts features from a raw dataset, resulting in new features**.

Feature engineering ranges from basic transformations such as aggregations to more advanced feature transformations such as word embeddings produced by machine learning algorithms.

The goal of feature engineering is to essentially **create a better dataset to improve the performance of machine learning** algorithms.

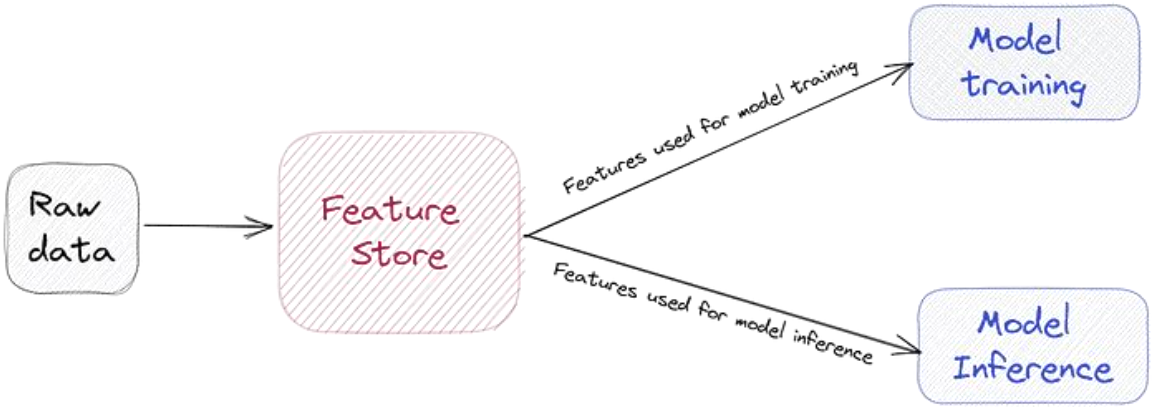
Feature Store role

a centralised repository that stores **curated features**.

a data management layer that allows data scientists, machine learning engineers, and data engineers to collaborate, share, and discover features.

An interface between the raw data and the models. It takes raw data and then transforms it into features subsequently used for model training and inference.

ensures that the features used across both models (and possibly many models) are consistent.



The biggest problem to face

POLITECNICO MILANO 1863

Features for a machine learning model are required in both the modelling and production (deployment) phases of the lifecycle.

This implies duplication of code and risk of inconsistency

A feature store grants:

- Consistency between model training and inference leads to better model accuracy
- eliminating online/offline skew
- Share and reuse features across models and teams, ensuring faster model development
- Provide feature versioning, lineage, provenance, and regulatory compliance

When needed

POLITECNICO
MILANO 1863

Feature stores are a must-have when working on a project where the intention is to deploy models at scale and in production

However, if you are working within a small team and on a small project or a POC, you may not need it

POLITECNICO
MILANO 1863

Technologies

POLITECNICO
MILANO 1863

1. Feast (Feature Store for Machine Learning)

• **Type**: Open-source

• **Key Characteristics**:

• **Online and Offline Stores**: Supports both real-time feature serving (low-latency online store) and batch feature serving (offline store).

• **Cloud-agnostic**: Can be deployed on any cloud provider or on-premise.

• **Integration**: Works well with major data platforms (BigQuery, Redshift, etc.), ML platforms, and orchestrators (Kubeflow, Airflow).

• **Simple Design**: Focuses on feature serving, keeping the design lightweight.

• **Versioning**: Allows for versioning of features.

• **Good for Startups/SMEs**: Its open-source nature and simple design make it suitable for teams just getting started with feature stores.

POLITECNICO
MILANO 1863

2. Hopsworks Feature Store

- **Type**: Open-source (with enterprise version)
- **Key Characteristics**:
- **End-to-End Integration**: Provides tight integration with Spark, TensorFlow, and Kubernetes, supporting batch and real-time feature engineering.
- **Real-time Capabilities**: Supports low-latency online feature serving via APIs.
- **Data Governance**: Includes advanced features for feature lineage, versioning, and auditing for data governance.
- **Online and Offline Stores**: Supports separation of online and offline feature stores, which are backed by distributed systems like Apache Hive and MySQL.
- **ML Frameworks**: Integration with popular ML frameworks like TensorFlow, PyTorch, and Scikit-learn.
- **Security**: Includes role-based access control (RBAC) and GDPR compliance for sensitive data.

POLITECNICO
MILANO 1863

3. Tecton

- **Type**: Commercial
- **Key Characteristics**:
- **Real-time Feature Serving**: Built specifically for real-time machine learning, offering very low-latency feature serving.
- **Automation**: Automates feature engineering pipelines for both batch and streaming data, minimizing manual feature work.
- **Data Monitoring**: Provides feature quality and drift monitoring to ensure that features stay accurate over time.
- **Integration with Cloud Services**: Natively integrates with cloud environments such as AWS, GCP, and Azure, and is built to work with data warehouses (e.g., Snowflake, Redshift).
- **Feature Management**: Offers version control, feature cataloging, and feature reuse.
- **Collaboration Features**: Supports cross-team collaboration with a feature library to reduce duplicated efforts.

POLITECNICO
MILANO 1863

4. AWS SageMaker Feature Store

• **Type**: Proprietary (part of AWS SageMaker)

• **Key Characteristics**:

• **Tight Integration with AWS Services**: Fully integrated with SageMaker, S3, Redshift, Glue, and other AWS services for seamless data pipeline management.

• **Online and Offline Store**: Supports low-latency online access to features for real-time inference, as well as batch access for offline training jobs.

• **Data Governance**: Provides strong data governance features, including versioning, auditing, and secure data access.

• **Real-time and Batch Processing**: Supports both real-time feature processing for live predictions and batch feature processing for model training.

• **Security**: Leverages AWS's security features like IAM (Identity and Access Management) for controlling access to feature data.

• **Scalable**: Built to handle large-scale, production-ready ML systems using the AWS cloud infrastructure.

POLITECNICO
MILANO 1863

5. Databricks Feature Store

• **Type**: Proprietary (part of Databricks)

• **Key Characteristics**:

• **Unified with Delta Lake**: Built directly into the Databricks environment, leveraging the Delta Lake for version control and data reliability.

• **Feature Lineage**: Tracks the provenance of features to ensure data integrity and enable auditing.

• **Real-time and Batch Support**: Offers both batch and real-time feature processing.

• **Integration with Spark**: Natively integrated with Apache Spark for distributed data processing.

• **Collaboration**: Allows teams to share and reuse features across different models and projects.

• **ML Lifecycle Integration**: Seamlessly integrates with the broader Databricks ecosystem, making it easy to deploy features into model training and inference pipelines.

POLITECNICO
MILANO 1863

6. Google Vertex AI Feature Store

• **Type**: Proprietary (part of Google Cloud)

• **Key Characteristics**:

• **Fully Managed**: A fully managed service as part of Google Cloud's Vertex AI platform.

• **Seamless GCP Integration**: Integrates with other Google Cloud services, including BigQuery, Dataflow, and Vertex AI, for streamlined data pipelines.

• **Online and Offline Store**: Offers both real-time feature serving and batch feature storage.

• **Versioning and Reusability**: Provides version control for feature engineering and allows for easy reuse of features across different models.

• **Scalable and Secure**: Built on Google's infrastructure, offering scalability and enterprise-grade security.

• **Data Monitoring**: Includes tools to monitor feature data for drift and anomalies.

POLITECNICO
MILANO 1863

7. Azure ML Feature Store

- **Type**: Proprietary (part of Microsoft Azure)
- **Key Characteristics**:
- **Azure Integration**: Tight integration with Azure's ecosystem, including Data Lake, Azure Synapse, and Azure ML.
- **Feature Reuse**: Promotes feature reuse across different ML models to reduce duplicated efforts.
- **Real-time and Batch Capabilities**: Provides both online and offline stores for real-time inference and model training.
- **Enterprise-Grade Security**: Utilizes Azure's built-in security features like Azure Active Directory for access control and compliance.
- **Version Control**: Supports feature versioning and lineage tracking to ensure reproducibility and data integrity.

POLITECNICO
MILANO 1863

8. Scribble Data Enrich Feature Store

- **Type**: Proprietary
- **Key Characteristics**:
- **Customizable Pipelines**: Provides high customization for feature engineering and enrichment pipelines.
- **Real-time and Batch Processing**: Supports both real-time feature serving and batch processing for training models.
- **Data Quality and Governance**: Includes extensive tools for ensuring data quality, compliance, and lineage.
- **Cloud-agnostic**: Can be deployed on various cloud platforms like AWS, GCP, and Azure.
- **SME Focus**: Tailored towards small and medium enterprises, providing cost-effective solutions without compromising on features.

POLITECNICO
MILANO 1863

9. Featureform

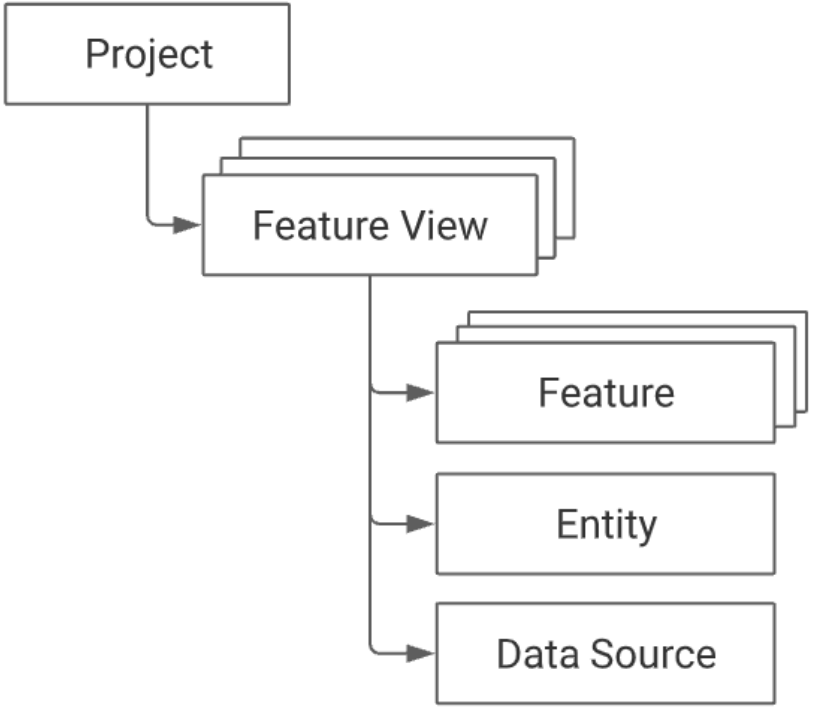
- **Type**: Open-source
- **Key Characteristics**:
- **Declarative Feature Definitions**: Provides a declarative framework to define features, simplifying the process of sharing features across teams and projects.
- **Multi-cloud Support**: Works across different cloud providers and data sources.
- **Real-time Serving**: Supports real-time feature engineering and serving, as well as batch pipelines.
- **Feature Versioning**: Includes version control for features to ensure that models use the correct data.
- **Lightweight Design**: Optimized for smaller teams or those looking for a simpler feature store solution.

POLITECNICO
MILANO 1863

Technology: FEAST

FEAST project structure

POLITECNICO
MILANO 1863



Projects provide complete isolation of feature stores at the infrastructure level. This is accomplished through resource namespacing

For ***offline*** *use cases* that only rely on batch data, Feast does not need to ingest data and can query your existing data

For ***online*** *use cases*, Feast supports **ingesting** features from batch sources to make them available online (through a process called **materialization**), and **pushing** streaming features to make them available both offline / online

POLITECNICO
MILANO 1863

An ***entity*** is a collection of semantically related features. Users define entities to map to the domain of their use case.

A ***feature view*** is a collection of features.

Feature views map to 0 or more entities:

- zero entities (e.g. a global feature like *num_daily_global_transactions*)
- one entity (e.g. a user feature like *user_age* or *last_5_bought_items*)
- multiple entities, aka a composite key (e.g. a user + merchant category feature like *num_user_purchases_in_merchant_category)*

A collection of entities for a feature view is called an **entity key**

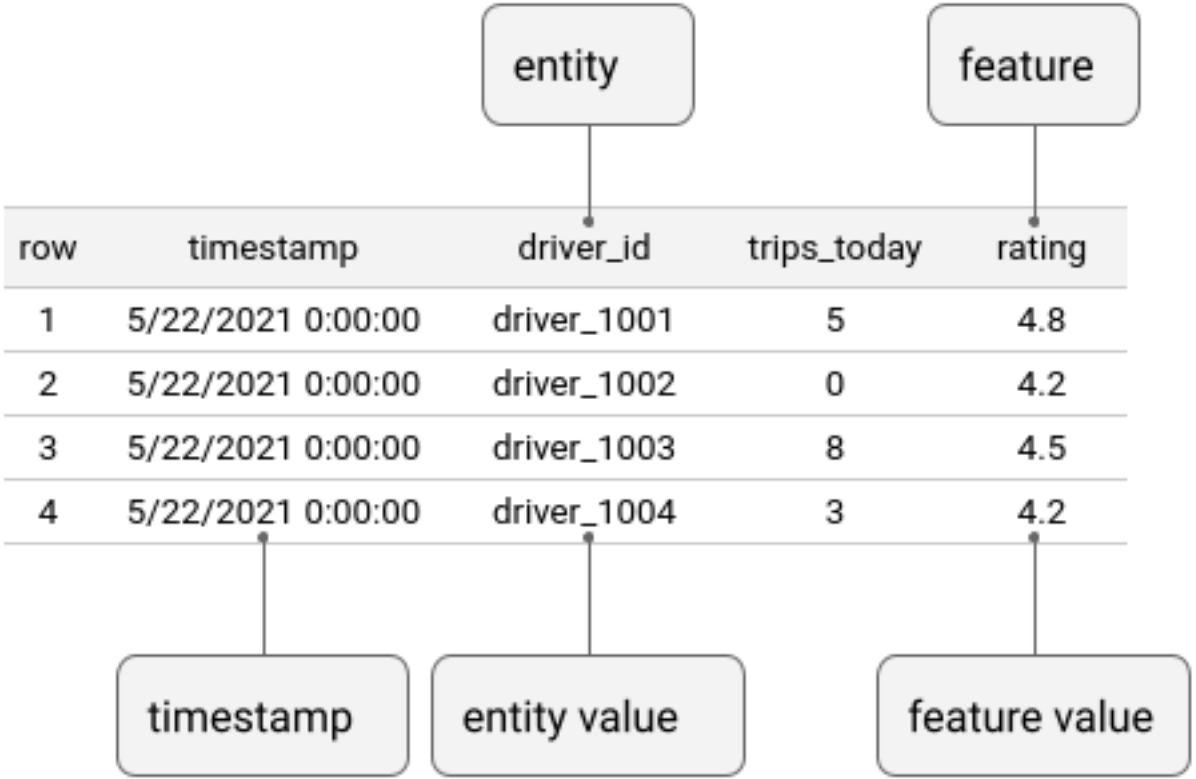
Entities should be reused across feature views

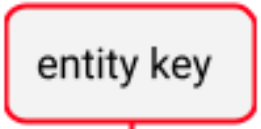
POLITECNICO
MILANO 1863

A ***data source*** in Feast refers to raw underlying data that users own (e.g. in a table in BigQuery). Feast does not manage any of the raw underlying data but instead, is in charge of loading this data and performing different operations on the data to retrieve or serve features.

Feast uses a time-series data model to represent data. This data model is used to interpret feature data in data sources in order to build training datasets or materialize features into an online store.

Feast uses a ***registry*** to store all applied Feast objects (e.g. Feature views, entities, etc). The registry exposes methods to apply, list, retrieve and delete these objects

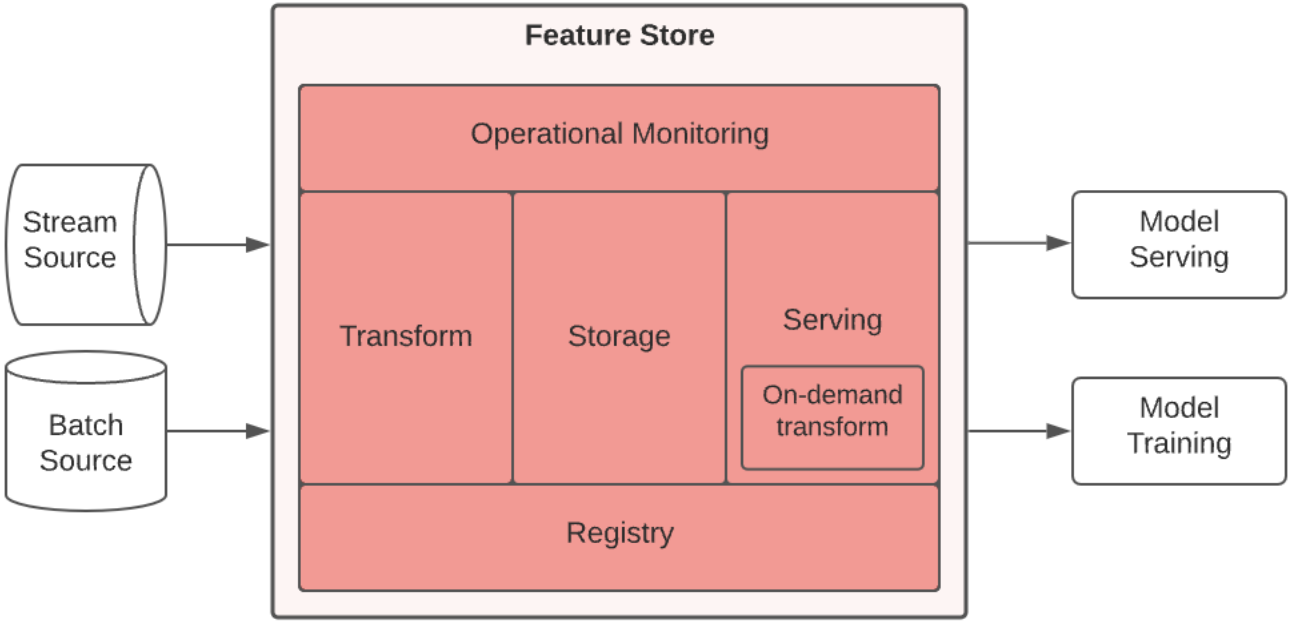

entity
feature
row
timestamp
driver_id
trips_today
rating
1
5/22/2021 0:00:00
driver_1001
5
4.8
2
5/22/2021 0:00:00
driver_1002
0
4.2
3
5/22/2021 0:00:00
driver_1003
8
4.5
4
5/22/2021 0:00:00
driver_1004
3
4.2
timestamp
entity value
feature value



row	timestamp	customer_id	driver_id	trips_today	rating
1	5/22/2021 0:00:00	customer_A	driver_1001	5	4.8
2	5/22/2021 0:00:00	customer_B	driver_1002	0	4.2
3	5/22/2021 0:00:00	customer_C	driver_1003	8	4.5
4	5/22/2021 0:00:00	customer_D	driver_1004	3	4.2

Use case	Example	API
Training data generation	Fetching user and item features for (user, item) pairs when training a production recommendation model	`get_historical_features`
Offline feature retrieval for batch predictions	Predicting user churn for all users on a daily basis	`get_historical_features`
Online feature retrieval for real-time model predictions	Fetching pre-computed features to predict whether a real-time credit card transaction is fraudulent	`get_online_features`

Tasks


Feature Store
Operational Monitoring
Stream Source
Batch Source
Transform
Storage
Serving
On-demand transform
Registry
Model Serving
Model Training

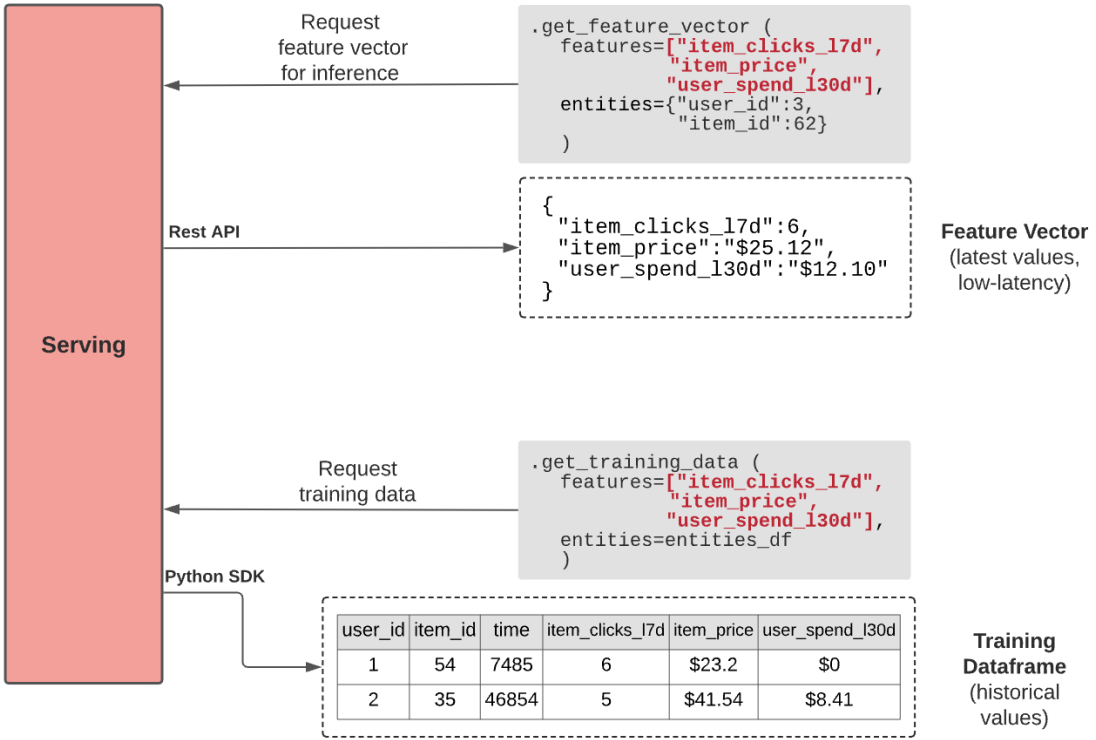
1. Serving

POLITECNICO MILANO 1863

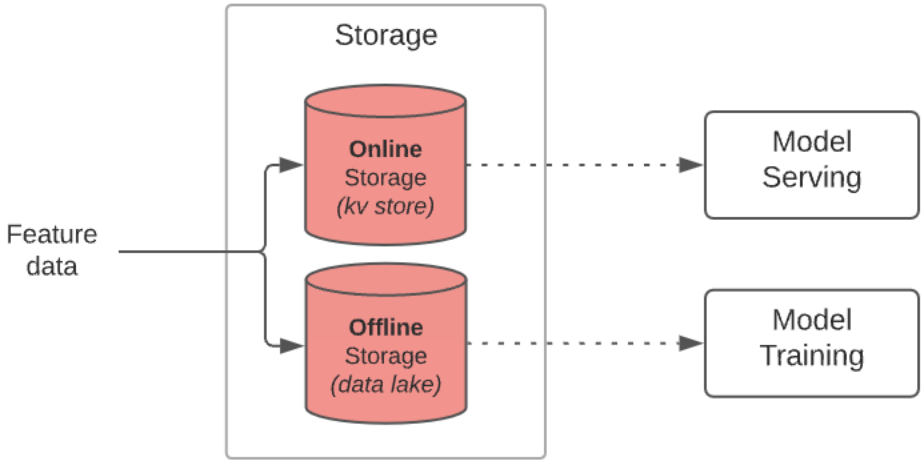
An **easy and canonical way** to access all features in a company consistently across all environments

When retrieving **data offline** (i.e. for training), feature values are commonly accessed through notebook-friendly feature store SDKs.

For **online serving**, a feature store delivers a single vector of features at a time made up of the freshest feature values, through a high-performance API backed by a low-latency database.



2. Storage



Offline storage layers are used to store months' or years' worth of feature data for training purposes.

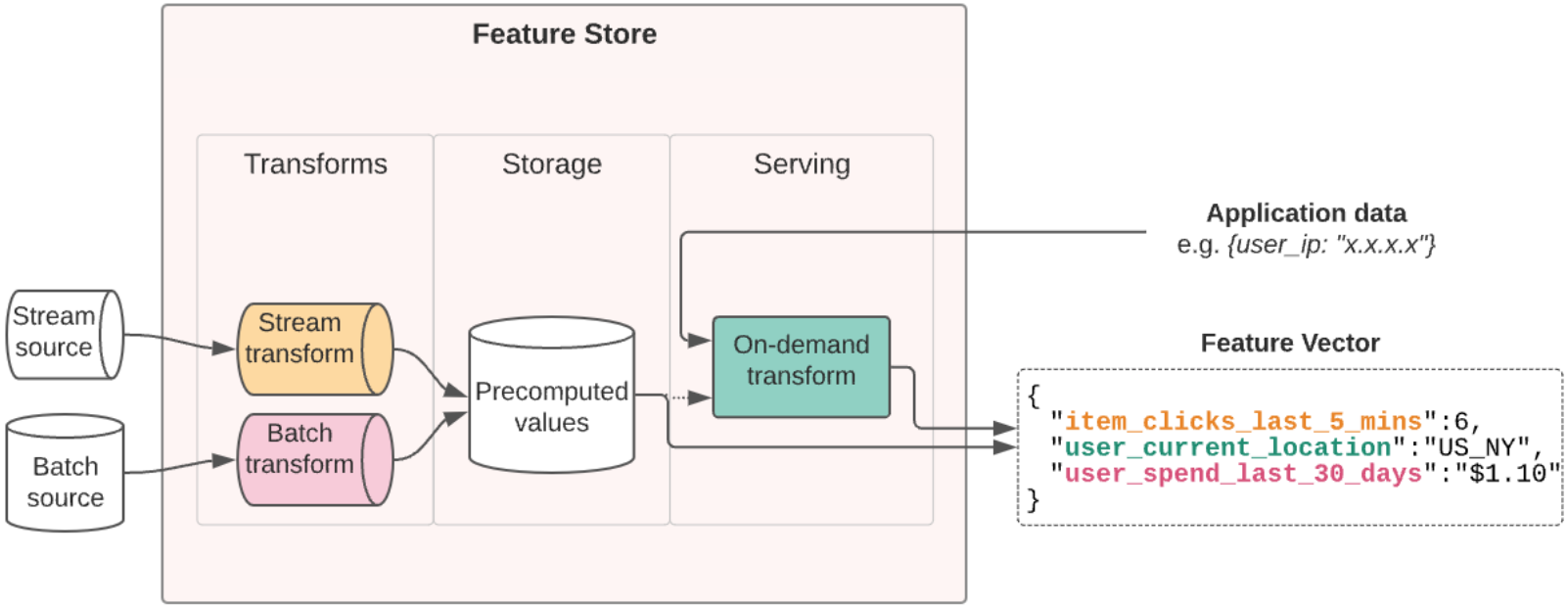
Offline data is often stored in data warehouses or data lakes like S3, BigQuery, Snowflake, Redshift.

Online storage layers are used to persist feature values for low-latency lookup during inference.

They typically only store the latest feature values for each entity.

Online stores are usually **eventually consistent**. They are usually implemented with key-value stores like DynamoDB, Redis, or Cassandra

3. Transform



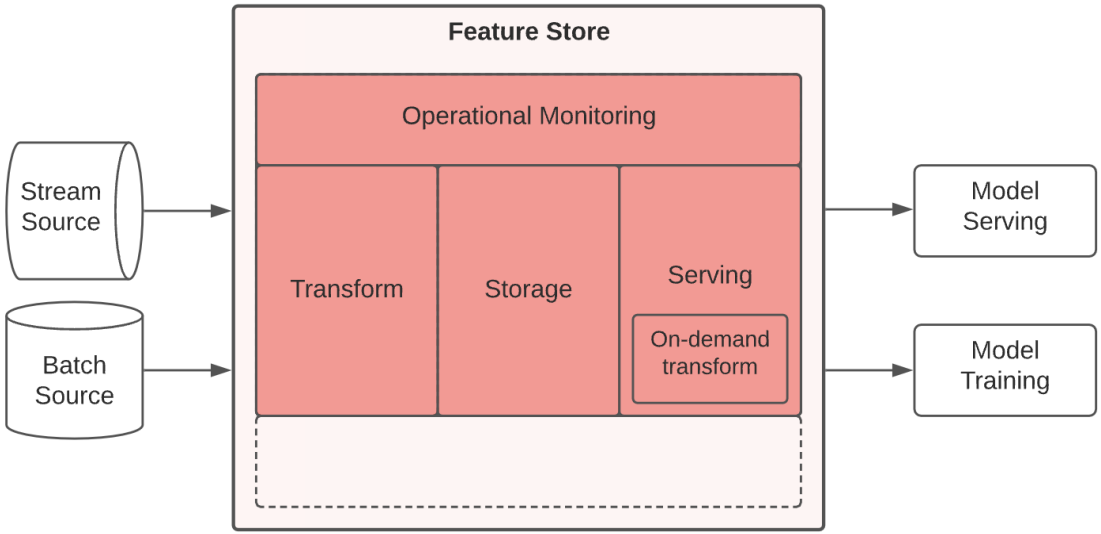
For inference, Transformation jobs are orchestrated to ensure new data is processed and turned into fresh new feature values. These jobs are executed on data processing engines (e.g. Spark or Pandas) to which the feature store is connected.

3. Transform (cont'd)

POLITECNICO MILANO 1863

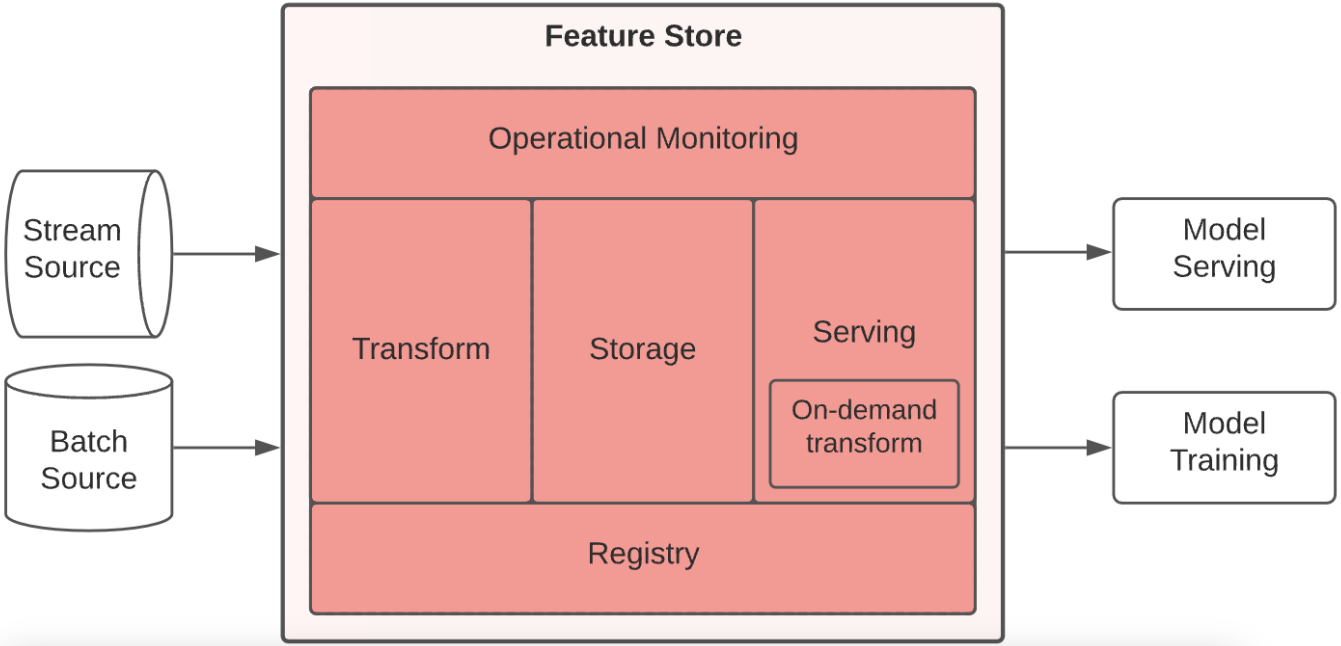
Feature Type	Definition	Common input data source	Common input data source
Batch Transform	Transformations that are applied only to data at rest	Data warehouse, data lake, database	User country, product category
Streaming Transform	Transformations that are applied to streaming sources	Kafka, Kinesis, PubSub	# of clicks per vertical per user in last 30 minutes, # of views per listing in past hour
On-demand transform	Transformations that are used to produce features based on data that is only available at the time of the prediction. These features cannot be pre-computed.	User-facing application	Is the user currently in a supported location? Similarity score between listing and search query

4. Monitoring



When running production systems, it's also important to monitor operational metrics. Feature stores track operational metrics relating to core functionality. E.g. metrics relating to feature storage (availability, capacity, utilization, staleness) or feature serving (throughput, latency, error rates).

5. Registry



The registry is a central interface for user interactions with the feature store. Teams use the registry as a common catalog to explore, develop, collaborate on, and publish new definitions. The registry allows for important metadata to be attached to feature definitions

POLITECNICO
MILANO 1863

Concluding

Conclusion

POLITECNICO
MILANO 1863

In summary, a feature store acts as the backbone of the feature lifecycle in machine learning systems, enabling teams to efficiently manage and serve high-quality features across different stages of the ML pipeline, from model development to deployment.

Ack

FEAST

Gavita Regunath

Marco Brambilla.

POLITECNICO
MILANO 1863

SYSTEMS AND METHODS FOR BIG AND UNSTRUCTURED DATA

Feature Store

Marco Brambilla

marco.brambilla@polimi.it

@marcobrambi