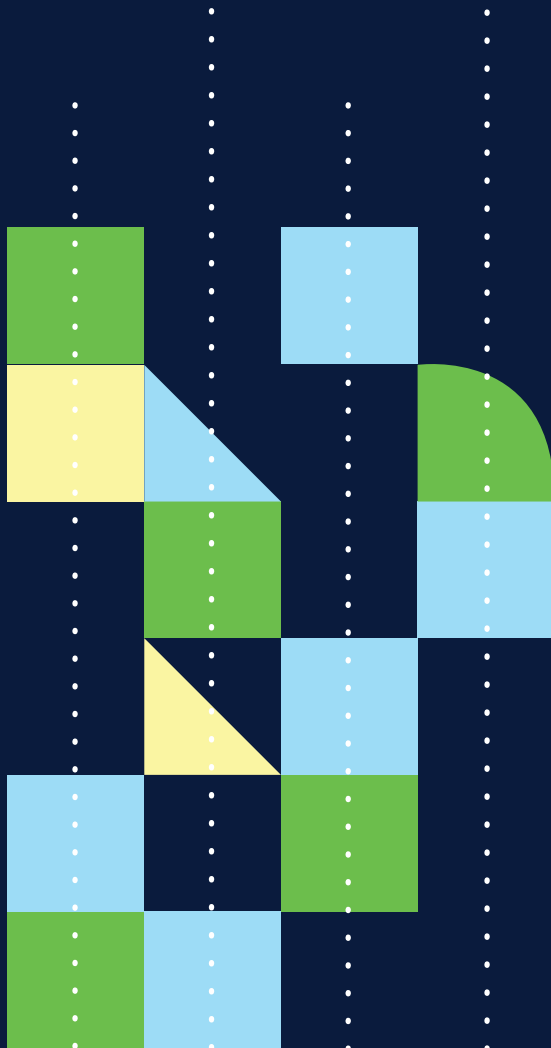


# Unsupervised Machine Learning with DataVisor

DISCOVER HOW DATAVISOR'S PIONEERING  
UNSUPERVISED MACHINE LEARNING ENGINE BROKE NEW  
GROUND IN THE FIGHT AGAINST MODERN DIGITAL FRAUD,  
AND LEARN WHY IT'S STILL THE REIGNING STANDARD  
FOR PROACTIVE, REAL-TIME FRAUD PREVENTION.



DataVisor's Unsupervised Machine Learning Engine is built upon our patented and proprietary unsupervised machine learning (UML) algorithms. The DataVisor UML Engine is a central component of both our approach and our solutions and is widely deployed across industries and enterprises.



# Table of Contents

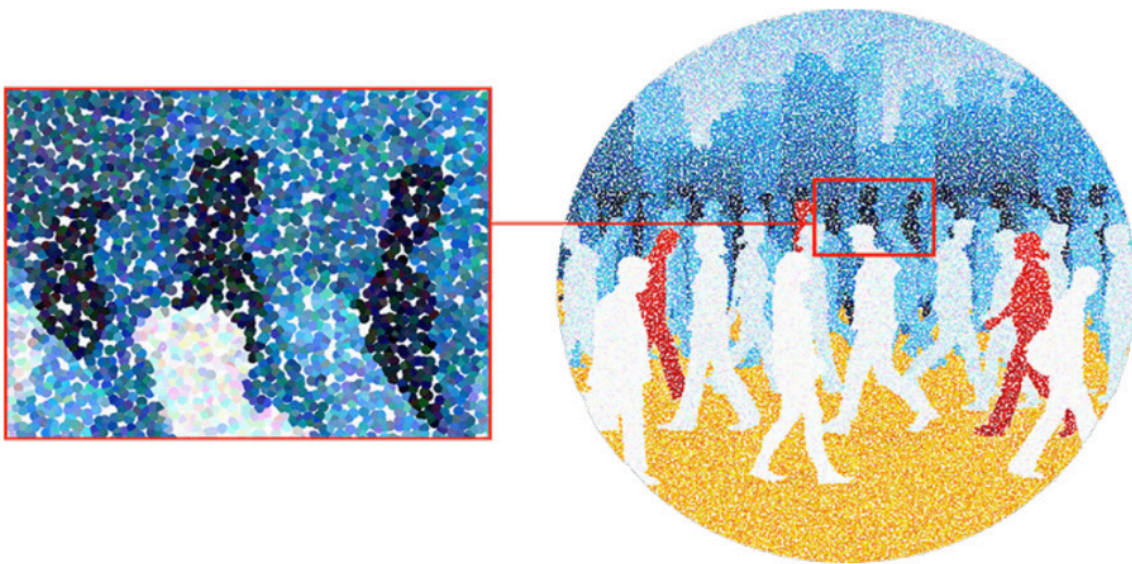
<b>ABOUT UNSUPERVISED MACHINE LEARNING.....</b>	<b>3</b>
<b>ABOUT THE DATAVISOR UNSUPERVISED MACHINE LEARNING ENGINE.....</b>	<b>4</b>
The DataVisor UML Engine Difference .....	4
Additional Benefits of the DataVisor UML Engine .....	5
The DataVisor UML Engine in Action .....	6
<b>A TECHNICAL DEEP DIVE .....</b>	<b>7</b>
<b>STEP #1: DYNAMIC FEATURE EXTRACTION .....</b>	<b>9</b>
<b>STEP #2: UNSUPERVISED ATTACK RING DETECTION .....</b>	<b>12</b>
<b>STEP # 3: SUPERVISED LEARNING DETECTION (OPTIONAL) .....</b>	<b>16</b>
<b>STEP #4: RESULT CATEGORIZATION AND RANKING .....</b>	<b>17</b>
<b>ARCHITECTURE AND ACHIEVING REAL-TIME DETECTION.....</b>	<b>18</b>
<b>CONCLUDING THOUGHTS .....</b>	<b>20</b>

# About Unsupervised Machine Learning

UML is a category of machine learning that works without requiring labeled input data. Instead, it infers a function to describe the hidden structures of “unlabeled” input data points.

Common UML approaches today include anomaly detection techniques that attempt to identify outliers, and clustering and graph analysis techniques that focus on studying the relationships and connectivity among input data.

Using the clustering method (the separation of data into groups of similar objects), an algorithm gathers observations into groups one by one, with each group containing one or more features. In the process, UML focuses on estimating connections strength between all data points, and thus, a UML algorithm can begin forming clusters once it learns how to recognize similarities.



*Figure 1: The DataVisor UML engine looks across all accounts (like dots in a painting) to see the overall picture*

# About the DataVisor Unsupervised Machine Learning Engine

The DataVisor UML Engine is developed based on this latter approach, combining clustering techniques and graph analysis algorithms to discover correlated fraudulent or suspicious patterns from unlabeled data. By analyzing the distance and connectivity between data points that represent accounts and their activities across a large time period, the DataVisor UML Engine can automatically discover new fraud attacks and techniques, including mass registration, account takeover, money laundering, and more.

## THE DATAVISOR UML ENGINE DIFFERENCE

The DataVisor UML Engine differs from other approaches in several ways. Some of its differentiating characteristics include the following:

- ▶ **Proactive detection of new attacks**

Because the UML Engine does not require labels or training data, it can start delivering detection results quickly, and in real time. This enables early detection and adaptive responses towards changing attack patterns. The UML Engine regularly powers detection rate increases of 30-50% over existing systems and can detect malicious activity even at the point of account application or registration.







- ▶ **Real-time account correlation**

The UML Engine processes all events and account activities together to analyze the correlations and similarities across millions—or even hundreds of millions—of accounts. It is able to reveal subtle, hidden structures across fake, fraudulent, and malicious accounts in real-time.

- ▶ **Effective digital signal management**

The UML Engine both feeds into—and ingests information from—the DataVisor Global Intelligence Network (GIN). The GIN is optimized for the consolidation and computation of digital intelligence from multiple digital fingerprints. In addition, the GIN aggregates learned attack patterns to derive fine-grained, rich signals that improve the overall detection performance of the UML Engine.

## Comprehensive Fraud Intelligence that Provides Fine-Grained Signals and Risk Scores

 410 Million+ IP addresses	 5.3 Million+ User agent strings
 3.6 Million+ Email domains	 160,000+ Device types
 300,000+ OS versions	 700,000+ Phone prefixes

## Insight from 4.1 Billion+ Users and 800 Billion+ Events







 Financial Services	 E-Commerce	 Social Platform
 Mobile & Gaming	 Telecom & Travel	 Insurance

Figure 2: DataVisor Global Intelligence Network

## ADDITIONAL BENEFITS OF THE DATAVISOR UML ENGINE

In addition to its distinctions against other detection methods, the DataVisor UML Engine has the following additional advantages:

### ► Input data flexibility

Common questions for any machine learning algorithm are what input data fields are needed and how much data is required to be effective. The Data Visor UML Engine is notably more tolerant of missing data fields and low data volumes.

### ► Low false positives

Unlike anomaly detection and other unsupervised approaches that generate many false positives, the DataVisor UML Engine delivers highly accurate results and can be used directly through APIs without manual review. Accuracy rates are uniformly above 90% and regularly meet or exceed 99%.

### ► Low re-tuning overhead

The UML Engine does not require frequent retunings as its predictive power is not based on intelligence derived from historical experience. It is proactive and adaptive to new changing attack patterns, and can, therefore, maintain long-term high performance without re-tuning. This is different from supervised machine learning models, which decay in effectiveness over time.

### ► Transparent detection reasons

Clusters of correlated accounts detected by the UML Engine have clear indicators of what attributes and values are shared across them, and what hidden links are present. As a result, the UML Engine outputs more transparent and convincing detection reasons as compared to other machine learning-based approaches.

## THE DATAVISOR UML ENGINE IN ACTION

Here is an example of how the UML Engine detected a real-world fraud ring that had previously evaded other detection systems. The fraud ring in this anonymized example comprises over 200+ credit card accounts from a large banking institution.

The accounts all resided in low-risk regions, had high FICO scores, matched bureau data, and were not in the existing fraud database. In short, they did not have any risky signals that were similar to any previously known or seen attacks. As a result, all of these accounts passed safely through existing fraud detection systems.

The DataVisor UML Engine not only examined the above data dimensions but also looked at other digital attributes of credit card applications across all accounts. In doing so, the UML Engine uncovered subtle suspicious correlations that were indicative of the presence of a fraud ring:

- ▶ All the emails evidenced a shared pattern of having been created using the account holder's first name, last name, initial, and birthday.
- ▶ The IP addresses were all associated with high-risk data centers.
- ▶ The accounts all used an older-model iPhone—iPhone 5 or 5s—with the same OS version.
- ▶ All of the accounts performed their activity with Chrome, despite Safari being the default browser app for iPhones.

In this use case, the UML Engine was able to reveal that these 200+ accounts belonged to the same fraud ring, and was able to do so right at the point of application. The UML Engine assigned the accounts a score of 0.92 (out of 1.0), indicating a high likelihood of their being fraudulent.

# A Technical Deep Dive

The DataVisor UML Engine also works in concert with other DataVisor tools, including the Supervised Machine Learning Engine, the Global Intelligence Network (GIN), and the Automated Rules Engine, which maintains the transparent nature of rules-based systems but automatically suggests rules to be created, modified, or deprecated.

As shown in Figure 3, the input to the UML Engine is raw data in the form of either continuous event streams (e.g., from a real-time integration setting) or multiple batch input files that describe user account profiles or different types of account activities (e.g., in a batch integration setting with various event logs). In a real-time setting, the output is a score, and reason codes, for each input event. In a batch setting, the output is a list of detected suspicious accounts with associated scores and reason codes. The UML Engine output also feeds into the threat dashboard, which visualizes attack rings and allows investigation of each detected incident and account.

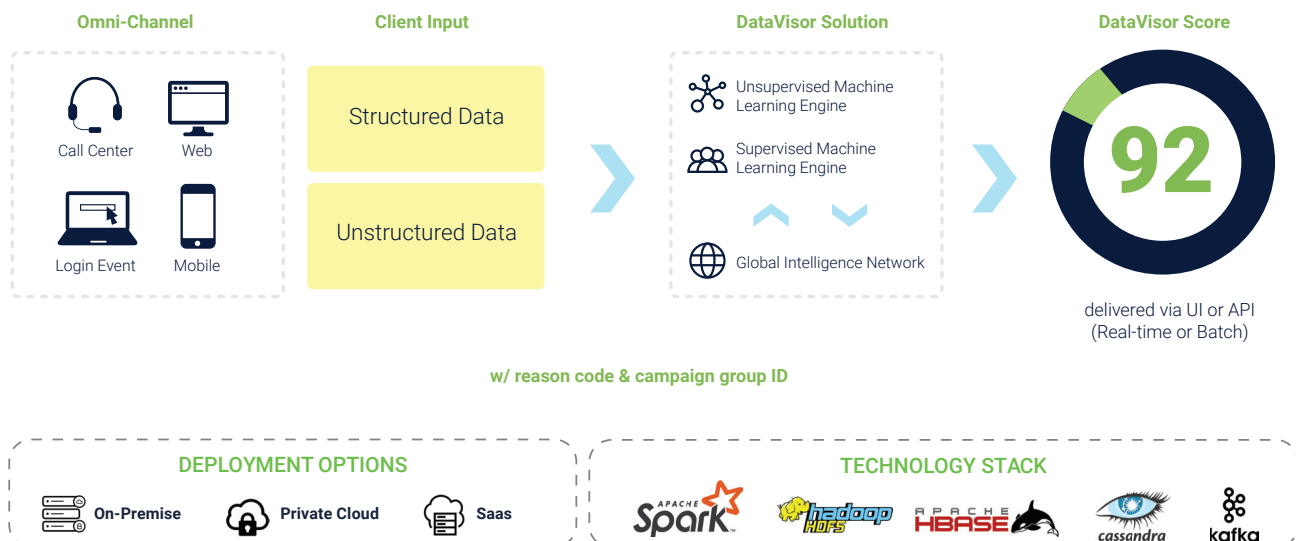


Figure 3: The DataVisor UML Solution Architecture

In between the input and output, there are four major steps that the UML Engine can perform:

- ▶ Dynamic feature extraction
- ▶ Unsupervised attack ring detection
- ▶ Supervised detection (optional)
- ▶ Result categorization and ranking

These four steps precisely identify fraudulent and malicious accounts that combine as large fraud rings. Detecting such a fraud ring is done by discovering the hidden links among the corresponding account-based data points. All of the algorithms are run on top of a distributed big data infrastructure with algorithmic optimizations that enable real-time detection.



# Dynamic Feature Extraction

The goal of unsupervised machine learning is to generate a comprehensive and meaningful set of features to describe each input account. The “unsupervised” nature means the system has no prior knowledge of new attack patterns, nor does it know in advance which features will be effective. The UML Engine is accordingly designed to operate across a very high-dimensional feature space and be comprehensive in extracting features.

The UML Engine generates the following categories of features to describe each user account, as shown in Figure 4.

- ▶ **Profile information:** Demographic information associated with an account, usually provided at the point of application or registration. Information may include user account nickname, income range, age, gender, and address.
- ▶ **Behaviors and activities:** What the account has done, and when; for example, payment events which include a timestamp, payment amount, and method.
- ▶ **Origins and digital fingerprints:** Information describing the access methods of an account, including its device type and version, browser information, IP address, and geographic origins.
- ▶ **Contents and metadata:** Text and pictures generated by an account, such as comments, profile photos, and phone call records.
- ▶ **Relationships between accounts:** Interactions and relationships between different accounts; for example, one account sending money to other accounts that are friends or contacts.

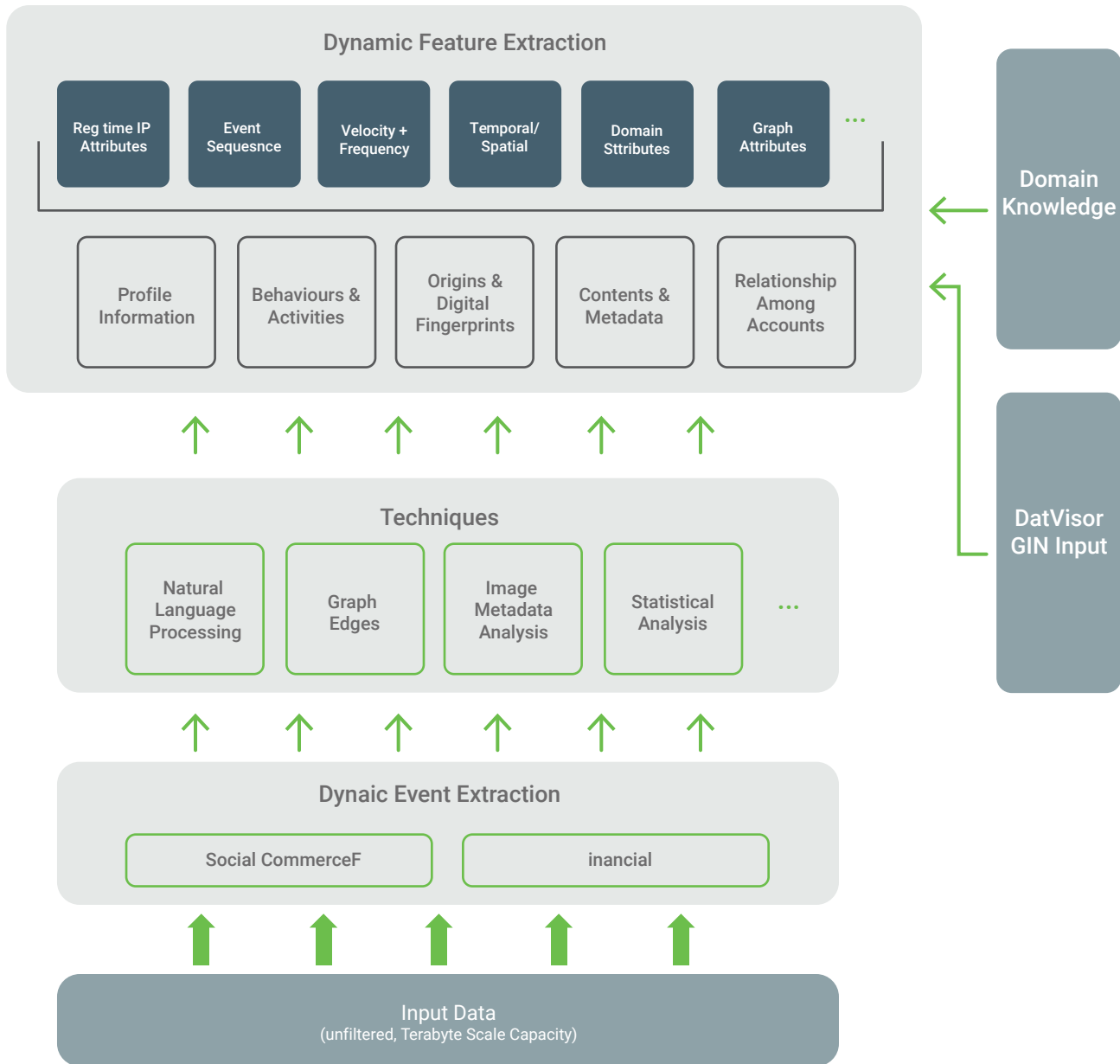


Figure 4: The UML Engine Dynamic Feature Extraction. The top line shows some example features that DataVisor extracts from the raw input data.

For unstructured text, the UML Engine leverages Natural Language Processing (NLP) techniques to derive rich textual and contextual features. For image input, the UML Engine supports meta-data attributes such as image title, creation time, location, generation source, format, and resolution (this can be expanded to support more sophisticated features). For relationship information between accounts, the UML Engine describes them using graph attributes such as directional and bi-directional links, degrees of nodes, edge weights, graph neighborhood features, and community features.

This process is dynamic in that, for each category, the UML Engine will derive as many features as applicable based on the input data schema—sometimes as high as hundreds of thousands. In certain cases, literally millions of features are created. For example, based on input account event types and sequences, the UML Engine can derive a variety of features including event frequencies, velocities, time interval gap distributions, diurnal patterns, and sub-sequence patterns. When the input data fields need to change or increase over time, the number of feature dimensions can automatically adjust accordingly.

The UML Engine's dynamic feature extraction differs from other feature engineering approaches in four major ways:

- ▶ **Unrivaled domain expertise:** The features created from input data are designed based on decades of experience researching and fighting real-life fraud, abuse, money laundering, and more.
- ▶ **Enhanced intelligence:** The features are enriched with DataVisor's Global Intelligence Network, particularly for digital information.
- ▶ **Broad scope:** The features describe both structured and unstructured input data, as well as a variety of relationships across accounts, whereas most existing solutions are based on a fixed set of pre-defined features from structured input data.
- ▶ **Flexibility:** The list of features dynamically expands when input data has more fields in the schema.

## STEP

# #2

# Unsupervised Attack Ring Detection

In this step, the UML Engine performs correlation analysis across all accounts and identifies attack rings. An attack ring can be broadly defined as a collection of malicious accounts that have strong similarities or correlations in their features, and which are likely operated by a single individual or group of attackers. Additional steps within this process include clustering and graph analysis.

## STEP 2.A: CLUSTERING ANALYSIS

Based on the input feature vectors, the UML Engine first identifies suspicious clusters of accounts that have strong similarities or correlations in the high-dimensional feature space. The keys to this step are **reducing feature dimensions** and **determining the distance function** that computes the distance between data points. When the distance function is properly designed with a subset of important features, only truly suspicious accounts will form tight clusters while legitimate accounts will not group.

Other methodologies create clusters of accounts and then rely on anomaly detection approaches to separate groups. These approaches can generate very noisy results and typically require another layer of human intervention. In contrast, DataVisor's clustering approach is much more accurate at finding true attack rings.

The selection of features and distance functions are guided in the following manner:

- ▶ Important features are assigned higher weights; their weighted importance is based on the industry sector, and targeted attacks. Here we leverage rich domain experience to guide defining important feature categories for different setups. For example, knowing that an account's address is valid and that their information matches credit bureau data is important when detecting bank account application fraud. On the other hand, this information is less important when detecting spam attacks in social networks, where legitimate users often provide an empty or vague address when signing up.
- ▶ The DataVisor Global Intelligence Network (GIN) can optionally augment the UML Engine with initial detection results based on digital fingerprint signals. The initial results provided by GIN are typically noisy but provide insights that guide the selection of important feature dimensions and their weights. In other words, signals from GIN do not lead to final clustering results but enhance the confidence of feature selection. For example, sharing IP addresses from a hotel location is not very suspicious, while sharing proxy IP addresses is a lot more suspicious.
- ▶ The UML Engine performs a variety of statistical analyses of various feature distributions from the raw input data. These feature distributions also serve as input to help auto-infer the most important feature dimensions.

With the important feature dimensions and distance functions selected, the UML Engine can then group all data points to generate clusters. The clustering process is iterative and non-exclusive. The engine iteratively clusters data points based on different combinations of selected features, weights, and distance functions. As a result, each account can belong to different clusters, where each cluster is created by a different set of criteria. For example, an account can belong to a cluster with address and location features being more important dimensions, and simultaneously belong to another cluster with event sequences and behavior patterns being more dominant features.

The detection does not always depend on the existence of important features. A strong correlation of multiple accounts across a large number of seemingly unimportant features is also an indication of suspiciousness, and the UML Engine considers this case as well. For example, a group of accounts that follows almost the same behavior pattern (e.g., when they log in, when they log out, when they update account information) is still very suspicious, even when, traditionally, these behaviors may not be considered very important features for transaction fraud.

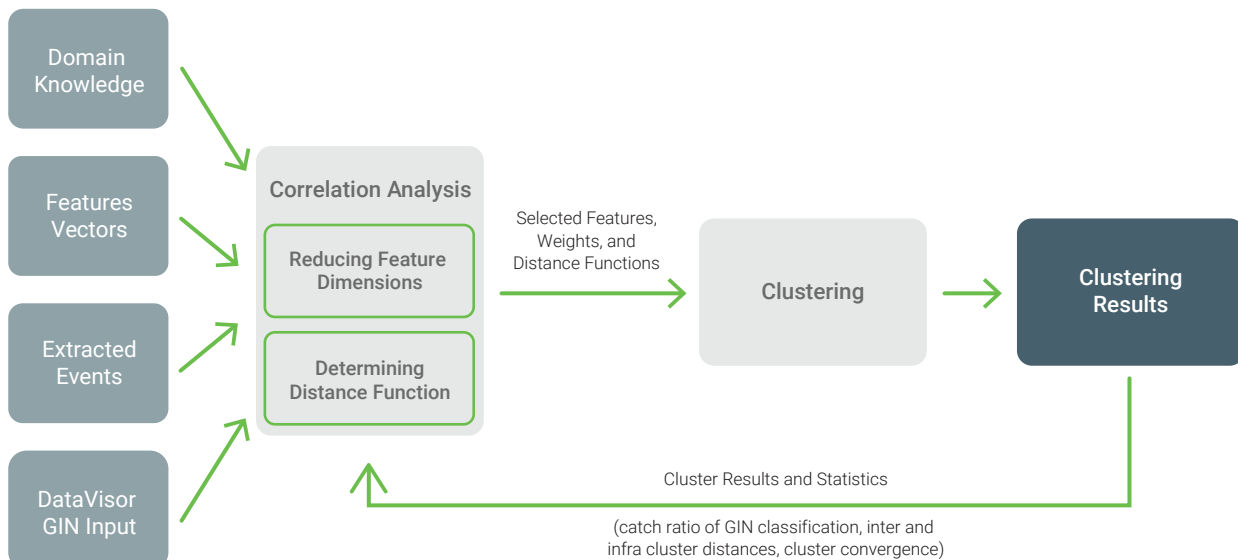


Figure 5: The UML Engine Clustering Analysis

During each iteration, the selected features, their weights, and the distance functions, are all automatically adjusted until they meet all of the following criteria:

- ▶ The ratio of intra-cluster distance to inter-cluster distance is small enough based on a threshold. The threshold can be pre-defined and can be adjusted according to the tradeoffs between coverage and false positives.
- ▶ The clustering results stabilize and converge.
- ▶ The output clusters capture at least a certain percentage of the Global
- ▶ Intelligence Network classification results. (Optional, as this can be a noisy measure.)

Given the high dimensionality of the data and the highly iterative process, the challenge is to bound the computation complexity so that the clusters can be computed and converge quickly, even in a real-time setting. The DataVisor UML Engine's efficient and scalable proprietary algorithms represent a major breakthrough for selecting features and computing distance functions with the following distinguishing characteristics:

- ▶ This approach is much more efficient than common methods of dimensionality reduction, such as Principal Component Analysis (PCA), and does not assume that the input data follows a linear relationship.
- ▶ The resulting clusters are highly accurate in pinpointing the exact fraud type, thanks to built-in domain knowledge and proprietary algorithms that automatically optimize clustering results towards different use cases.

## STEP 2.B: GRAPH ANALYSIS

Clustering results produce suspicious groups of accounts that are highly similar or correlated on either important features or over a large number of feature dimensions. The UML Engine further consolidates these results using graph analysis to link clusters that share similar accounts or strong features. As a graph problem, the clusters are nodes, and edges link these similar clusters. The edge weight between two clusters is a function of the number of shared accounts, the shared feature dimensions, and the cluster sizes.

This process examines a different aspect of connectivity between accounts. It discovers not just direct correlations but also indirect, transitive similarities and correlations across accounts (e.g., this process can group accounts A and C together if A is similar to B and B is similar to C). After the graph analysis, a “weak” cluster may be linked together with several “strong” clusters to raise the confidence of detection of this weak cluster. This process enhances the detection coverage and increases detection accuracy.

Tightly connected sub-graph components usually indicate the existence of attack rings.

## STEP

# #3

## Supervised Learning Detection (Optional)

This is an optional step where the output from the unsupervised attack ring detection can serve as training data to automatically train a supervised learning model and detect additional individual malicious accounts that share similar patterns with the already captured ones.

Compared to common supervised approaches, the DataVisor Supervised Learning Detection involves two different design decisions:

- ▶ It is optimized for fast and constant retraining without manual tuning so that the outputs from the unsupervised model can help adjust the model quickly.
- ▶ It is optimized for high accuracy and low false positive rates so that the output results can directly be applied to client production use.

This step outputs a set of detected individual bad accounts, which will then be combined with the detected attack rings to maximize detection coverage, according to coverage and false positive requirements.



# Result Categorization and Ranking

The final step is to rank detected accounts, assign them confidence scores, and categorize attack rings by the nature of their attacks.

All accounts are assigned a score from 0.0 to 1.0, with 0.0 being not suspicious at all and 1.0 being most suspicious. The scores help guide the policy setting for client actions, such as auto-blocking accounts that are 0.8 and above and manually reviewing the remaining ones above 0.0 (or another chosen threshold). Result ranking and score assignment uses a function based on the associated attack ring size and the corresponding cluster distances. Intuitively, the smaller the cluster distance is, and the larger the ring size is, the higher the score is.

Categorization is the last step in this process. This includes adding reason codes to each cluster, that highlight the important characteristics of the attack ring. To categorize attack rings, the UML Engine classifies them by many attributes including event types, attack techniques, bot versus human patterns, and account ages. The following shows some example categorizations:

- ▶ Automated account opening fraud
- ▶ Mass account takeover password testing ring
- ▶ Successful account takeover spam ring
- ▶ Manual transaction fraud
- ▶ Structured transfers indicative of money laundering
- ▶ Rapid transfer of funds indicative of money laundering

This categorization helps users to determine appropriate actions. For example, for account takeover fraud rings, a client may not wish to directly shut down the accounts, opting instead to contact the owners to help them recover their accounts.

# Architecture and Achieving Real-Time Detection

The entire DataVisor UML Engine—from computation to data access and storage—is built on a hyper-modern big data infrastructure. Apache Spark, HDFS, Hadoop, Apache HBase, and Casandra, Kafka are all being used to support the system in different capacities.

Figure 7 shows the system diagram. The left side of the figure is a batch UML engine implemented on top of Apache Spark. If real-time processing is required, the UML engine additionally includes a real-time component shown on the right side of the figure.



Figure 6: The UML Engine Big Data Infrastructure Stack

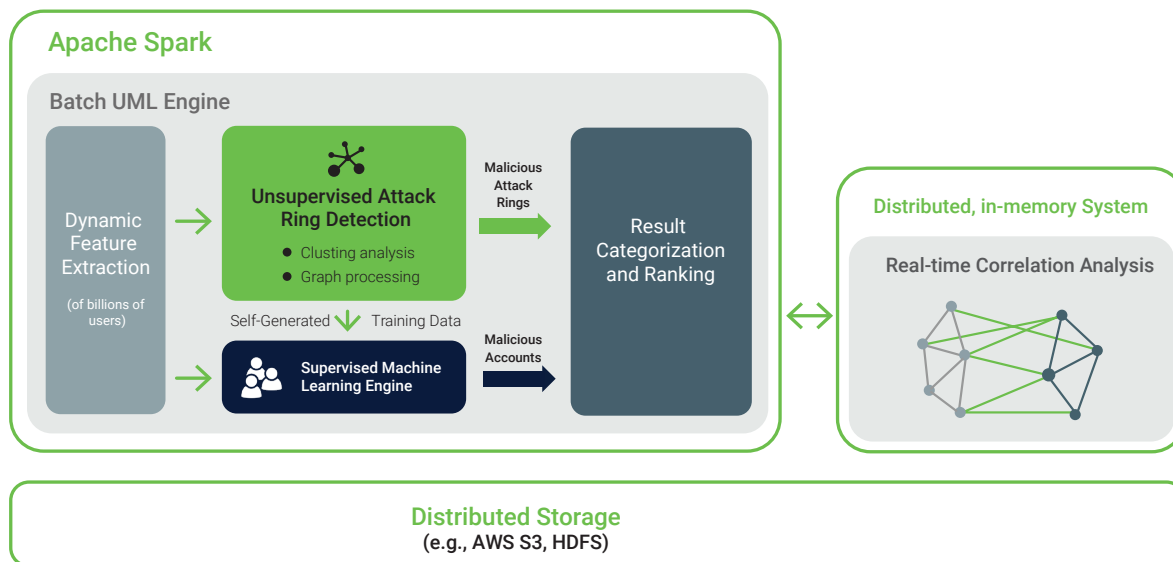


Figure 7: The UML Engine Architecture

The real-time component is implemented as a distributed, in-memory system with the same algorithmic logic as the batch system. As events come in, the real-time component keeps historical account states and continuously updates and evaluates them. The clustering and graph analysis is triggered—on demand, by an incoming event—to search and analyze a subset of related accounts and events. Due to memory and latency constraints, the system will perform a parallel search of the most relevant clusters and sub-graph components. This search expands when more space and time become available. The batch system periodically calibrates the real-time component for computational drifts caused by memory and latency constraints, so that the real-time results are close to the fidelity of a batch system.

Both the batch system and the real-time component are built using the same algorithms, but they are designed with different optimization goals. The batch system targets maximal coverage and accuracy, while the real-time system also needs to meet latency and throughput requirements. The performance of the real-time system with respect to the batch system is easily forecasted by latency and throughput requirements versus machine capacity. The real-time system is built to be horizontally scalable that requirements can be easily met with additional machine resources.

# 7

## Concluding Thoughts

Unsupervised machine learning technologies enable an entirely new way of combating even the most complex and sophisticated modern fraudsters. These technologies power more proactive and adaptive approaches for catching new and continually evolving attacks.

The key challenges are handling the vast volume of data in the new digital era, and being able to analyze all accounts and events at once to discover patterns rapidly and accurately.

The DataVisor UML Engine was the first of its kind in the field, and as the backbone of all DataVisor offerings today, it remains the leading unsupervised machine learning solution for addressing the challenges of modern digital fraud. To date, DataVisor protects more than 4.2 billion accounts

# About DataVisor

DataVisor is the leading fraud detection platform powered by transformational AI technology. Using proprietary unsupervised machine learning algorithms, DataVisor restores trust in digital commerce by enabling organizations to proactively detect and act on fast-evolving fraud patterns, and prevent future attacks before they happen. Combining advanced analytics and an intelligence network of more than 4B global user accounts, DataVisor protects against financial and reputational damage across a variety of industries, including financial services, marketplaces, ecommerce, and social platforms.

**For more information on DataVisor:**



[info@datavisor.com](mailto:info@datavisor.com)



[www.datavisor.com](http://www.datavisor.com)



967 N. Shoreline Blvd. | Mountain View | CA 94043



**DATAVISOR**