**What is a Feature Store in ML, and do I need one for my model?**

[[Qwak](https://medium.com/@mlengineering?source=post_page-----ba2682b628b0--------------------------------)](https://medium.com/@mlengineering?source=post_page-----ba2682b628b0--------------------------------)

[[Infer](https://medium.com/infer-qwak?source=post_page-----ba2682b628b0--------------------------------)](https://medium.com/infer-qwak?source=post_page-----ba2682b628b0--------------------------------)

[Qwak](https://medium.com/@mlengineering?source=post_page-----ba2682b628b0--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F5ccc0b76c62e&operation=register&redirect=https%3A%2F%2Fmedium.com%2Finfer-qwak%2Fwhat-is-a-feature-store-in-ml-and-do-i-need-one-for-my-model-ba2682b628b0&user=Qwak&userId=5ccc0b76c62e&source=post_page-5ccc0b76c62e----ba2682b628b0---------------------post_header-----------)

Published in

[Infer](https://medium.com/infer-qwak?source=post_page-----ba2682b628b0--------------------------------)

·

15 min read

·

Dec 5

185

5

Learn what a Feature Store in Machine Learning is, its significance, how it streamlines the MLOps workflow, and when it is best to use.



By Grig Duta, Solution Engineer at [Qwak](https://www.qwak.com/" \t "_blank)

In essence, a feature store is a dedicated repository where features are methodically stored and arranged, primarily for training models by data scientists and facilitating predictions in applications equipped with trained models. It stands as a pivotal gathering point, where one can formulate or modify collections of features drawn from a variety of data sources. Moreover, it empowers the creation and enhancement of fresh datasets from these groups of features, catering to models in training phases or applications that prefer to simply fetch precomputed features for their predictive analyses.

**What is a Feature Store — your ML Kitchen Pantry**

What better way to demystify complex machine learning concepts than by comparing them to something as universal and familiar as food? Just as the right ingredients and their preparation are crucial in cooking, similar principles apply in the world of machine learning, especially when it comes to managing data. To bring this idea closer to home, let’s use a kitchen analogy to explain one of the key components in ML workflows: the Feature Store.

Imagine you’re working on a machine learning project, and you’ve got all these ingredients (data) that you need to prepare (process) before you can cook up a great model. In this scenario, a Feature Store is like your go-to kitchen pantry where you keep all your prepped ingredients (ML features) ready to use.

So, what are these ML features? They’re bits of data you’ve carefully selected and transformed, ready to be fed into your machine learning models. Think of them as the chopped veggies, marinated meats, or spices that you’ve prepared beforehand. You want these prepped ingredients to be consistent, high-quality, and easily accessible whenever you’re cooking up different dishes (models).

A Feature Store keeps all these prepped ingredients in one place. It ensures that whenever you or your teammates are cooking (building models), everyone uses the same quality of ingredients. This way, you don’t end up with one person using stale garlic while another uses fresh, leading to different tastes (model performances).

In summary, a Feature Store in ML is like your kitchen pantry for data. It helps you keep your prepared ingredients (features) fresh, consistent, and ready for use, making your cooking (model building) process a whole lot smoother and more reliable.

**The Purpose of Feature Stores**

Feature Stores have been around for some time, originating with concepts like [Uber’s Michelangelo](https://www.uber.com/en-RO/blog/michelangelo-machine-learning-platform/) and Airbnb’s [Zipline](https://conferences.oreilly.com/strata/strata-ny-2018/public/schedule/detail/68114.html), and they come in different flavors. Some are open source and focus on an idiomatic way to describe features, while others are more “tabular” and oriented towards fixed schemas for the data they’re populated with.

In this section, we’ll explore the primary purposes Feature Stores serve without a specific focus on how they’re built. We’ll delve into the technical aspects in later sections.

Feature Stores are crucial when working with ML in production for three main reasons:

* **Features Reusability:** Often, teams work in silos, developing models that use the same features. However, these features are defined differently and computed separately, leading to duplicate efforts and higher compute and storage costs. Feature Stores act as a hub for standardizing ML features. With Feature Stores, one ML team can use another team’s features. Since these features are cataloged and shared within the organization and pre-computed before consumption, the organization saves on cloud bills by avoiding recomputing the same features.
* **Standardized Feature Definitions:** Building on the previous point, when a team defines a feature set for their model, they might not always consider documenting how those features are extracted and computed. With a Feature Store, the data source and feature transformations follow a consistent and easily understandable pattern, fostering the reuse of work across teams.
* **Consistency between Training and Serving:** Feature engineering is not only relevant in data scientist’s notebooks; a significant portion of ML features is used in real-time models. Ensuring that the data is handled the same way between training and serving is crucial for consistent performance in real-time predictions. This consistency is known as [training-serving skew](https://www.qwak.com/post/training-serving-skew-in-machine-learning).

Beyond these major reasons, it’s worth noting that Data Scientists often don’t have direct access to data pipelines, which falls more on the Data Engineering side. With a Feature Store, the feature engineering logic from a notebook can also power the training and inference feature sets in production, enabling Data Scientists to operationalize their models themselves.

This doesn’t imply that every DS team will have access to all data sources; in fact, many Feature Store solutions include Governance and Compliance capabilities, allowing specific teams access to specific data and features while still providing easy exploration.

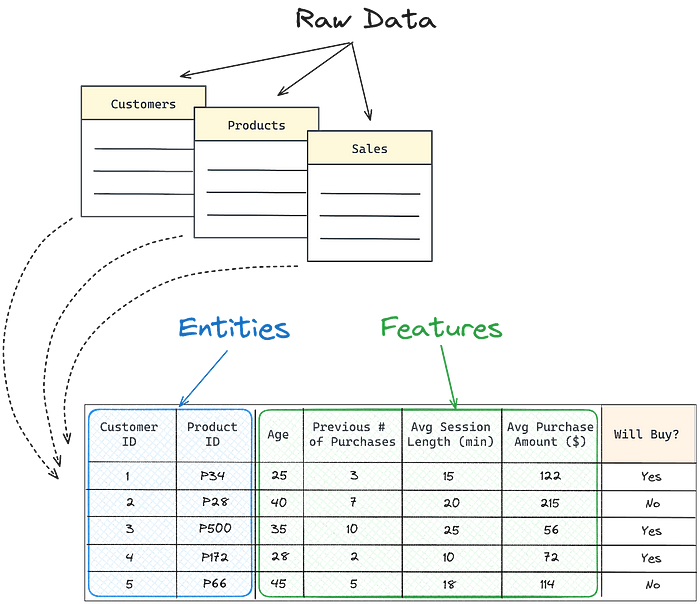
Finally, Feature Stores are feature storage systems that allow you to look up feature vectors based on their associated entities with very low latency. This capability means that models, which typically run inference in batches due to requiring pre-computed features, can now run in real-time and query the Feature Store whenever input for a prediction is required.

**From Raw Data to ML Features**

Features in machine learning are akin to the building blocks that models use to understand and predict the world around us. Imagine raw data as the rough material — it’s abundant and varied, often messy and unstructured. This data could be anything: rows of figures in a database, text from customer feedback, or even pixels in a photograph.

Turning this raw data into features is a transformative process. It’s like refining ore into gold; the data is cleaned, shaped, and structured into a form that machine learning models can not only comprehend but also learn from. This transformation, known as feature engineering, is an art in itself, requiring both skill and creativity.

Once crafted, these features are stored — often in a feature store — ready to be called upon by machine learning models. In this organized form, they become the vital ingredients for training models and making accurate predictions. Features are the bridge between the chaotic world of raw data and the orderly domain of predictive analytics, enabling AI systems to make sense of and interact with the world.



**The Role of Feature Stores in Data Science Workflows**

Feature stores have become a cornerstone in the data science landscape, revolutionizing how machine learning models are trained and operationalized. Their introduction has bridged a critical gap in the machine learning workflow, particularly in large-scale enterprise environments where operationalizing machine learning remains a significant challenge​​.

**Centralized Management and Accessibility of Features**

At the heart of a feature store’s utility is its ability to manage and provide access to both historical and live feature data. This management extends to supporting the creation of point-in-time correct datasets from historical feature data. Essentially, feature stores serve a dual role: they function as a data warehouse for machine learning features, enabling the discovery, monitoring, analysis, and reuse of features within an organization, and as an operational store for precomputed features, which online models can use to augment their feature vectors with historical and contextual data​​.

**The Feature Store: A Hub for Collaboration and Consistency**

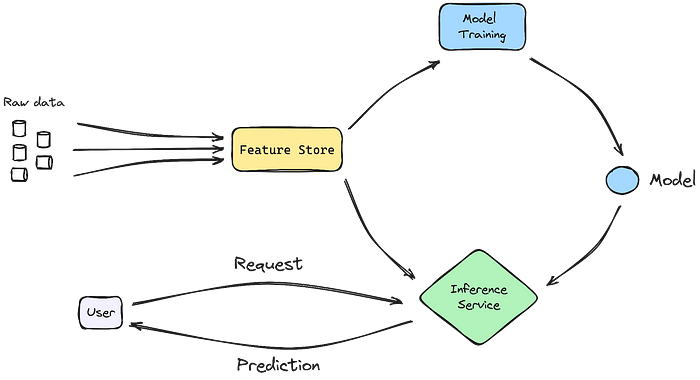
The feature store is not just a storage space but a dynamic environment where data scientists can create or update groups of features derived from multiple data sources. These feature groups are then used either to train models or to make predictions by applications that leverage trained models. A critical aspect of feature stores is their ability to facilitate the discovery, documentation, and reuse of features, ensuring their correctness for both batch and online applications. This capability is particularly beneficial in maintaining consistent feature computations across batch and serving APIs​​.

**Enhancing Real-time Applications**

A practical example of a feature store’s impact is evident in e-commerce recommendation systems. Here, user queries, typically poor in information content, can be enriched with precomputed features from the feature store. This enrichment process transforms an information-poor signal into a rich one by incorporating features representing the user’s history and context, such as past interactions and current popular items. Consequently, applications can become AI-enabled, providing personalized and relevant user experiences​​.

Feature stores act as a critical data layer connecting **feature pipelines**, **training pipelines**, and **inference pipelines**. They are essentially **dual-database systems**, containing a **columnar data store** for historical (offline) feature data and an online(low-latency)**row-oriented data store** for serving precomputed features to online applications. This architecture allows feature stores to integrate seamlessly into existing enterprise data infrastructure and machine learning tooling, offering various solutions to meet diverse organizational needs.

In summary, feature stores have significantly redefined the approach to **feature engineering and management** in data science. They **streamline workflows**, **enhance collaboration**, ensure **consistency in feature usage**, and support the **adaptability of ML applications** to real-world scenarios. This evolution marks a significant leap forward in the operationalization of machine learning at scale.



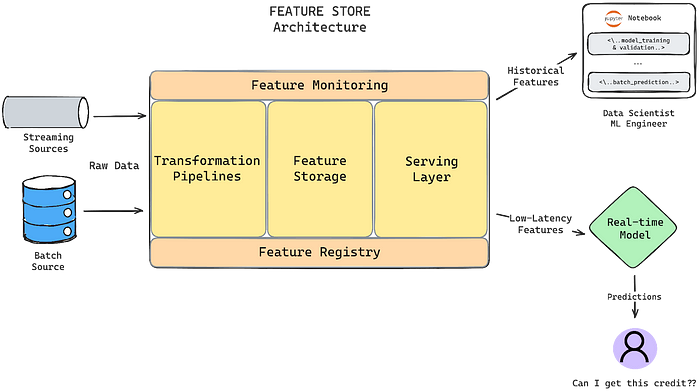
**Offline Feature Stores** are the backbone for training and batch predictions. They are repositories of historical feature data, crucial for training machine learning models with comprehensive, time-sequenced information. This historical perspective is vital for models that require extensive analysis over time to understand trends and patterns.

**Online Feature Stores**, in contrast, are engineered for speed and responsiveness. They cater to real-time, low-latency prediction needs of models in production. This is particularly critical in scenarios where decisions need to be made instantaneously based on the most current data, such as in fraud detection systems or real-time personalization in digital platforms.

The dual nature of feature stores — encompassing both offline and online functionalities — allows them to offer a comprehensive solution for machine learning models. They ensure that models are not only trained on rich, historical data but also capable of reacting swiftly and accurately in real-time scenarios​​.

This bifurcation in feature store architecture underscores its versatility and essential role in modern machine learning infrastructures, bridging the gap between data preparation and real-world application.

**Components of a Feature Store**



**Feature Engineering (Transformations)**

When it comes to Feature Stores, feature transformations trace the path from data ingestion to the desired shape and form for ML training or inference. There is no one-size-fits-all solution for these transformations, primarily due to the diverse sources your data might originate from and the creative engineering applied to transition data into ML models.

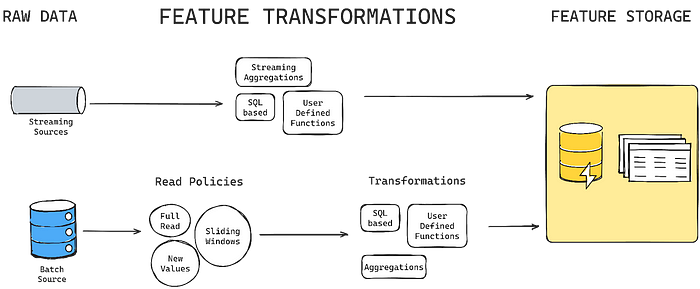
As a general guideline, feature transformations should facilitate the **automation** and **standardization** of data pipelines while offering the flexibility to select, filter, aggregate, and manipulate raw data into reusable features for ML models.

Various types of transformations exist, with **SQL-based transformations** being common, especially in schema-based feature stores. Alternatively, you can leverage **user-defined functions**written in **Python**. This allows you to define aggregations and more complex computations, such as statistical functions.

Different data sources present unique challenges. Streaming sources involve continuous data ingestion and processing, while batch sources deal with more static data, ingested regularly or on-demand. Some sources maintain a consistent schema, like RDBMs, while others store unstructured data, like NoSQL databases. Data formats range from tabular and queryable to simple CSV files on an S3 bucket, or something in between, like Parquet files behind an AWS Athena query engine. This diversity underscores the importance of flexibility when building or adopting a Feature Store solution.

Given that many feature store architectures focus on storing time-series data, an integral aspect of feature ingestion relates to timeframes or **ingestion windows**, particularly when dealing with batch data sources. Specifying ingestion windows in feature sets allows you to define which data is relevant for regular processing in ingestion jobs. For example, you might have new data arriving every minute, and an ingestion job could run every hour, considering only the new data from the past hour.

The complexity of your data systems and the ingenuity of your feature engineering are crucial factors when selecting a**Feature Store design** to meet the specific needs of your organization.



**Feature Storage**

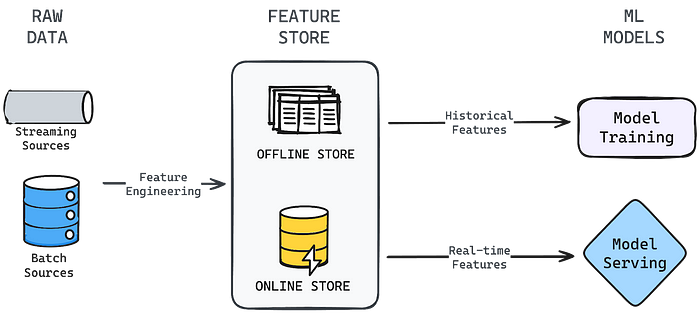
Tightly coupled with Feature Transformations and serving as the natural progression after processing data, the feature storage layer can be envisioned as a dual-database system. On one side, it stores historical data with a focus on columnar retrieval. On the other, it features a row-oriented retrieval system emphasizing quick, low-latency data lookup.

The feature storage exclusively contains pre-computed values resulting from feature ingestion and transformations, guided by the logic and definitions stored in the feature registry. This storage layer comes in two types: **Offline Store** and **Online Store**.

The **offline store** is inherently more static. As data is often time-based, it’s appended to rather than rewritten. This means the offline store theoretically contains all processed data, forming part of the feature lineage. In terms of architecture, the offline store must be cost-efficient for storing large amounts of data. It also needs to be effectively partitioned or sharded for easy accessibility.

In contrast, the **online store** stores only the latest feature vectors for a specific feature set entity. For instance, a user on an online platform, identified by an ID, will have in the online store only their last visit. When looking up this user for a model predicting whether 2FA is necessary, only the last visit is relevant and retrieved.

Access to the **Feature Storage** is typically carried out via an API or SDK, through the Serving layer, diverging from the direct connection approach common in typical data stores.



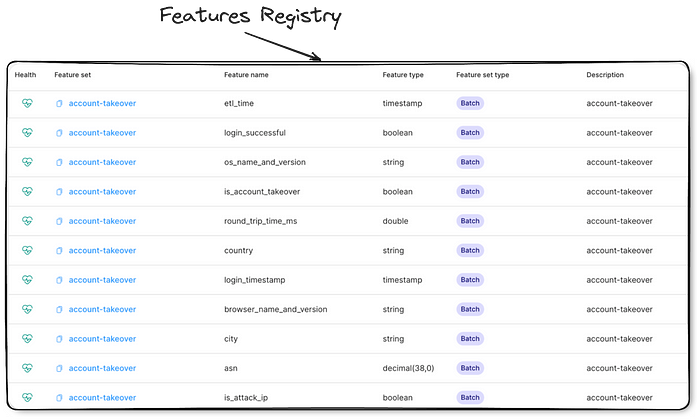
**Feature Registry**

This component acts as a **centralized repository** for all features within the feature store. The **feature registry** stores comprehensive metadata, ensuring the consistency of feature definitions. This includes details about the raw data’s origin (ingestion), data transformations, and the format, as well as how features are utilized by models.

Another crucial role of the feature registry is managing **access control**. It stores information about who has access to a set of features and the type of access granted.

Beyond its metadata storage function, the **feature registry** is a pervasive yet somewhat unseen component of the feature store. It serves as the glue that binds together all the other pieces constituting the feature lineage.

In some feature store solutions, the feature registry is the backbone of the system. In others, it might serve as a means to store metadata connecting data sources to transformations, ingestion logic, and feature formats. However, the specific role can vary, and the presence of some form of a registry is a common feature across most feature store solutions.

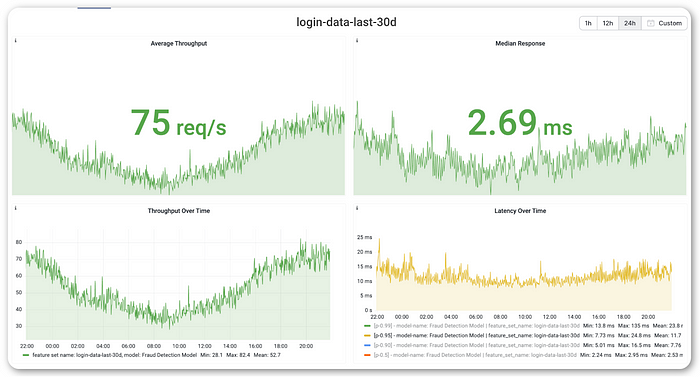


**Feature Serving**

**Feature serving** enables Data Scientists and ML engineers to interact with the Feature Storage, retrieving historical features for training or fetching the latest feature vector for a specific entity. In some feature stores, there’s the added capability of on-the-fly features — computed on-demand at request time rather than being pre-computed and stored.

In contrast to a data-store connection client that allows complex querying and management operations, the **feature serving client** relies on an API or SDK and is designed with a more specific purpose. Typically, its role is to fetch selected features between two dates or for one or more entity IDs.

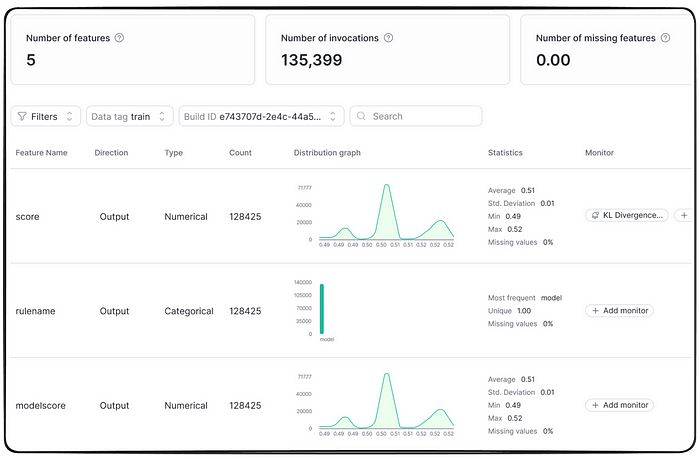
While certain feature stores may specialize in particular types of operations and possess varying capabilities in their serving mechanisms, the fundamental concepts mentioned above remain relatively consistent across the most common types of feature stores.



**Feature Monitoring**

In any productionized ML system, **monitoring** is crucial to ensure ongoing quality. For a feature store, monitoring helps detect changes in data quality over time, identify concept drift, assess training and serving data skew, and, in the case of real-time features, ensure serving meets desired latency thresholds. Let’s break down these aspects:

* **Data Quality:** This involves ensuring data anomalies stay within defined error limits, such as null percentages, date formats, or unexpected values. Feature store solutions typically provide alerts or UI dashboards to highlight such events and patterns.
* **Data Drift or Concept Drift:** This monitors the statistical distribution of data over time, notifying you of significant changes compared to the last defined state. Statistical measurements like [KL divergence](https://rmoklesur.medium.com/understanding-kl-divergence-a-fundamental-measure-in-machine-learning-2ebdf384851#:~:text=KL%20divergence%20is%20a%20powerful,language%20processing%2C%20and%20reinforcement%20learning.), [PSI](https://www.machinelearningplus.com/deployment/population-stability-index-psi/), and others help observe patterns in data distribution.
* **Serving Performance:** The serving layer, often a backend application, requires monitoring for metrics like throughput, serving latency, and requests per second.
* **Training-Serving Skew:** Checks for training-serving skew in ML pipelines can be implemented, as discussed in[our other article](https://www.qwak.com/post/training-serving-skew-in-machine-learning). These checks help ensure consistency between the conditions during model training and real-time serving.



**A Features Store Example**

In a detailed ML model focusing on **e-commerce personalization**, the introduction of a feature store significantly enhances the model’s capability. Imagine a user searching for products on an e-commerce platform. The search query, typically limited to text and user/session IDs, offers limited information. However, the feature store comes into play by leveraging these identifiers to access an extensive array of **precomputed features**.

This is where the feature store truly shines, transforming the basic query into a rich dataset. It pulls in detailed user history, like previous purchases and viewed items, and combines it with real-time data, such as trending products or seasonal preferences. This enrichment process not only adds depth to the user’s search but also enables the ML model to deliver highly **tailored product recommendations**. The feature store acts as a bridge, turning simple user queries into detailed, actionable insights for personalized user experiences.

**When You Need a Feature Store (And When You Don’t)**

When navigating the decision to implement a feature store in your machine learning workflow, it’s essential to consider its benefits in managing a large number of features across various models, particularly in **complex, collaborative environments.** The store’s value is notably high when the **same features are used across multiple models**, offering a centralized, consistent repository that reduces redundancy and engineering efforts.. A feature store is an invaluable asset in scenarios where managing a plethora of features across various models is critical. Its importance is magnified in complex, collaborative environments requiring consistent, high-quality features accessible to multiple teams.

For**real-time applications** such as fraud detection or dynamic pricing, the ability of a feature store to rapidly process and serve up-to-date features is crucial. It ensures that live decision-making models receive the most relevant, current data, directly impacting their effectiveness.

On the flip side, in smaller-scale projects or early-stage machine learning endeavors where complexity and feature reusability are minimal, the investment in a feature store might not be justified. Similarly, if your ML applications primarily rely on batch processing without immediate needs for real-time feature updates, the real-time capabilities of a feature store may not be essential.

Ultimately, the decision to adopt a feature store should be balanced against the sophistication and scale of your ML operations, the nature of your projects (**real-time vs. batch processing**), and the resources available for implementation and maintenance.

Balancing these factors — feature reusability, real-time needs, and the scale of ML operations — against the resources needed for a feature store’s implementation and upkeep is key. This ensures that its adoption aligns with your project’s specific requirements and overall machine learning strategy.

**Getting started with Features Stores**

Getting started with feature stores involves a strategic decision: **build your own, buy a managed solution, or use an open-source option**. Building from scratch allows for complete customization but requires significant resources and expertise. Managed solutions like **[Qwak](http://qwak.com/" \t "_blank), AWS SageMaker, or Google Cloud Vertex** offer ready-to-use, scalable platforms with less overhead. Open-source alternatives like **Feast or Hopsworks** provide flexibility and community support. Each option has its trade-offs in terms of cost, control, and ease of integration, making it crucial to assess your team’s capabilities, project needs, and long-term goals when choosing your feature store path.

**Conclusion**

In conclusion, our exploration of “What is a Feature Store” in machine learning reveals its indispensable role in modern data science workflows. Feature stores are not just advantageous but essential for organizations aiming to scale their ML efforts efficiently. They address real-world challenges by eliminating data silos, reducing the time-to-market for ML models, ensuring data quality, and maintaining consistency across different ML applications.

Qwak’s Feature Store is designed to be your one-stop-shop for feature management, offering out-of-the-box solutions for data ingestion, transformation, storage, and monitoring. It’s not just a tool but a comprehensive platform that fits seamlessly into your existing MLOps pipeline, enhancing both productivity and performance.

As machine learning continues to mature, feature stores will become as ubiquitous as data lakes and warehouses. If you’re looking to operationalize your ML models efficiently, this might be the time to invest in a feature store.