Feature engineering

Article · October 2019

CITATIONS
0

READS
787

1 author:

Amit Agarwal
Cognizant Technology Solutions
1 PUBLICATION **0** CITATIONS

SEE PROFILE

Digital Business

Accelerating Machine Learning as a Service with Automated Feature Engineering

Building scalable machine learning as a service, or MLaaS, is critical to enterprise success. Key to translate machine learning project success into program success is to solve the evolving convoluted data engineering challenge, using local and global data. Enabling sharing of data features across a multitude of models within and across various line of business is pivotal to program success.

Executive Summary

The success of machine-learning[1] (ML) algorithms in a broad range of areas has led to ever-increasing demand for its wider and complex application, proliferation of new automated ML platforms/solutions and increasingly flexible use of these techniques by nonexperts. Most enterprises began their ML journey with projects of simpler analytical complexity because they were primarily focused on the maturity of their data infrastructure, ML model development process and deployment ecosystem.

October 2019

Cognizant

According to a recent O'Reilly published study[2,3,4] roughly 50% of enterprise respondents said they were in the early stages of exploring ML, whereas the rest had moderate or extensive experience of deploying ML models into production.

Enterprises, irrespective of their maturity, are currently focused on managing data pipelines and evaluating/developing ML platforms. But as they ascend the maturity curve, they need to solve the problem of the ML model-related data pipeline labyrinth as creation and management of these elements are labor-intensive, which over time introduces data complexities and related operational risks.

ML is core to the success of digitally native businesses such as Uber and LinkedIn for creating new products and redefining customer experience standards at a global scale. There are certain aspects of ML architecture that can be deftly adopted by digital immigrant enterprises as they seek to mature their use of artificial intelligence (AI).

Creating a feature store, a central repository of features (basically any input into an ML model) in a store with a marketplace construct, enables producers like ML engineers (creating and populating new features) to share them with consumers like data scientists (building ML models). This will reduce GTM substantially, along with enabling data lineage and bringing governance into the data pipeline labyrinth. For enterprises to mature in ML, a focus on setting up a feature store will be as essential as the adoption of auto ML frameworks, model monitoring and model visualization — which was also the outcome noted by the recent O'Reilly survey.

This white paper offers insights into why enterprises need a fully functional feature store in their ML maturity journey and how this can be achieved using an operating model that can accelerate ML scale goals through automation, making ML learning algorithm features reusable, cost-effective and tangible. This is critical because our approach automates one of the most laborious activities in the model lifecycle — feature engineering.

The need for a centralized feature engineering ecosystem

ML[5] is a powerful toolkit that enables businesses to strive for excellence, whether it's new product development or achieving operational efficiencies. However, ML initiatives entail the development of complex systems that behave differently than traditional IT systems.

In fact, ML systems contain inherent risks (e.g., complex data pipelines, unexplainable code) which, unless addressed properly, lead to high maintenance costs over the long run. The development of ML code is generally seen as labor-intensive and complex, whereas other essential activities surrounding it are seen as less critical — which is incorrect. Rather, data (functions such as quality, features, etc.) and resource management are equally important for building a successful ML infrastructure (see Figure 1).

The process of building and deploying an ML model goes beyond setting up a requisite infrastructure. ML projects have a typical timeline of two to four months for idea validation and prototype development, which often gets extended by several more months if prototypes are pushed into production. The cycle is repeated for each model rebuild iteration or new model development.

Figure 2 (page 4) illustrates an ML project, depicting various stages and related efforts. Processes with relatively less effort have been addressed by the deployment of ML platforms like Sagemaker, but key labor-intensive processes around data acquisition and processing are still repeated in each iteration of the model development exercise.

A day in a life of a data scientist (DS) consists of deriving insights, knowledge and model

ML heat map depicting processes and related efforts[6]

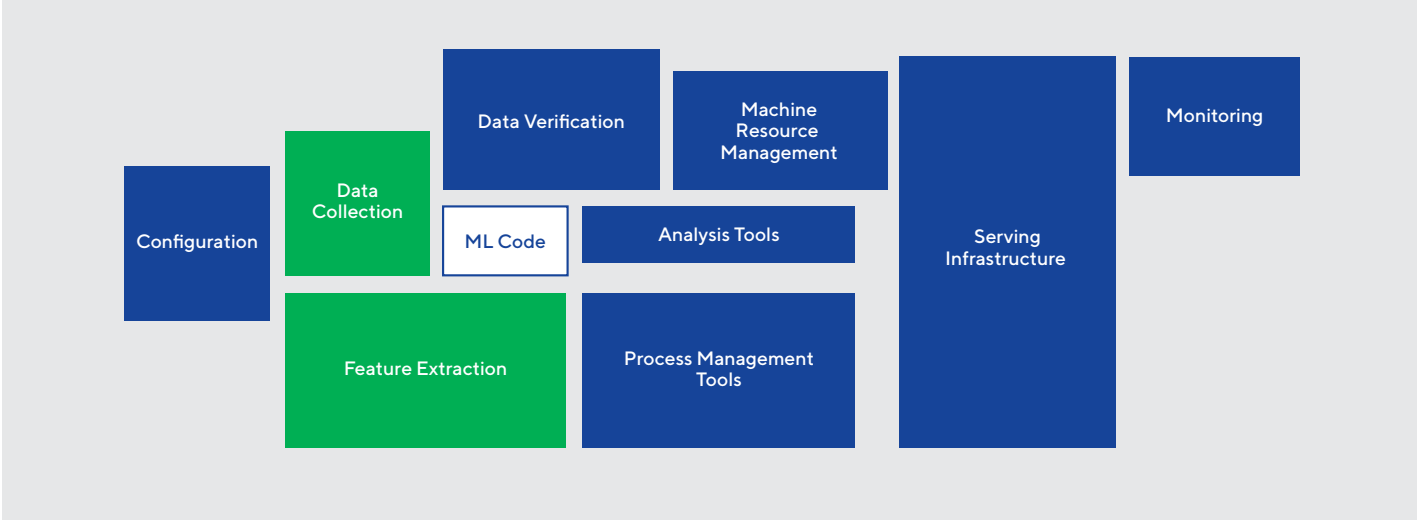


Figure 1

Illustrative model lifecycle

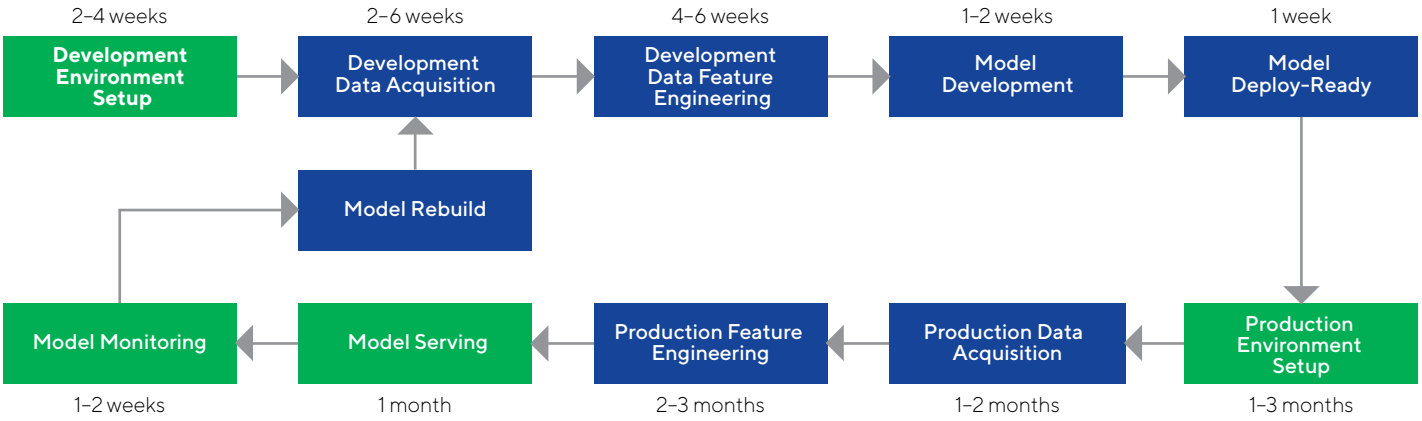


Figure 2

development from data. (For more on this, read "Learning from the Day in the Life of a Data Scientist" in our Digitally Cognizant blog). This requires data cleansing, transformation and feature extraction before building a stitch of ML code. The process starts with data extraction in a modeling sandbox, on to hypothesis validation, followed by deployment of code that requires designing a fully fledged data pipeline. The activities happen primarily in isolation, which is typical of an experimentation phase.

Upon successful exploration, other key role players — like ML engineers and an ML architect — must come up to speed and plan necessary support activities, which results in a longer development lifecycle (see Figure 3).

During model development, the data scientist will build common features and features that are specific to the model. Industry standard practice is to create extract, transform, load (ETL) pipelines for common features while generally bundling model-specific features within the model itself — which leads to the following situations:

Working solo

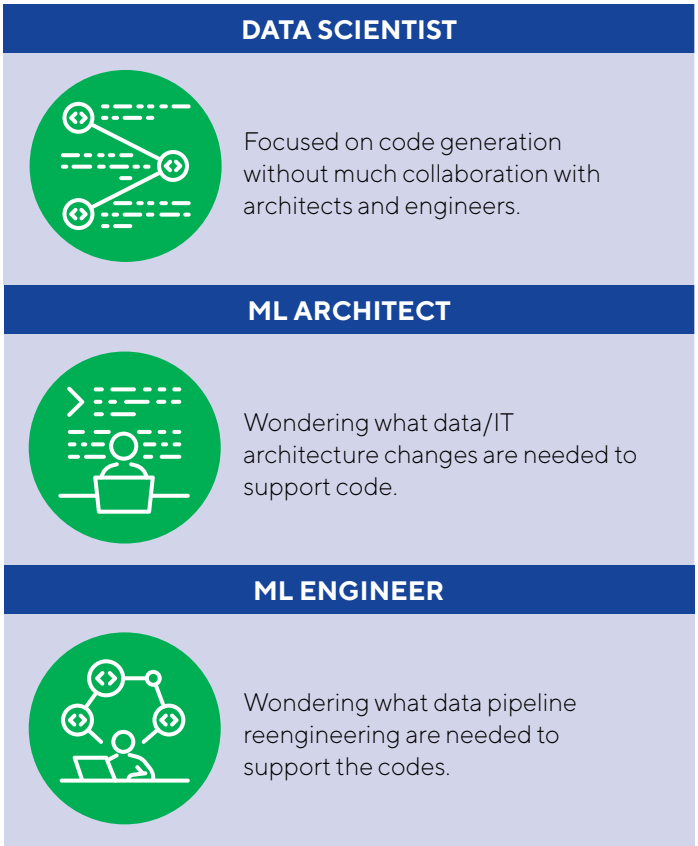


Figure 3

- Each model ends up with a customized data pipeline, which become complex to manage as they proliferate.
- The same features are created again and again with each iteration of model development or model rebuild.
- It is difficult to track data-feature/model lineage since features are spread across data stores, which makes it hard to access impact on the model due to data drifts.

Of late, the exploration and adoption of automated ML frameworks (like our Learning Evolutionary AI Framework, or LEAF) is gaining traction with data scientists as they endeavor to address automated feature engineering (limited scope), model selection and hyper-parameter tuning for the rapid development of models. Their inherent automated feature engineering is limited to a certain model, i.e., it is technology with limited reusability.

One common mitigation plan to reduce all data-dependency-related risk in a ML system is by adopting a centralized ecosystem, wherein all features are captured and catalogued for use.

The anatomy of a 'feature' and 'feature store'

A feature is basically any input into an ML model. It is a set of variables that are incorporated into an ML model with the intention to improve model performance and accuracy. Features are derived values extracted from files and tables (a database) — and more importantly, computed from one or more tables. These are usually grouped together to minimize operational overhead and optimize storage. A feature, for example, can be any column with a calculated, flagged or one hot encoded value.

Here are some sample questions that can be turned into features by following relationships and aggregations:

- "How often does this customer make a purchase?"
- "How long has it been since this customer's last login?"
- "How much does the energy usage vary for this customer?"
- "Does this customer typically buy luxurious or economical holiday packages?"

A feature store is a central repository for storing documented, curated, and access-controlled features. It is a central place to store features that are properties of data, be it in the form of statistical derivations, piece of text, image pixel coordinates, aggregated value of purchase history, etc. This enables feature management to be uniform, reliable, reusable and governed.

A feature store shouldn't be perceived as a type of new data store. In fact it should be recognized as a store of feature recipes with occasional time dependencies. For example, a feature like "number of login attempts in the last hour" is used in fraud

A feature store shouldn't be perceived as a type of new data store. In fact it should be recognized as a store of feature recipes with occasional time dependencies.

models, customer service models and retention models. Each model will be computing it at a different historical point — the fraud model perhaps during a logon attempt, the customer service call at the point someone calls a call center and the retention model a certain date for the model to be built. But when the feature goes into production, it needs to run every single time a customer calls the call center (for example).

A feature store enables reusability of features across the enterprise, as existing features are visible to all potential users (e.g., business analysts, business intelligence developers, data scientists, etc.) across the business domain. The feature store supports feature enrichment, ranking, discovery, lineage (both data to feature and feature to model) and lifecycle management.

Both development and model serving teams need a diverse feature set, which can be met easily through the store. This will enable both teams to discover, store and manage features, while also decommissioning features that are no longer needed.

Figure 4 highlights some examples of features that are commonly used across different lines of business and which get recreated in every instance of modeling. If harnessed properly, they can be reused across different model sets, thus bringing greater operational synergy and accelerating time-to-market.

Business use cases where similar features are created and used in different models

Business Use Case	Common Features	Use-Case-Specific Features
Credit Risk Modeling	**Historical transaction data:** • Count of transactions in last period • Amount of transactions in last period **Customer demographics data:** • Age/age group • Marital status • Tenure of customer **Geospatial data:** • Transaction location • Geographical location of customer **POS data:** • Merchant data • Vendor data **Accounts holding data:** • Account information **Time-specific data:** • Time interval of transactions • Weekdays/weekend indicators • Month-end/quarter-end indicators	**Historical payment information:** • Number of on-time payments in last periods • Amount of missed payments in last periods **Customer lifecycle data:** • Number of active accounts holding data **Historical delinquency data:** • Number of times the customer has been identified as delinquent in last periods
Customer Attrition Model		**Customer lifecycle data:** • Number of active accounts holding data NPS Score **Cost to customer data:** • Customer purchase amount • Customer purchase products **Customer engagement data:** • Call history with customer executives
Recommendation Model		**Product data:** • Product information data • Historical product search data **Spend pattern data:** • Historical spend in product type and categories **User generated content — ratings and reviews data:** • Ratings and reviews of the product provided by the customer

Figure 4

How to build a feature store

Digital-native organizations, like LinkedIn,[7] Uber and AirBnB,[8] have achieved ML scale to serve all of their AI needs across all their products and regions from a single enterprise-wide ML system. In each case, a feature store plays a central role in training and serving ML models.

For example, Uber's Michelangelo[9] framework democratizes ML and makes scaling AI to meet the needs of business as easy as requesting a ride. It has a centralized feature store for collecting and sharing features. Its platform team curates a core set of widely applicable features and data scientists contribute more features as part of the ongoing model-building process. And metadata created for each feature is used to track ownership, how it is computed and where it is used. It provides functionality to select features by name and join keys, and both online and offline pipelines are auto-configured.

Building a feature store can be broken down to development of four key functional areas — feature extraction, feature selection, feature synthesis and feature governance — along with other functionalities (see Figure 5).

The technical solution design has to be in sync with existing IT technologies and related policies, as introducing a new technology stack in any digital immigrant is a long, drawn-out process.

Key functionalities that must be built in a feature store in a phased manner

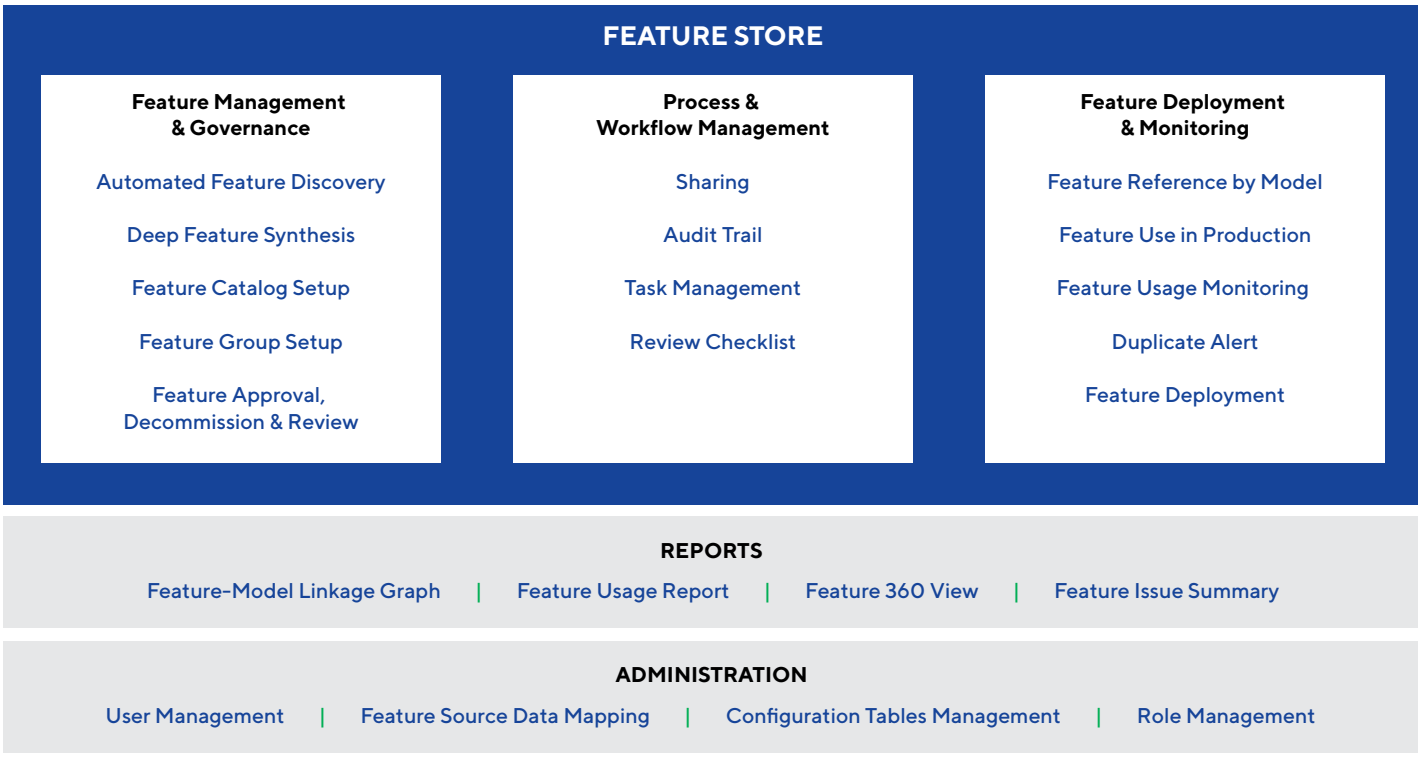


Figure 5

Key components required to build a feature store

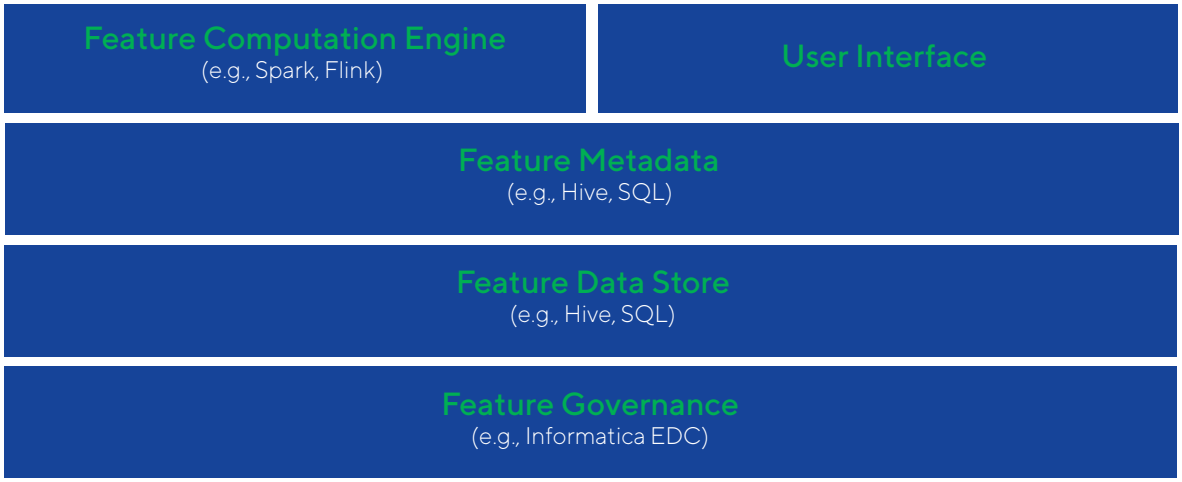


Figure 6

Figure 6 offers a peek inside into the key technical components required and suggested technologies to achieve them.

- **Feature computation engine:** This executes the feature engineering jobs (scheduled or ad hoc) using standard frameworks.
- **User interface:** An interface for DS (consumers) and ML engineers (producers) to explore and use features for creating training data sets and supporting production models. The store provides feature information like ranking, definition, version and new request for achieving operational efficiencies in the ML model lifecycle.
- **Feature metadata:** A storage layer that retains feature documents like owners, definitions, versions, hierarchies, etc. which is referred to for feature discovery.
- **Feature data store:** A data layer wherein computed features are stored for easy access either through an API or UI.
- **Feature governance:** Essential component of the feature store that governs the access and rights to a feature. As the feature store expands with thousands of features, controlling access to sensitive features and retiring unused features becomes more important.

How feature stores function

To build and deploy models rapidly, organizations must address the need of feature computing and serving as it takes significant amounts of time for them to compute. This can be solved by building a feature store within each data store and using a shared feature computation engine for serving features to both batch and real-time models. In an ideal scenario, features for each model would be pulled directly from the feature store with reference to common entities like customer IDs. Figure 7 (next page) illustrates how data flows into a feature store and how it serves both model training and production.

Data flow in a feature store ecosystem

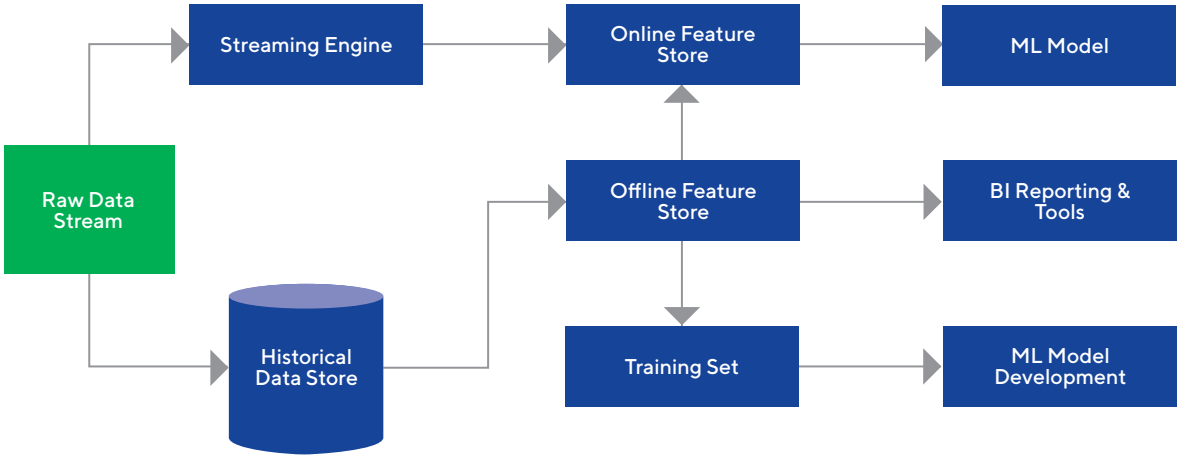


Figure 7

How to generate and capture a feature in a store

There are several approaches to creating features in a feature store:

- **Automated feature synthesis:** Automated feature engineering has become a key ML research area. It has led to the proliferation of frameworks that can automatically synthetize features from one or multiple data tables in relational data stores. Genetic algorithms offer another approach not only for feature selection but also for generation.[10]
  - > One of the methods for populating feature stores is deep feature synthesis,[11] which can automatically derive predictive models from raw data. This is used to automate the population of feature stores for structured, transactional and relational data sets.
  - > To achieve this automation, we first propose using deep feature synthesis algorithms to automatically generate features for relational data sets. The algorithm follows relationships in the data to a base field, and then sequentially applies mathematical functions along that path to create the final feature.
  - > Second, we implement a generalizable ML pipeline and tune it using a novel Gaussian copula process-based approach.[12]
- **Automated extraction of features from the existing model** can be achieved by using the REGEX approach, fuzzy matching and entity extraction.
- **Manual creation of new and custom features** as identified by the data scientist.
- **Feature DNA:** Predict feature engineering based on attribute usage in new models (e.g., guided feature engineering).[13]

Challenges while populating features in a feature store

As enterprises embark on building shared feature stores and computation engines, the following must be addressed to achieve success:

- **Feature importance:** A feature store provides important attributes to ML algorithms, but

having too many precomputed elements doesn't necessarily guarantee increased model accuracy and it also means additional computation and storage increases in processing overhead. Defining the right feature level, historical duration and hierarchy is key.

- **Feature processing:** As enterprises generally have different processing engines and stores for real and batch data streams, serving both through a common feature store can require complex reengineering.
- **Feature complexity:** Derived features can be complex; it takes time to compute such complex features in real time or in the production environment. Data scientists and feature store governors need to agree as to what needs to be preprocessed and which features need to be served directly in production.
- **Operating model:** Enterprises will have multiple data silos and varied data storage technologies. Defining the right operating models for functionalities that need to be centralized and decentralized in a feature store will aid adoption across the enterprise. See Figure 8 for an illustration of the pros and cons of centralized vs. decentralized stores.

Feature store operating model pros & cons

	PROS	CONS
Centralized	▪ One feature execution environment — centralized historical information on logs/execution during fails. ▪ One version control environment — ability to see history, enforce peer review before rollout of new version/updates (just like with any software updates to prod code using Github as an example). ▪ Ability to manage central execution in federated environments (H20, IPython, Spark, etc.). ▪ Centralized database of ALL features (no hidden information, or some teams not knowing what others are doing and, usually, doubling their work instead of reusing). ▪ If for example, S3 is used as the input and output for each feature engineering task — then central environment is a key for any future migrations (migrating from one environment) rather than migrating a spaghetti of different team jobs without central view/control — this future-looking view in terms of future migrations/changes is probably the most compelling. ▪ Data governance process to validate new rollouts/changes against duplication. ▪ Next gen features on top of MDM/data governance like natural language search of the features. ▪ Ability for BI-level users to find data they need, understand it and use it for reporting, etc.	▪ Create overflow jobs to the execution infrastructure (H20, IPython, Spark, etc.) but cannot manage that execution environment directly — need to manage execution in DAG fashion. ▪ Hard to control/failover final execution environment. ▪ Teams are forced to use central data governance/execution governance tool (e.g., Domino + Alation) which slows down a bit going to production/changes.
Decentralized	▪ Ad hoc infrastructure usage — independent execution environments, so there is less overflow risk. ▪ Teams can create their own data governance/execution environment per their own strong views. ▪ Easier to put into production/changes to a small sub-environment.	▪ Migration is a huge problem — any central changes will lead to months of reengineering in a decentralized fashion prone to mistakes. ▪ No central view of all features — hard to reuse/avoid duplication of effort. ▪ BI users can't see what they can use.

Figure 8

Feature store business benefits

A feature store does not merely bring cost and operational efficiencies through shared computation and storage; it also enables the enterprise to achieve other business benefits like faster go-to-market, improved model accuracy and enhanced IT agility (see Figure 9).

Key feature store benefits

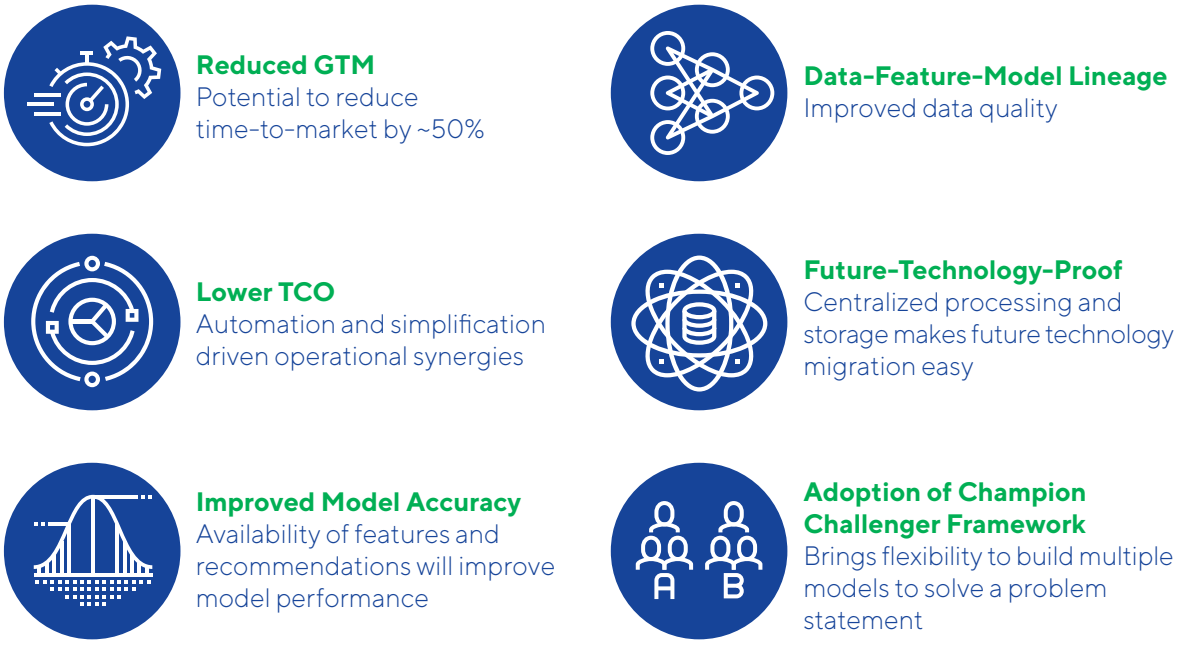


Figure 9

A feature store does not merely bring cost and operational efficiencies through shared computation and storage; it also enables the enterprise to achieve other business benefits like faster go-to-market, improved model accuracy and enhanced IT agility.

Quick Take

How a Bank Used AutoML & a Feature Store to Enhance Fraud Detection

A large global bank wanted to radically transform its credit and debit card transaction fraud scoring by moving away from a rule-based system to an AI model-led decision engine. The objective was to design a new architecture to build AI models as challenger models to existing systems for reducing false positives and improving the fraud detection rate.

The solution adopted was a new fraud modeling ecosystem with AutoML pipelines which supported the requirement of running four different fraud models (batch and real time) as part of a champion/challenger framework: The champion was the incumbent model handling transaction scoring, while the challenger model offered an independent model that processed the same transaction in parallel to the champion model before a rule system chooses which model score to rely on for transaction approval.

The flexibility of switching models between batch and real time was achieved by developing a feature store with hundreds of features (real time and batch) to support all existing and in-flight models. The new system was able to reduce false positives by more than 80%; and with the improved fraud detection rate, overall savings were more than $60 million.

Looking ahead

As digital immigrant enterprises ascend the ML maturity ladder and embrace new methodologies, frameworks and technologies used by digital native businesses, they should be mindful that their technology platforms, organizational structures, business use cases, IT policies and governance, and regulatory environment are completely different and complex. Frameworks like Uber's Michelangelo need to be customized to individual needs.

Following some initial success in ML projects, it is imperative that the organization focuses on how to translate that individual success into an ML program success. As the journey evolves, focus needs to be on change management, governance and automation to achieve at-scale ML objectives. With a focus on streamlining and automation in data engineering, building a feature store is the solution.

Building a feature store is a gradual process and should be seen as such, since with every new model built, the set of features available expands with the store, which influences the development of subsequent models as more features become available for exploration and modeling.

Enterprises should consider the following steps:

- **Build a feature store for databases that are commonly used, like customer records or transactional data sets** (e.g., payment record, order history, etc.), which are referenced in most business use cases such as customer acquisition, retention, fraud and KYC. Understanding that they are used for modeling and reporting will help in deriving the right feature groups to create, compute and share.
- **Start by targeting commonly performed featured computations like converting categorical to numerical variables, one-hot-encoding, feature binning, aggregates and transformations**, as these tasks are performed by data scientists in nearly all model iterations. This could be followed by more complex features like joining multiple tables and performing nested functions, or complex approaches such as automated feature extraction from existing models and feature recommendation engines.
- **Pursue change management in terms of how model development and deployment need to be managed and governed.** Every data scientist has their own tool preferences and preferred ways of performing data analysis in their daily activities. Establishing feature stores will require that existing preferences and processes will be changed or new ones implemented. Governance must be put in place so that set guidelines on feature computation, model development strategies, etc. are followed by key stakeholders like ML engineers and data scientists.
- **Establishing requisite feature store governance is essential from day zero**, as no enterprise wants to publish and maintain duplicate, decaying features. The governance needed must provide that fine balance between promoting innovation (i.e., identification of new features by data scientists) and mandating the use of features from the store.

Endnotes

1 For this report, we will be using machine learning (ML) and AI interchangeably with no strong distinction.

2 Lorica, B., & Nathan, P. (2019), *AI Adoption in the Enterprise.*

3 Lorica, B., & Nathan, P. (2019), *Evolving Data Infrastructure.*

4 Lorica, B., & Nathan, P. (2019), *The State of Machine Learning Adoption in the Enterprise.*

5 Machine learning (ML) is a subset of artificial intelligence (AI) that enables systems or applications to self-learn and improve from each experience without being explicitly programmed.

6 D. Sculley, G. H., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., & Young, M. "Machine Learning: The High-Interest Credit Card of Technical Debt." Google Inc. (2014).

7 https://engineering.linkedin.com/blog/2019/01/scaling-machine-learning-productivity-at-linkedin.

8 Atul Kale and Xiaohan Zeng, "ML Infra @ AirBnB." https://cdn.oreillystatic.com/en/assets/1/event/278/Bighead_%20Airbnb_s%20end-to-end%20machine%20learning%20platform%20Presentation.pdf.

9 Hermann, J., & Balso, M.D. (2017, Sept. 5). "Meet Michelangelo: Uber's Machine Learning Platform." Retrieved from eng.uber.com; https://eng.uber.com/michelangelo/.

10 Julio, M.R. C., Luna-Rosas, F. J., Miguel, M.G., Alejandro , C.O. & López-Rivas, V. (2012). *Optimal Feature Generation with Genetic Algorithms and FLDR in a Restricted Vocabulary Speech Recognition System.*

11 James Max Kanter and Kalyan Veeramachaneni, *Deep Feature Synthesis: Towards Automating Data Science Endeavors.*

12 Ibid.

13 Muthiah, P., & Li, J. (2018, Nov. 13). "Airbnb Engineering and Data Science." Retrieved from medium.com: https://medium.com/airbnb-engineering/druid-airbnb-data-platform-601c312f2a4c.

About the authors

Amit Agarwal

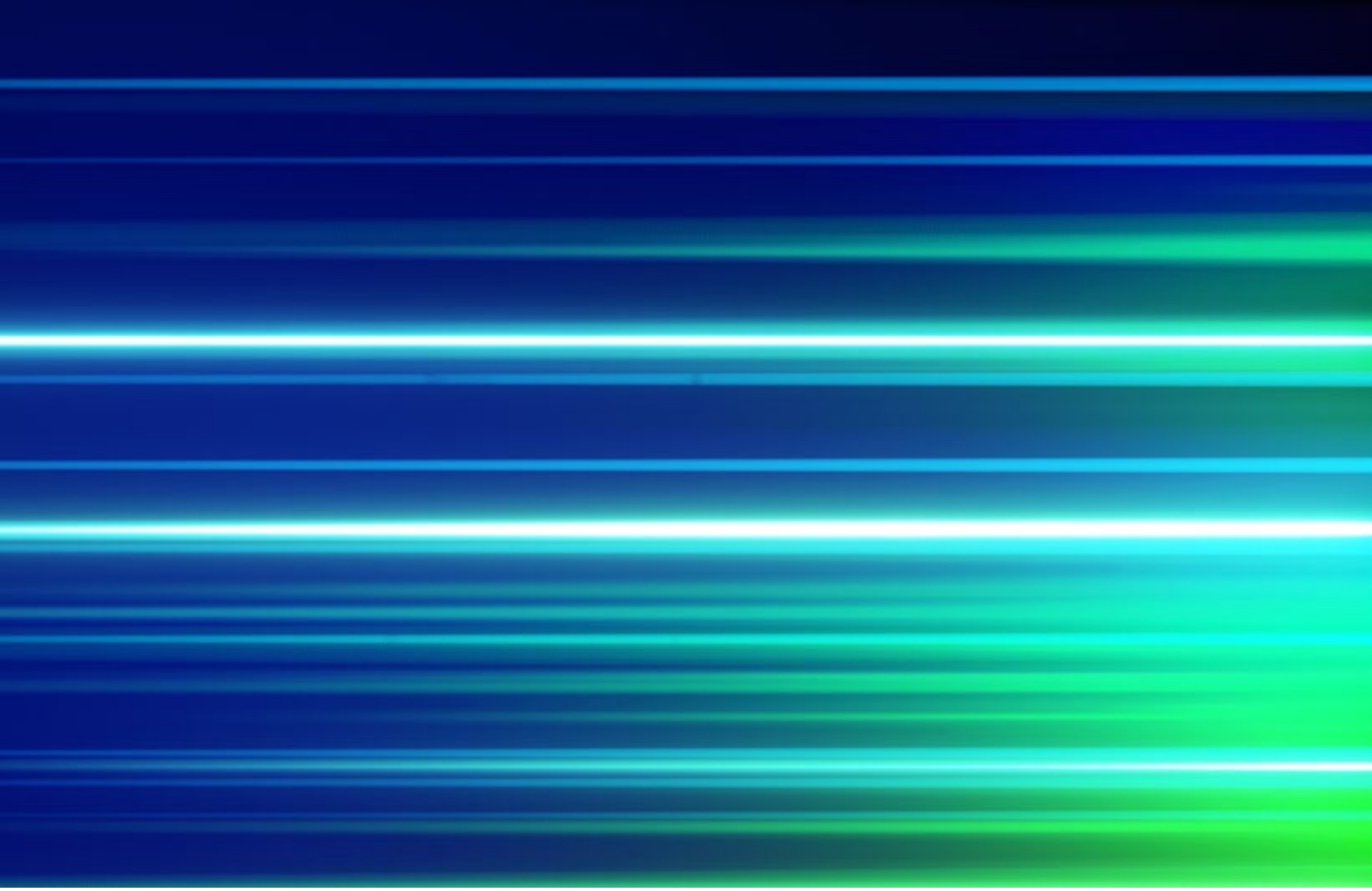
Associate Principal Data Scientist, Cognizant

Amit Agarwal is Associate Principal Data Scientist in Cognizant's AI, Data Sciences and Machine Learning Practice in UK&I, where he is responsible for leading work in advanced analytics, AI and ML work conducted for banking and financial services clients. In his role, Amit leads team of data scientists and ML experts engaged in developing and delivering innovative AI solutions for banking clients across various domains. He works closely with clients to help them achieve their strategic objectives, architecting AI solutions, delivering large-scale complex AI/ML solutions and consulting on ML as a service (MLaaS) propositions. Amit also collaborates with the wider fintech and partner ecosystem to create disruptive industry solutions and propositions, and he is responsible for data sciences' go-to-market strategy in the UK&I market. He has over 15 years of experience in modeling and analytics consulting including at UBS and PriceWaterhouseCoopers. He also holds a CFA charter and has FRM certification. Amit can be reached at Amit.Agarwal@cognizant.com | www.linkedin.com/in/amit-agarwal-98a4969/?originalSubdomain=uk

Matthew O'Kane

European AI & Analytics Practice Lead, Cognizant

Matthew O'Kane leads Cognizant's AI & Analytics practice across Europe. His team helps clients modernize their data and transform their business using AI. Matthew brings close to two decades of experience in data and analytics, gained across the financial service industry and in consulting. He joined Cognizant after leading analytics practices at Accenture, EY and Detica (now BAE Systems Applied Intelligence). Over this period, he has delivered multiple large-scale AI/ML implementations, helped clients transition analytics and data to the cloud and collaborated with MIT on new prescriptive ML algorithms. He brings a passion for the potential for AI and analytics to transform clients' businesses across functional areas and the customer experience. Matthew lives with his wife and two children in Winchester, England. He's an avid cook, enjoying everything from baking with his daughter to experimenting with 'sous vide' techniques. He can be reached at Matthew.OKane@cognizant.com | www.linkedin.com/in/matthewokane/.



About Cognizant Artificial Intelligence Practice

As part of Cognizant Digital Business, Cognizant's Artificial Intelligence Practice provides advanced data collection and management expertise, as well as artificial intelligence and analytics capabilities that help clients create highly-personalized digital experiences, products and services at every touchpoint of the customer journey. Our AI solutions glean insights from data to inform decision-making, improve operations efficiencies and reduce costs. We apply Evolutionary AI, Conversational AI and decision support solutions built on machine learning, deep learning and advanced analytics techniques to help our clients optimize their business/IT strategy, identify new growth areas and outperform the competition. To learn more, visit us at www.cognizant.com/ai.

About Cognizant

Cognizant (Nasdaq-100: CTSH) is one of the world's leading professional services companies, transforming clients' business, operating and technology models for the digital era. Our unique industry-based, consultative approach helps clients envision, build and run more innovative and efficient businesses. Headquartered in the U.S., Cognizant is ranked 193 on the Fortune 500 and is consistently listed among the most admired companies in the world. Learn how Cognizant helps clients lead with digital at www.cognizant.com or follow us @Cognizant.

Cognizant

World Headquarters
500 Frank W. Burr Blvd.
Teaneck, NJ 07666 USA
Phone: +1 201 801 0233
Fax: +1 201 801 0243
Toll Free: +1 888 937 3277

European Headquarters
1 Kingdom Street
Paddington Central
London W2 6BD England
Phone: +44 (0) 20 7297 7600
Fax: +44 (0) 20 7121 0102

India Operations Headquarters
#5/535 Old Mahabalipuram Road
Okkiyam Pettai, Thoraipakkam
Chennai, 600 096 India
Phone: +91 (0) 44 4209 6000
Fax: +91 (0) 44 4209 6060

APAC Headquarters
1 Changi Business Park Crescent,
Plaza 8@CBP # 07-04/05/06,
Tower A, Singapore 486025
Phone: + 65 6812 4051
Fax: + 65 6324 4051

Codex 4971.02