Big Data
Volume 00, Number 00, 2024
© Mary Ann Liebert, Inc.
DOI: 10.1089/big.2023.0065

Balancing Protection and Quality in Big Data Analytics Pipelines

Antongiacomo Polimeno,[1] Paolo Mignone,[2] Chiara Braghin,[1] Marco Anisetti,[1] Michelangelo Ceci,[2] Donato Malerba,[2] and Claudio A. Ardagna[1,*]

Abstract
Existing data engine implementations do not properly manage the conflict between the need of protecting and sharing data, which is hampering the spread of big data applications and limiting their impact. These two requirements have often been studied and defined independently, leading to a conceptual and technological misalignment. This article presents the architecture and technical implementation of a data engine addressing this conflict by integrating a new governance solution based on access control within a big data analytics pipeline. Our data engine enriches traditional components for data governance with an access control system that enforces access to data in a big data environment based on data transformations. Data are then used along the pipeline only after sanitization, protecting sensitive attributes before their usage, in an effort to facilitate the balance between protection and quality. The solution was tested in a real-world smart city scenario using the data of the Oslo city transportation system. Specifically, we compared the different predictive models trained with the data views obtained by applying the secure transformations required by different user roles to the same data set. The results show that the predictive models, built on data manipulated according to access control policies, are still effective.

Keywords: anomaly detection; access control; big data; data governance; data protection

Introduction

Big data techniques and solutions are pushing toward a data-driven ecosystem where decisions are supported by more accurate analytics and (privacy-aware) data management. The complexity of a data-driven ecosystem is amplified by the heterogeneity and diversity of both infrastructures/tools and data themselves. Data are in fact intrinsically heterogeneous and collected at high rates and with different formats from different sources in dynamic and collaborative environments.

This data-driven ecosystem comes with two conflicting requirements on data governance. On one side, there is the need to share data to extract value from them, that is, to support fast and accurate approaches for data collection, preparation, and analysis. On the other side, there is an increasing need for new solutions protecting data owners from an unregulated data management that could result in a violation of their private sphere. Existing approaches often target a single need independently, aiming to maximize either the quality data analysis by increasing data sharing[1] or the privacy of data owners by favoring data protection,[2–4] or either the performance in case of real-time stream processing.[5,6] As a result, there is a lack of proper solutions that balance the two needs, which are often replaced by *ad hoc* solutions in which each step of the pipeline execution being monitored separately for data governance compliance.[7–9]

This article presents a data engine architecture that supports the execution of big data analytics driven by a new data governance solution based on access control, which ensures a proper data management throughout the pipeline while maximizing the amount of shared data. It extends our work in Anisetti et al.[10]

[1]Dipartimento di Informatica, Università Degli Studi di Milano, Milan, Italy.
[2]Dipartimento di Informatica, Università Degli Studi di Bari, Bari, Italy.

*Address correspondence to: *Claudio A. Ardagna, Dipartimento di Informatica, Università Degli Studi di Milano, Milan 20133, Italy,* E-mail: claudio.ardagna@unimi.it

along four main directions: (1) the adoption of the access control-based data governance in Anisetti et al.[10] to enforce data protection requirements at ingestion time, when data are collected and stored, and before data are fed to the analytics pipeline for analysis and processing; (2) the definition and implementation of a data engine architecture that supports the enforcement of the access rules; (3) the integration of the data engine with modern big data analytics pipelines; and (4) the evaluation of the solution on an anomaly detection analytics task.

The data governance approach (point 1) is built upon the utilization of data annotations and the implementation of secure *ad hoc* data transformations, conducted during both the data ingestion and processing phases. The secure transformations are selected according to the privileges of the various tasks/users using the data and produce different data views. In this way, our data engine better balances data protection and usefulness of the data. The result is a platform-independent solution integrated into a novel data engine solution (point 2). The proposed data engine architecture introduces granular data control within a big data environment. It has been implemented using Apache tools and components, such as Apache Spark.

This implementation guarantees the sufficient scalability for processing the substantial volume of real-time data generated by sensors (point 3). We quantitatively evaluated the solution by extending the architecture with a novel state-of-the-art anomaly detection method from sensor network data capable of distributing the workload over multiple computational resources[11] (point 4). In particular, the anomaly detector uses different predictive models trained with different data views retrieved according to the privileges of the requesting user and corresponding secure transformations. Then, we evaluated the predictive performance of the different anomaly detection pipelines and the impact of the transformations on the accuracy of the results.

The remainder of this article is organized as follows. The Motivation and Reference Scenario section describes the motivations and the reference scenario. The Data Governance section describes our access control-based data governance solution. The Data Analytics for Anomaly Detection section describes the anomaly detection analytics pipelines used during the evaluation phase. The Data Engine Implementation section describes the implementation of our data engine. The Experimental Evaluation section presents the experimental evaluation of our approach. Finally, the Related Work section discusses the related work and the Conclusions section gives our concluding remarks.

Motivation and Reference Scenario

We present the motivation (the Motivation section) and the reference scenario (the Reference Scenario section) at the basis of the solution in this article.

Motivation

Big data environments add lots of complexity to data governance and analytics techniques, especially when the need to balance data protection and data sharing arises. Indeed, they are highly dependent on cloud-edge computing, which extensively uses multitenancy. Multitenancy allows sharing one instance of an infrastructure, platform, or application by multiple tenants to optimize costs. This leads to situations where a service provider offers subscription-based analytic capabilities in the cloud, or the same data engine is accessed by multiple customers. As a consequence, a big data pipeline often entails data and services belonging to different organizations, becoming a serious threat to security and privacy.

Traditional data governance solutions are struggling to keep up with technological evolution for data analytics, and their rigid structure poorly copes with a context where the need for flexibility is one of the fundamental requirements. The emerging conflict between accurate and effective data governance enabling data protection, on one side, and effective data analytics for accurate decision-making, on the other side, introduces the need to rethink existing solutions to tune data engine operation to the actual use of the data, as well as to maximize data sharing while providing a suitable level of data protection. New requirements driven by the peculiarities of the big data environment emerged as follows:

Dynamism and context awareness. Data are diverse, constantly evolving, and pose new requirements for flexible data governance. Traditional solutions struggle with changing environments and varying user types. User labeling can enhance effectiveness and flexibility. Making such solutions on user labeling rather than static users might contribute to a more effective and flexible approach.

Ingestion-time enforcement. Applying data governance at data ingestion permits to reach a better compliance with privacy and data protection laws, avoiding conflicts. By managing and preparing data before storage, transformations can be enforced based on user privileges and processing needs, maintaining compliance, and enabling effective data management.

Balancing data protection and data quality. Although data protection and privacy are fundamental requirements when operating in big data environments, they must not clash with the ability to extract as much value as possible from the processed data. An increasing pressure is put on solutions that prioritize the protection of data still preserving a level of data quality and usability.

Analytics pipeline robustness. The quality of the data collected is a crucial aspect of any analytics pipeline. In fact, both supervised and unsupervised machine learning methods aim to generate labeled historical data to predict future events. Such patterns and models can produce inaccurate results if data quality is not maintained, and information is lost at earlier stages of the pipeline. This is particularly true for unsupervised learning tasks where manually generated labels (i.e., ground truth) are not provided.

Data analysis explainability. Data analysis processes must be explainable to each stakeholder. Several applications perform effective analyses without paying sufficient attention to the audience and their ability to understand technical aspects. This is crucial in several applications where the comprehensibility of the responses of the predictive models is fundamental.[12,13]

Balancing operativity and compliance with the General Data Protection Regulation. Nowadays, systems involve many users with different roles and thus different privileges on the sensitive data to inspect. Therefore, there is the need to maximize the amount of sensitive data and predictive models' output each user can access according to its role. No access should happen only in the worst case scenario.

In this article, we present an Apache-based big data engine architecture driven by the above requirements, with a novel data governance approach based on an attribute-based access control (ABAC) system.[14] Our data governance approach provides a data governance solution that manages data sharing between different actors of the analytics pipeline with different privileges. The goal of our approach was to balance, while maximizing, quality and protection of shared data and the corresponding analytics. We integrated the data engine with a smart data analytics solution for anomaly detection and experimentally evaluated our approach in a smart city reference scenario.

Reference scenario

Our reference scenario considers a smart city scenario, where public transportation data in the city of Oslo (i.e., buses and trams) are used for anomaly detection in traffic patterns. Data are collected in real time and aggregated on their position every 5 minutes. Aggregated data are ingested in batches in the data storage. Each batch contains information about the GPS location (latitude and longitude) of the vehicles, the status of the vehicles (malfunction, traffic congestion), and details about the path (origin, destination, stop point, delay).

The transportation data are used by a traffic monitoring application (i.e., anomaly detector) that comes in three configurations, one for each category of users: policeman, air quality control officer, and citizen. Each user has different access rights applying different data transformations since not all users can access the whole data set. For instance, local policy department can use the data to detect specific locations in which intervention is needed, whereas the local air quality department can use the data to detect areas with spikes in traffic, possibly helping citizens avoid congested areas.

Each configuration of the anomaly detector consists of a different machine learning pipeline built on a different machine learning model. All models are derived from the same data set at different levels of granularity due to the applied data transformations. A model trained on "normal," real-time traffic data of 5-minute batches represents the usual traffic behavior and is used to spot any divergences from the expected behavior. Indeed, once collected and ingested, data are prepared for the analysis with traditional data preparation (e.g., cleaning). Then, data are transformed according to a set of applicable access control policy rules that evaluate the intended user and the requested data. Finally, data are used by the anomaly detector.

To this aim, a cloud-based multi-tenant data engine is implemented to support our anomaly detector, and in general third-party smart city applications, to work on the same data sets. The data engine is extended with data governance approach built on the ABAC system. Considering data analytics, our anomaly detector assumes data to be shared for data analysis. This is architecturally and methodologically different from the approach used in federated learning,[15] where data are generated and used locally at multiple nodes to create models, whose parameters are shared while retaining raw data private.

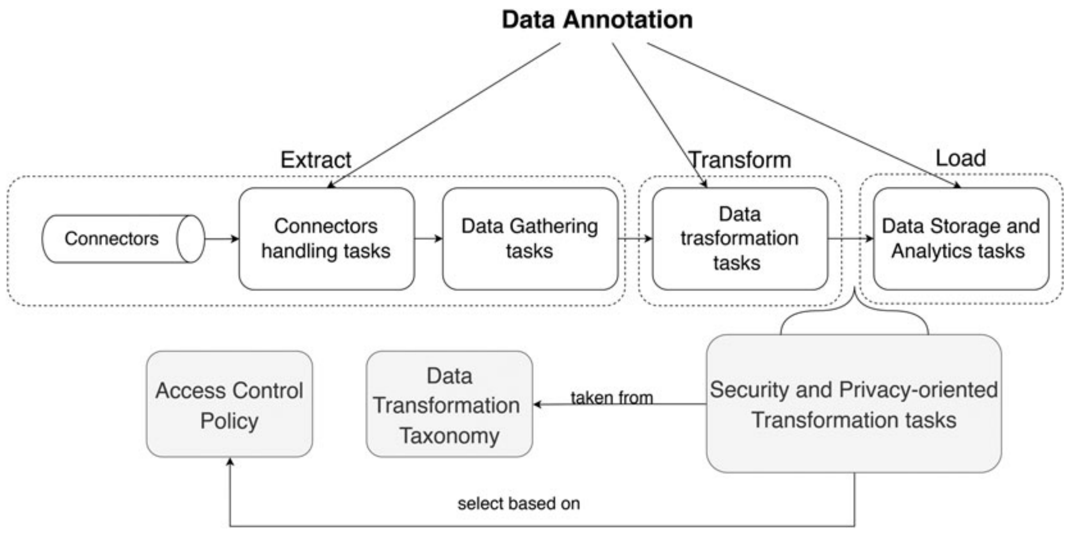
Data Annotation
Extract
Transform
Load
Connectors
Connectors handling tasks
Data Gathering tasks
Data trasformation tasks
Data Storage and Analytics tasks
Access Control Policy
Data Transformation Taxonomy
taken from
Security and Privacy-oriented Transformation tasks
select based on

FIG. 1. Annotation transformations and access control policies applied to our ETL ingestion pipeline. ETL, Extract, Transform, Load.

To guarantee explainability, the proposed trained model can distinguish between normal and anomalous cases by labeling the current data gathered by sensors. The algorithm also provides the "motivations" of an anomaly, explaining the raised alerts to the security operators. The motivations are implemented as a ranking of the variables under analysis in descending order with regard to their importance (e.g., for a bus stuck in traffic jam, the average delay of vehicles within the involved zone will appear as top-ranked, whereas the level of smog in the air will appear as the second-ranked, and so on).

This approach is motivated by the reference scenario where it is requested that the predictive models must explain the urban problems since police operators can promptly understand their type and develop appropriate measures and that must be suitable for the type of problem encountered.

Data Governance

Our policy-aware data ingestion procedure[10] for data governance implements an advanced Extract, Transform, Load (ETL) approach. It extends traditional ETL with data transformations triggered by access control policies and is used to sanitize ingested data. These policies are based on the identity of the user/service using data and the annotations on the ingested data.

Access control-based ETL

Figure 1 shows how a traditional ETL is extended for better data governance. The white boxes in Figure 1 refer to a traditional ETL ingestion procedure in the form of a big data pipeline that includes: (1) *connectors* establishing the connection with the data sources extracting raw data with producer speed; (2) *connectors handling tasks* dealing with the different connectors' peculiarities, for example, raw data format/size and the velocity of data producers; (3) *data gathering tasks* collecting data; (4) *data transformation tasks* transforming data formats; (5) *data storing/forwarding tasks* either saving or forwarding data to the analytics procedures.

Data annotation on top of Figure 1 annotates data at different steps of the pipeline as follows: (1) connection handling tasks, to annotate raw data at gathering time; (2) data transformation tasks, to annotate specific data fields before their transformation; (3) data storing and analytics tasks, to provide annotations on data resulting from the transformation tasks.

The gray boxes in Figure 1 extend the traditional ETL ingestion pipeline (white boxes) to address the security and privacy challenges of multitenancy scenarios with data transformation processes built on access control policies and data annotations. Security and privacy-aware transformations are used, ranging from pruning and reshaping to encrypting/decrypting or anonymizing the full resource or part of it. These transformations are selected based on a taxonomy of data transformations and triggered by access control policies, leading to a more dynamic and flexible ETL ingestion procedure.[†] This approach makes it possible to decouple the data transformations typical of a traditional ingestion procedure from the policy-driven data transformations we propose in this article, which are linked to a specific processing task and the requesting user/service.

Access control policies

The access control language proposed in this article is a refinement of our previous work in Anisetti et al.[10] It extends the ABAC model that offers flexible fine-grained authorization capabilities and exploits XACML (eXtensible Access Control Markup Language)[16] obligations to introduce preemptive data transformations. Indeed, in a collaborative big data scenario, preemptive data transformations are more suitable than denying access to data. For this reason, when a subject makes an access request to a resource, our access control filters the set of results according to the subject's privileges, obtaining data protection by removing or obfuscating sensitive attributes rather than denying access to full data.

We tailor the common key elements of the ABAC model to deal with the peculiarities of a big data scenario. The access control policy expresses access conditions based on key elements and their attributes as follows:

Definition 3.1 (Policy Condition). A *Policy Condition* is a Boolean expression of the form (*attr_name* op *attr_value*), with op$\in\{<, >, =, \neq, \leq, \geq\}$, *attr_name* an attribute label, and *attr_value* the corresponding attribute value.

A policy is then defined as follows:

Definition 3.2 (Policy). A *policy P* is 5-uple $<$*subj, obj, action, env, datatrans*$>$, where:

- Subject *subj* defines a user or the service provider of a job issuing access requests to perform operations on objects. It is of the form $<$*id*, *PC*$>$, where *id* defines a class of users (e.g., policeman), and *PC* is a set of *Policy Conditions* on the subject, as defined in Definition 3.1. For instance, $<$*user*, $\{(\text{age} \geq 18)\}>$ refers to a person of legal age.
- Object *obj* defines any data whose access is governed by the policy. It is of the form $<$*type*, *PC*$>$, where *type* defines the type of object, such as a file (e.g., a video, text file, image), an SQL or noSQL database, a table, a column, a row, or a cell of a table, and *PC* is a set of *Policy Conditions* defined on the object's attributes. For instance, $<$*file*,$\{(\text{creation\_date} < 2020/12/05)\}>$ refers to a file created before 2020/12/05.
- Action *action* defines any operations that can be performed within a big data environment, from traditional atomic operations on databases (e.g., CRUD operations varying depending on the data model) to coarser operations, such as an Apache Spark Direct Acyclic Graph (DAG), a Hadoop MapReduce, an analytics function call, or an analytics pipeline.
- Environment *env* defines a set of conditions on contextual attributes, such as time of the day, location, IP address, risk level, weather condition, holiday/workday, and emergency. It is a set *PC* of *Policy Conditions* as defined in Definition 3.1. For instance, $\{(\text{weather\_conditions} = \text{red\_alert})\}$ refers to a red alert broadcasted for weather conditions.
- Data Transformation *datatrans* defines a set of security and privacy-aware transformations on *obj*, focusing on compliance to regulations and standards, in addition to simple format conversions.

More specifically, the data transformations in the policy include, but are not limited to, suppression or generalization of data, masking and encryption, distortion, and swapping. They are grouped in a taxonomy of functions used to sanitize the ingested data according to the security property they aim to guarantee. We consider the Condidentiality Integrity Availability triad defined by the National Institute of Standards and Technology (NIST)[17] as the reference for our work as follows:

- *Confidentiality* describes the process of hiding information that may lead to personal identification or proprietary information, keeping users anonymous. The conventional security mechanisms used to protect data confidentiality are encryption or hashing.[18–20]
- *Integrity* ensures that data are non-repudiable, authentic, and protected from modifications and deletions. Some of conventional security mechanisms used to ensure data integrity are encryption and signing.[18–20]
- *Availability* ensures that authorized parties should have constant and easy access to information. This includes correctly maintaining hardware,

[†]We note that the transformations triggered by access control policies can be implemented as big data pipelines themselves.

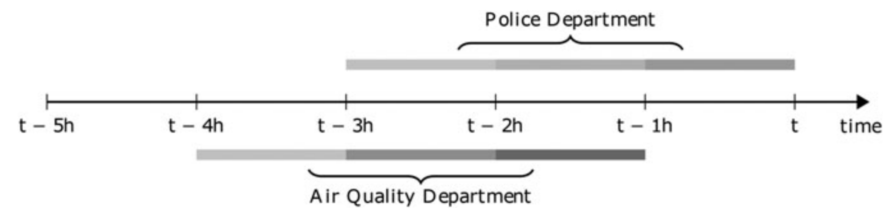

FIG. 2. Example of temporal transformation.

technological infrastructure, and systems that store and show data. Some of conventional security mechanisms used to ensure availability are replication, distribution, high availability, and disaster recovery.

Data transformations can be further classified on the type of information that is sanitized. *Temporal transformations* delay access to data. Data produced at a time T will be accessible at a time $T + \delta$, with δ depending on the policy applied and the desired level of visibility. For example, as shown in Figure 2, a Policeman accesses data when produced (upper segment labeled Police Department), whereas an air quality control officer retrieves them with 1-hour delay (lower segment labeled Air Quality Department).

Spatial transformations alter the granularity of geolocation information, making it more or less granular based on the applied policy and desired level of visibility. For example, in the smart city context, if an unexpected event occurs (e.g., fire, accident), law enforcement could have a more spatially accurate view, whereas citizens could be shown only the macro-area where the event occurred.

Access control policy enforcement

When an access request is submitted, the set of applicable policies is first selected and then enforced on the target object. First of all, the access request is evaluated against the *subj*, *obj*, *action*, and *env* in P. If the conditions *cond* in *subj*, *obj*, and *env* are positively evaluated according to the access request, and the actions in the policy matches the action in the request, the policy is enforced. Policy enforcement consists of the execution of *datatrans* in P against the object target of the access request.

In the following, we present two examples of policies in the reference case study we presented in the Reference Scenario section.

- **Policy 1 (No Transformation).** This policy guarantees unrestricted access to data (no transformation). It is triggered when a *local police department* officer (subject), in an *ordinary* context (env), requests to *read* (action) data tagged as *PII-id* (object). It can be defined as follows:

$$subject = (\text{user}, \{(\text{department} = ''\text{LocalPolice}'')\})$$

$$object = (\text{file}, \{(\text{tag} = ''\text{PII} - \text{id}'')\})$$

$$env = (\text{traffic conditions} = ''\text{Ordinary}'')$$

$$action = \text{read}$$

$$datatrans = \emptyset$$

- **Policy 2 (Temporal Transformation).** It performs a *temporal transformation* granting access only to data older than 24 hours. The policy is triggered when an *Air Quality Department* officer, in an *ordinary* context, requests to *read* data tagged with *PII-id*.

$$subject = (\text{user}, \{(\text{department} = ''\text{AirQuality}'')\})$$

$$object = (\text{file}, \{(\text{tag} = ''\text{PII} - \text{id}'')\})$$

$$env = (\text{traffic conditions} = ''\text{Ordinary}'')$$

$$action = \text{read}$$

$$datatrans = \{TimeShift\_1D\}$$

- **Policy 3 (Spatial Transformation).** It performs a *spatial transformation* aggregating data. The policy is triggered when a *Citizen*, in an *emergency* context, requests to *read* data tagged with *PII-id*.

$$subject = (\text{user}, \{(\text{department} = {''}\text{Citizen}{''})\})$$

$$env = (\text{traffic conditions} = {''}\text{Emergency}{''})$$

$$action = \text{read}$$

$$datatrans = \{LowGranularityAggregation\_1D\}$$

Data Analytics for Anomaly Detection

Anomaly detection is one of the most important tasks within a smart city reference scenario, where several sensors produce data that could be continuously collected and monitored. In this article, we adopt a machine learning method based on deep neural networks to perform anomaly detection in a distributed manner. Our method explains the detected anomalies and works in an unsupervised setting.

Anomaly detection

Anomaly detection is a machine learning task that aims to learn a predictive function by exploiting the historical data representing normal cases. The ability to distinguish normal and anomalous cases in data is crucial to create good predictive models.

In this article, we build a resilient model based on Spark-Growing Hierarchical Self-Organizing Maps (GHSOM)[11] that is robust to noisy data representing (usually) few anomalies within the training set. Furthermore, Spark-GHSOM[11] is a state-of-the-art distributed method for learning predictive models. It is mainly designed for multi-target regression and forecasting problems. We integrate Spark-GHSOM into the full architecture to perform explainable anomaly detection, where explainable predictions are provided to the final users. The process for training the predictive models of the Spark-GHSOM follows the classical process of the GHSOM training[21,22] and is enhanced to work properly also for mixed heterogeneous attributes.[23] It also exploits the Apache Spark framework via the Map-Reduce paradigm for efficient and distributed learning.

The first step in the GHSOM algorithm computes the inherent dissimilarity in the input data with different types of attributes. For numerical attributes, the method exploits the *mean quantization error*, whereas for categorical attributes, the *unlikability* represents a good measure to estimate the difference among values.[24] The method uses such dissimilarity functions to construct SOMs, where a single SOM represents a neural network composed of a grid of neurons, usually at the starting point of a 2×2 map of neurons, which can be enlarged, if needed, at the training phase.

During the training phase, a procedure modifies each neuron that is not sufficiently descriptive of its surrounding training examples to enhance the quality of the current model.[11] Therefore, such a neuron is substituted by another SOM in a hierarchical structure of SOMs (Fig. 3), and the training proceeds recursively.

After the training stage, the set of neurons within the GHSOM are used to identify, for each new unseen instance, its closest neuron within the hierarchy of SOMs.

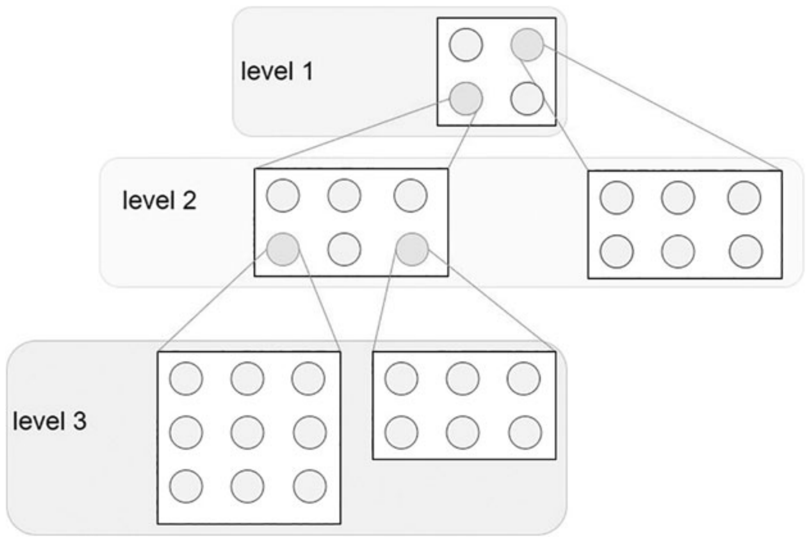

FIG. 3. A Growing Hierarchical Self-Organizing Map.

Such a neuron represents the model part that best fits and represents the input instance. If such an instance is relatively close to the identified neuron, then the input instance is a normal case; otherwise, the input instance is considered an anomaly.

Formally, let x be the new instance to be considered and n one of the possible neurons within the GHSOM, and let $n(x) = \arg\min_n dist(x, n)$ the closest neuron to x. The instance is considered an anomaly if $dist(x, n(x)) > (\mu + t\sigma)$, where μ represents the average distance of the training instances and the neurons of the model, σ represents the standard deviation of such distances, and t represents the user-defined threshold factor parameter.

Explainability

Neural networks such as SOMs are good predictive models but with low explainability, often characterized by the impossibility of understanding the motivations of their output. In several real applications, the need of explainability is high. For instance, in a smart city scenario, a security operator should be able to understand why an anomaly is raised by a system to act accordingly.

Our extended version of Spark-GHSOM is capable of explaining the detected anomalies. Indeed, a more informative approach is considered combining the Boolean response (normal/anomaly) with a ranking of the features describing the detected anomalous phenomenon. Therefore, it is possible to produce a feature ranking according to their importance, indicating their contribution to the detection of the potential anomaly. More specifically, a feature ranking produces a ranking, according to the feature importance measure, of the whole set of features describing the whole set of historical instances considered for training the predictive model, whereas the feature importance is quantified through a number between 0 and 1 indicating how anomalous the value expressed by the feature is with respect to the data collection (the total sum up to 1).

The importance score is determined starting from the Euclidean distance between the current test instance under analysis and its closest neuron within the GHSOM model. More formally, the method computes a ranking with regard to the contribution provided by the single feature in the distance between x and $n(x)$. The ranking function for the instance x is computed as $r(x) = \frac{(x[i] - n(x)[i])^2}{\sum_j (x[j] - n(x)[j])^2}$, where i, j represent feature indexes.

This approach indicates the prominent feature(s) to describe the raised alert explaining its reason. For example, in urban scenarios, for a specific area and time, an event could be described through the level of traffic, the concentration of pedestrians, the level of air pollution, the local temperature, the road visibility level, and so forth. Supposing that a detected anomaly comes with an indication that the traffic level and air pollution level are unusually high in and around a specific location, the operator can gain insights into the nature of the anomaly, which in this example could be associated with a fire in a building facing the street.

Data Engine Implementation

Figure 4 shows the architectural view of our data engine enforcing access control before data are used. The data engine consists of an ecosystem of data management tools and components that support the management of the entire data life cycle, from ingestion to

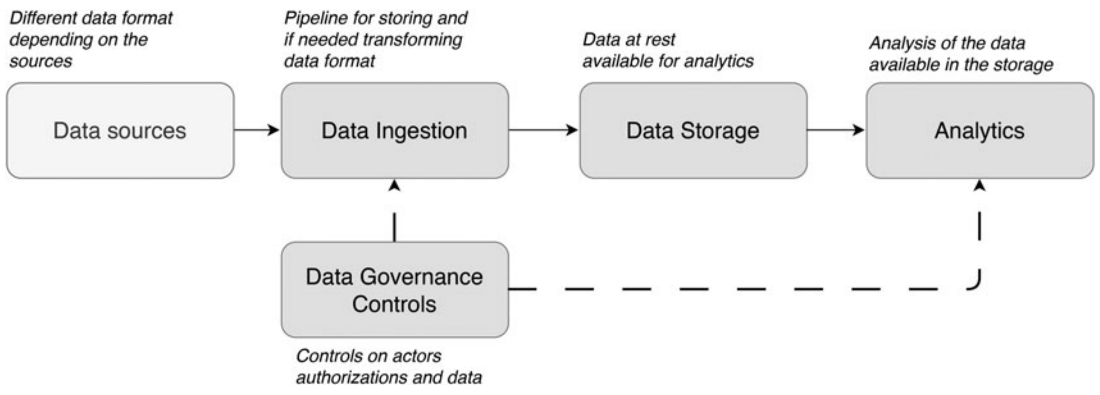
Different data format depending on the sources
Pipeline for storing and if needed transforming data format
Data at rest available for analytics
Analysis of the data available in the storage
Data sources
Data Ingestion
Data Storage
Analytics
Data Governance Controls
Controls on actors authorizations and data

FIG. 4. Architectural view of the data engine.

analytics. It defines the whole data management process, that is, the process of collecting, storing, using, and maintaining data in a secure, efficient, and cost-effective way. It includes all the practices devoted to the protection of data, as well as all the activities needed to prepare properly data for an analytics process.

The data engine acts as a middleware between *data sources*, that is, physical data collection points, and *analytics*, that is, the pipeline that implements a specific analysis on the data available in the data storage to extract value, forming a complete big data engine.

The solid black arrows in Figure 4 show the standard flow of data from data sources to analytics. First, the data are collected from a variety of data sources, each of which has a possible different format. The data are then forwarded to the ingestion pipeline that specifies the activities needed to transform and reconcile the different data formats. Finally, the ingested data are stored in a data store for further analytics activities. The dashed arrows highlight that the whole data management process is subjected to data governance controls, where *ad hoc* controls are implemented to monitor actors' activities, data exchanges, and operations. In particular, access control enforcement is executed both during data ingestion and before any data processing task to manage access to data before its storage and manipulation.

We implemented the data engine in Figure 4 using services and components of the Apache-based ecosystem, integrating the data governance approach in the Data Governance section and the data analysis approach in the Data Analytics for Anomaly Detection section within a complete Apache-based Big Data engine. Figure 5 highlights all the building blocks of the Big Data Engine, their interactions, and the technologies on which they are based, as follows:

Data ingestion. The data ingestion procedure involves storing data in the data storage, which may require format transformations. It is implemented as a pipeline with tasks for data collection, transformation (if necessary), and saving to the storage. Data ingestion supports batch and streaming modes using our data governance approach based on access control.

In batch ingestion, data are collected either through the source API or manual upload, with options for active (event-based) or passive (time-based) data collection. *Apache NiFi*, *Hive*, *Trino*, and *HDFS* are used for batch ingestion.

For stream ingestion, *Apache Kafka* and *Druid* are used. Druid enables real-time transformations on data streams and works together with Kafka for ingestion-based access control enforcement on stream data.

Data storage. The data storage component is responsible for storing data while ensuring confidentiality, integrity, and availability. It provides data to analytics and includes features for data availability, fault tolerance, and reliable data distribution across cluster nodes. The data storage indirectly participates in the processing procedure as nodes typically process local data.

All operations, such as read and write, in the data storage are mediated by our data governance approach. The data storage is implemented using components, such as *Hive*, *Trino*, and *HDFS*.

Data governance. The framework of controls, including processes, policies, standards, and metrics, ensures efficient and secure data access. It implements access control policies during data ingestion and analytics using the ETL mechanism. The key components Ranger and Atlas enable effective data access controls and support the data governance framework. Ranger provides a centralized platform for defining and enforcing access

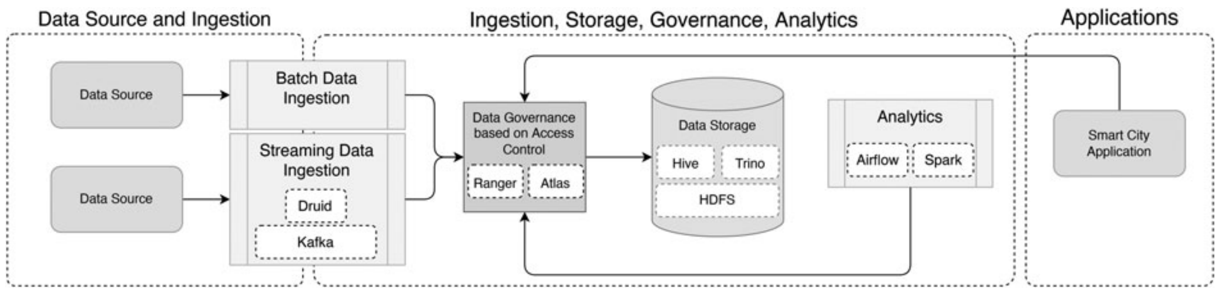

FIG. 5. Apache-based big data engine.

policies on resources and tags, whereas Atlas implements annotations and data lineage. Policies in Ranger mediate access to resources, allowing for data governance flexibility. The result of a policy execution is a data transformation that addresses specific needs emerging in the considered scenario. We note that data transformations refer to Hive transformations. Hive, in fact, provides several manipulation functions that the user can extend to define her own user-defined functions (UDFs).

All functions, those provided by the system and UDFs, can be used as custom masking option in Ranger.

Data analytics. The set of components responsible for implementing the data analytics pipeline built on the data available on storage. It implements the big data pipelines analyzing data in the data storage and relies on components *Spark* and *Airflow* as follows:

- *Spark* is an advanced unified analytics engine for large-scale data processing. The core idea of Spark is to provide an expressive computing system (not limited to the Hadoop MapReduce model) by exploiting in-memory processing of cached data to avoid saving intermediate results to disk and caching data for repetitive queries. Apache Spark framework enables four different programming languages for map-reduce programming (i.e., Java, Scala, Python, R). We considered Scala mainly thanks to the greater documentation support and community.
- *Airflow* is an orchestrator of processing pipelines that aims to provide scheduling and monitoring capabilities. Airflow pipelines are implemented in Python for better dynamic pipeline generation. The pipeline is modeled as a DAG of tasks that can be Spark jobs.

We recall that each access to data by big data pipelines is mediated by our data governance approach.

Experimental Evaluation

We experimentally evaluated our approach in the smart city reference scenario presented in the Reference Scenario section. In the following, we first present the considered data set and our experimental settings (the Experimental Settings section); then, we present a concrete execution of our data governance approach in terms of access control policy enforcement based on spatial transformations (the Data Transformation: Aggregation and Filtering section); we further explain the training of five anomaly detection models on five different spatial views of the considered data set and the results of their execution at inference time (the Data Analytics: Model Training and Inference section); finally, we discuss how policy enforcement affects the privacy and quality of the overall analytics (the Discussion section). We note that the five different detection models represent the five different configurations of our anomaly detector and, in turn, the multitenancy proper of our reference scenario.

Experimental settings

We tested our methodology in a smart city scenario using a data set taken from open data on local public transport in Oslo. The data set counts 9.88 M rows and 25 columns (attributes). Each row contains information about the location of a single bus/tram and other information, such as origin, destination, delay, and malfunctioning. All attributes in the data set are detailed in Table 1.

An anomaly detector was implemented to spot abnormal situations, that is, any situation diverging from the *normal* behavior. The goal of the detector is to retrieve early indicators of critical events, such as accidents, unauthorized protests, or terrorist attacks.

In the above settings, we considered that the traffic anomaly detector could be used by three categories of users (i.e., policeman, air quality control officer, citizen) under two different environment conditions (i.e., *normal*, *emergency*), leading to five different access control policies. These policies mediate access to data by means of spatial transformations that limit the view of the detector on the data.

Our experiments were run on a single node virtual machine running Ubuntu 20.04.2 LTS, installing the big data engine in the Data Engine Implementation section. The virtual machine was equipped with Intel Xeon E5 2620- 2.095GHZ CPU and 64 GB of RAM.

Data transformation: aggregation and filtering

The data transformation process consists of a preparation and a filtering phase, working as follows:

Aggregation. Data aggregation is a fundamental step for anomaly detection on time series and properly defines the concept of *time-unit-anomaly*. Our experiments represented and collected data on movable points (buses and trams) from a "fixed point of view" (i.e., a specific area of the city) and aggregated them in a 5-minute time window representing our time-unit-anomaly.

Table 1. Attributes and related descriptions of the data set used in the experiments

Attribute	*Description*
DateTime	Timestamp specifying when the position/status was recorded/updated
LinkDistance	Distance in meters between the previous stop (or current, if located at stop) and the next stop
Percentage	How much of the total distance (percentage) that has been traversed at the time of the message
LineRef	Reference to the line in question
DirectionRef	Reference to the direction in question
PublishedLineName	Name describing the line in question
OriginRef	Reference to the origin in question
OriginName	Name describing the origin of the departure
DestinationRef	Reference to the destination in question
DestinationName	Name describing the destination of the departure
OriginAimedDepartureTime	Origin aimed departure time
DestinationAimedArrivalTime	Destination aimed arrival time
VehicleRef	Reference to the vehicle in question
Delay	Delay time, defined as "PT0S" (0 seconds) when there are no delays
HeadwayService	Field defining whether the service is a headway service
InCongestion	Field defining whether the traffic is in congestion or other circumstances, which may lead to further delays
InPanic	Boolean field, indicating that the driver reported an "out of order" status
GPS (Latitude and Longitude)	Position of the public vehicle, according to the timestamp recorded
StopPointRef	Reference to the stop point in question
VisitNumber	Number of the stop point in question
StopPointName	Name of the stop point in question
VehicleAtStop	Field defining whether the vehicle is at the stop
DestinationDisplay	Name of the next destination

Table 2. Aggregated data set

Attribute	*Description*
Date	The timestamp of the current measurement/instance
ClusterLatitude	The latitude of the cluster centroid from where aggregate
ClusterLongitude	The longitude of the cluster centroid from where aggregate
Delay	Average of the buses
Percentage	Average total distance that has been traversed
InPanic	Average buses in panic mode
InCongestion	Average buses in congestion
DestinationAimedArrivalTime	Average destination aimed departure time
OriginAimedDepartureTime	Average origin aimed departure time
HeadwayService_False	Count of instances for non-headway services
Anomaly	0,1 indicating if the instance represents a normal case 0 or an anomaly 1 (for quantitative evaluation purpose only)

Starting from the vehicles traffic data set in Table 1, data were prepared for analytics according to a time aggregation executed at ingestion time and producing the data set in Table 2. Collected data were aggregated in 5-minute time windows considering data coming from n distinct locations. $k < n$ was then selected, where k represents the number of spatial areas (spatial clusters) for spatial aggregation. Each spatial area is represented in terms of its cluster centroid, that is, a set of GPS (latitude and longitude) coordinates. Obviously, changing the value of k also changes the number of zones and centroids considered, and of the level of spatial precision: a larger k gives larger granularity and thus more precise information, whereas a smaller k leads to smaller granularity and thus less precise information. Our experiments used five different predictive models based on *k-means* trained according to five different aggregated data sets with $k \in \{5, 10, 25, 50, 100\}$.

Properly modeling the spatial dimensions of the data considering spatial autocorrelation phenomena[25–27] emphasizes possible agreements or discrepancies related to the different sensors' measurements located in different positions. Therefore, according to the considered clustering model, we collected different aggregated data for training the final predictive models.

Filtering. We exploited the values of k to define the access control policies performing spatial filtering on data according to the role (i.e., policeman, air quality control officer, citizen) and the conditions of the environment (*normal*, *emergency*). Intuitively, a user with more permissions can use a larger k, whether a user with fewer privileges should use a smaller k. Filtering on the aggregated and tagged data was performed at ingestion time by changing k as specified in the following policies.

Policy 1: ⟨Citizen, TrafficData, Read, Normal, AggregationClusters = 5⟩, with *Citizen* ≡ ⟨user,{(department = "Citizen")}⟩ and *TrafficData* ≡ ⟨file,{(tag = "traffic")}⟩ This policy specifies that *citizens* can access low-accurate data in a *normal* situation. Data quantization is very low (five centroids); thus, the retrieved data show only approximately the anomalous area.

Policy 2: ⟨Citizen, TrafficData, Read, Emergency, AggregationClusters = 10⟩, where *CitizenUser* ≡ ⟨user, {(department = "Citizen")}⟩ and *TrafficData* ≡ ⟨file, {(tag = "traffic")}⟩ With this policy, during an *emergency*, *citizens* can access medium-low-accurate data. Data quantization is low (10 centroids). Retrieved data allow to see the anomalous area.

Policy 3: ⟨AirQualityUser, TrafficData, Read, Normal, AggregationClusters = 25⟩, where *AirQualityUser* ≡ ⟨user,{(department = "AirQuality")}⟩ and *TrafficData* ≡ ⟨file,{(tag = "traffic")}⟩ In this case, the policy states that *air quality control officers* can access medium-accurate data in both *normal* and *emergency* situations. Data quantization is medium (25 centroids). Retrieved data are still useful to outline the zone where an anomaly occurred.

Policy 4: ⟨Policeman, TrafficData, Read, Normal, AggregationClusters = 50⟩, with *Policeman* ≡ ⟨user, {(department = "Police")}⟩ and *TrafficData* ≡ ⟨file,{(tag = "traffic")}⟩ The policy states that *policemen* can access accurate data in a *normal* situation. Data quantization is high (50 centroids).

Policy 5: ⟨Policeman, TrafficData, Read, Emergency, AggregationClusters = 100⟩, where *Policeman* ≡ ⟨user, {(department = "Police")}⟩ and *TrafficData* ≡ ⟨file, {(tag = "traffic")}⟩ During an *emergency*, *policemen* can access the most accurate data. Data quantization is very high (100 centroids) resulting in a more accurate model guaranteeing a more precise identification of the anomaly location.

Policy enforcement led to the generation of the five different data sets in Figure 6.

Data analytics: model training and inference

Our experiments used five time series data collected from March 1, 2022, at 17:40 to October 18, 2022, at 14:45. The training set included time series from March 1, 2022, at 17:40 to the end of September 2022; the test set included the remaining 18 days of October. The testing time series were randomly perturbed by considering for each test instance 100 times the real value of average delay, average in panic, and the average in congestion buses. 1% of the testing instances were randomly selected and perturbed. The output files of the experiments are available online.[‡]

Our training process differs from traditional processes where a single model is trained on the original data set, and the inference results are aggregated and filtered according to users' privileges. It rather builds on five different models trained on the five different data sets generated at ingestion time, which natively address the access control requirements. Each model (M1–M5 in Table 3) corresponds to the enforcement of a policy (Policy 1–Policy 5 in the Data Transformation: Aggregation and Filtering section). In particular, our analytics pipeline performs a training phase on the five subset of data obtained after spatial filtering (representing normal situations) and generates five predictive models.

At this point, each time a specific user decides to perform monitoring through the anomaly detector, it uses the model corresponding to his/her role. In this way, the model natively addresses data protection because it is built with data aggregated according to the access control policies based on the user's privileges and only responses that can be accessed by the user requesting the analytics are used. In other words, the system presents as much sensitive data as possible and the detected anomalies at the level of detail granted by the user's privileges, thus balancing operativity and General Data Protection Regulation (GDPR) compliance.

We note that we automatically configured the threshold factor t in the Anomaly Detection section by using the trained GHSOM for predictions on the training set to achieve a given percentage *fp%* of false positives, that is, zero false alarms. This is coherent with the assumption that the considered training set represents the set of historical normal cases.[11] Although this approach solves the problem of manually defining the appropriate value for t, it still requires defining *fp%*, which is, anyway, much easier to define by the end-user than t. In our experiments, the best *fp%* value is identified by maximizing the f1-score in an internal cross-validation for $fp\% \in \{0, 0.01, 0.1\}$.

Table 3 shows the results in terms of weighted precision, recall, and f1-score with regard to the support of the classes normal and anomaly. f1-score is particularly suitable for the imbalanced task at hand due to the rare anomalous items and abundant normal cases.

We considered the different predictive models M_1–M_5 generated according to the number of data clusters (5, 10, 25, 50, 100, respectively) imposed by Policy 1–Policy 5 transformations in the Data Analytics: Model Training and Inference section. The results in Table 3 show that the number of clusters considered has a very low impact on the predictive performance (precision between 0.981 and 0.998, recall between 0.983 and 0.997), whereas it has a high impact on the amount of retrieved information. M_1 and M_2 being

[‡]Output files are available at www.di.uniba.it/~mignone/materials/MBDAaaS/anomdet.7z

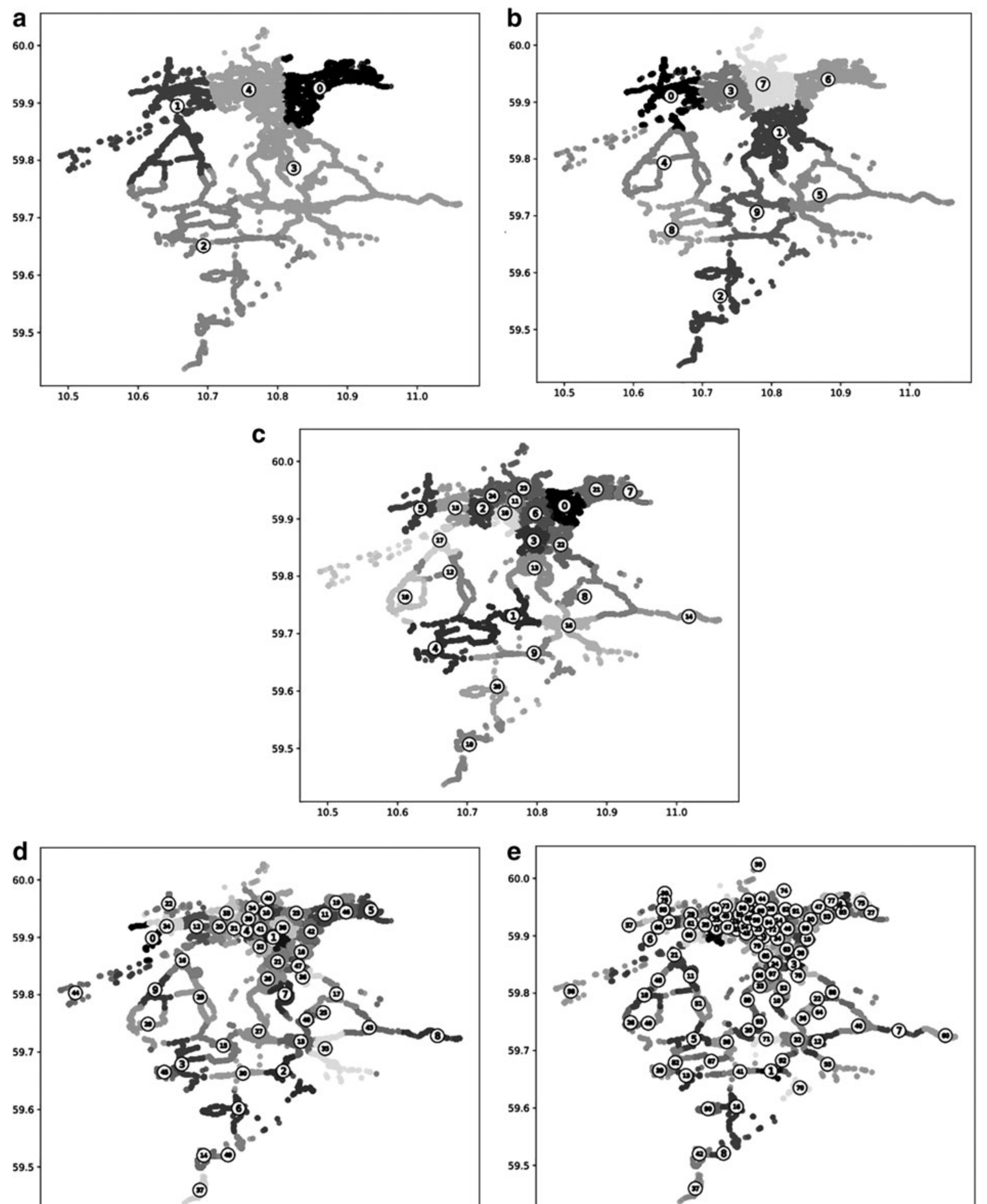

FIG. 6. Results of the enforcement of policies in the Data Transformation: Aggregation and Filtering section. Spatial transformation (number *k* of clusters) depends on the user role (*user*) and the status of the environment (*env*). **(a)** Policy 1 (user: Citizen, env: Normal, *k*: 5). **(b)** Policy 2 (user: Citizen, env: Emergency, *k*: 10). **(c)** Policy 3 (user: Air Quality, env: Any, *k*: 25). **(d)** Policy 4 (user: Policeman, env: Normal, *k*: 50). **(e)** Policy 5 (user: Policeman, env: Emergency, *k*: 100).

Table 3. Results in terms of precision, recall, and f1-score

Model	*Precision*	*Recall*	*f1-score*	*fp%*	*Train time*	*Data card*
M_1 (5_clus)	0.995	0.993	0.994	0	140	198312
M_2 (10_clus)	0.998	0.997	0.997	0	362	399677
M_3 (25_clus)	0.990	0.990	0.986	0	1997	910724
M_4 (50_clus)	0.981	0.983	0.982	0.1	1488	1577192
M_5 (100_clus)	0.982	0.990	0.985	0.01	7888	2546608

We also report the training time (in seconds), the training set cardinality, as well as the value for *fp*% that maximizes the f1-score.

built on 5 and 10 clusters, respectively, provide a rough idea of the area affected by an anomaly (e.g., north-south-east-west-downtown); M_3 being built on 25 clusters provides a more detailed idea of the area affected by an anomaly (e.g., district); M_4 and M_5 being built on 50 and 100 clusters, respectively, provide a very detailed idea of the area affected by an anomaly (e.g., street, precise location).

Finally, we observed the results of the ablation analysis aimed to identify the best value for *fp%* and, therefore, of *t*. The experiments showed that, the more the clusters, the data sensitivity, and the data volume, the higher the value of *fp%* that maximizes the f1-score. Indeed, with $k=5$, $k=10$, and $k=25$, the best results are obtained with a threshold *t* that is identified by assuming no false positives on the training set. On the contrary, with $k=50$ and $k=100$, it is necessary to admit a small percentage of false positives on the training set to avoid too conservative models that do not detect abnormal cases properly. This behavior was somehow expected since a larger number of clusters require the algorithm to consider cluster boundaries that are not clearly identifiable.

Discussion

Our main research objective focused on how to achieve a better balance between protection and quality in cloud-based and multi-tenant big data pipelines. That is, we aimed to ensure that the sensitive data used in the pipeline were protected even in case they are processed by users with different privileges, still producing high-quality results. Our key findings are as follows:

Full Privacy Scenario. It considers models M1 and M2 to achieve a high degree of anonymization at the expense of a substantial loss of quality in the data. What is important to note is that aggregation, performed before the training phase, actually changes the work and results of anomaly detection. In the case study, the system monitors areas of different sizes for each level of aggregation (Fig. 6a, b). The citizen's view, for example, reduces the sensitivity of the model as small anomalies have less impact in a larger area. This behavior prevents the citizens from detecting anomalies he/she should not have access to and which are not of interest to him/her anyway.

Full Quality Scenario. It considers models M5 and M6 to greatly reduce anonymization, while supporting a higher quality of data. As already discussed, aggregation carried out upstream changes the behavior of anomaly detection. In this case, the model is much more sensitive, making it possible to detect even point anomalies of greater interest to a security officer. We note that it is possible to take into account other signals (e.g., an emergency situation) to alter the granularity of the system, making it more flexible and adaptive.

Full Post-Aggregation Scenario. It considers the common approach with a single prediction model and a posteriori aggregation to anonymize the results. The model underlying such a system must be as granular as possible, avoiding the risk of removing information that is fundamental for the most privileged users. On the contrary, it is necessary to find an aggregation heuristic that is not a simple average over larger or smaller aggregation windows, so as to avoid an anomaly detected at higher granularities being visible also in those at lower granularities. This approach could also breach the rules imposed by the GDPR: although not all data are visible to the final end users, during the analytics they are managed by possibly unauthorized services or users.

Related Work

A growing number of companies and organizations use big data analytics tools to identify business opportunities and drive decision-making. Thus, the number of data security concerns has recently grown. It is commonly recognized that securing big data platforms takes a mix of traditional security tools, newly developed toolsets, and intelligent processes for managing and monitoring security throughout the entire data life cycle,[28,29] and this has consequently led to a new set of data security and privacy requirements.[7,30,31]

Most of the current solutions are based on ABAC[14] for its ability to define attribute-based policy rules, with attributes presented at run-time, coming from multiple sources. However, they suffer from limitations that range from focusing only on a subset of the requirements we highlighted in the Motivation and Reference

Scenario section, or adding delay in updating and enforcing the security policies due to the increased complexity and computational overhead. Moreover, they are often tailored either to a very specific scenario or to a particular technology.

Some recent proposals, such as Chen et al.[9] or Gupta et al.,[32,33] propose a solution working only with the Apache Hadoop stack, leveraging on the native access control features of the platform, that still presents some limitations, such as the complexity of deployment and the consumption of resources.

There are also database-oriented approaches, focusing on a particular database, hence on a particular type of analytics pipeline. For example, Chabin et al.[34] presents an access control system for graph-based models, whereas Gupta et al.[35] and Huang et al.[8] deal with NoSQL databases, such as MongoDB and HBase. All database-oriented approaches are based on query rewriting mechanisms to avoid leakage of sensitive resources at the cost of high computational complexity and low efficiency in enforcing security policies. Scenario-specific approaches often focus on specific big data scenarios, such as the case of federated cloud and edge Microservices[36] or Internet of Things use cases.[37] The challenge mainly addressed by these works is the management of access control decisions in a scenario with multiple stakeholders in continuously evolving federations.

Finding unusual patterns or events in data that change over time and space, such as urban traffic patterns, is a common problem faced in various fields. These anomalies, such as traffic congestion or large crowds, can be difficult to detect due to their rarity and the fact that their definition varies based on location and time. Current methods for identifying anomalies in this type of data often fail to consider the relationships between different points in space and time and the specific characteristics of the anomalies themselves.[38]

For instance, in Zhang et al.,[39] the authors proposed a decomposing approach to detect anomalies of different types, such as abnormal pedestrian flows or traffic accidents with varying locations and times. Specifically, they distinguish between the normal component, that is, urban dynamics decided by spatiotemporal features, and the abnormal component that is caused by anomalous events. In Riveiro et al.,[40] the authors proposed a framework that provides support for the exploration of multidimensional road traffic data through visual analytics. The anomaly detection method is based on a traditional SOM used for clustering. The number of clusters is optimized through Silhouette cluster analysis. This approach is inspired by previous work presented in Kraiman et al.,[41] which used a traditional approach for anomaly detection based on clustering algorithms, SOMs, and Gaussian mixture models. However, these fail to consider the relationships between different points in space and time and the specific characteristics of the anomalies themselves.

In Zhang et al.,[42] the authors proposed a survey on research in urban anomaly analytics discussing various types of anomalies in the analyzed contexts, such as urban anomalies, traffic anomalies, unexpected crowds, environment anomalies, and individual anomalies. Furthermore, the authors emphasized that one of the open challenges is posed by urban data that have full of noise for precise predictions. The consequence is that anomalous events can easily be less represented than noise. The anomaly detection method considered in this article is aimed at resolving the open problems presented in Zhang et al.[42] by taking into account additional information coming from the spatial and temporal information. Such additional information allows the system to exploit spatial proximity information or temporal recurrence/periodicity in identifying truly anomalous cases.

Conclusions

We presented a novel data engine solution based on advanced access control-based ETL, supporting fine-grained data management and protection. We detailed how we implemented it within a complete Apache-based Big Data engine. Adding access control during the data extraction process offers several benefits with respect to traditional solutions segregating users within the data lake and building different extraction processes. First of all, it enhances data governance by providing more control over data, reducing the risk of unauthorized access or misuse of information, and protecting sensitive data from unauthorized disclosure. Moreover, it enables data customization, allowing users to have personalized views or subsets of data relevant to their needs, improving decision-making.

We evaluated the impact in terms of the performance of our solution in a case study in the smart city domain. Specifically, we considered the task of identifying anomalies in geo-referenced and time-stamped data, continuously produced by sensors. The experimental evaluation shows that the identified anomalies, built on data manipulated according to access control policies, provide different perspectives of the anomalies, also at different levels of granularity, revealing only relevant information for the specific role/user. For future work, we plan to extend the study to other machine learning tasks for data generated by sensors, other than anomaly detection, such as time series classification and regression.

Author Disclosure Statement

No competing financial interests exist.

Funding Information

This work was partly supported by (1) project IMPETUS, funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 883286; (2) project MUSA—Multilayered Urban Sustainability Action—project, funded by the European Union—NextGenerationEU under the National Recovery and Resilience Plan (NRRP) Mission 4 Component 2 Investment Line 1.5: Strengthening of research structures and creation of R&D "innovation ecosystems," set up of "territorial leaders in R&D" (CUP G43 C22001370007, Code ECS00000037); (3) the project SERICS (PE00000014), funded by the EU—NextGeneration EU under the NRRP MUR program.

References

1. Qureshi SR, Gupta A. Towards efficient Big Data and data analytics: A review. In: 2014 Conference on IT in Business, Industry and Government (CSIBIG); 2014; pp. 1–6.
2. Samarati P. Protecting respondents' identities in microdata release. IEEE Trans Knowledge Data Eng 2001;13(6):1010–1027.
3. Machanavajjhala A, Gehrke J, Kifer D, et al. L-diversity: Privacy beyond k-anonymity. In: 22nd International Conference on Data Engineering (ICDE'06); 2006; pp. 24–4.
4. Dwork C, McSherry F, Nissim K, et al. Calibrating noise to sensitivity in private data analysis. J Priv Confidentiality 2017;7(3):17–51.
5. Basanta-Val P, Sánchez-Fernández L. Big-BOE: Fusing Spanish official gazette with big data technology. Big Data 2018;6(2):124–138.
6. Basanta-Val P, Fernández-García N, Sánchez-Fernández L, et al. Patterns for distributed real-time stream processing. IEEE Trans Parallel Distrib Syst 2017;28(11):3243–3257.
7. Colombo P, Ferrari E. Access control technologies for Big Data management systems: Literature review and future trends. Cybersecurity 2019; 2(1):1–13.
8. Huang L, Zhu Y, Wang X, et al. An attribute-based fine-grained access control mechanism for HBase. In: Database and Expert Systems Applications. (Hartmann S, Küng J, Chakravarthy S, et al. eds) Springer International Publishing: Cham; 2019; pp. 44–59.
9. Chen C, Elsayed MA, Zulkernine M. HBD-authority: Streaming access control model for Hadoop. In: 2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys); 2020; pp. 16–25.
10. Anisetti M, Ardagna CA, Braghin C, et al. Dynamic and scalable enforcement of access control policies for big data. In: Proceedings of the 13th International Conference on Management of Digital EcoSystems. MEDES'21. Association for Computing Machinery: New York, NY, USA; 2021; pp. 71–78. Available from: https://doi.org/10.1145/3444757.3485107
11. Malondkar A, Corizzo R, Kiringa I, et al. Spark-GHSOM: Growing Hierarchical Self-Organizing Map for large scale mixed attribute datasets. Inform Sci 2019;496:572–591.
12. Ding W, Abdel-Basset M, Hawash H, et al. Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. Inform Sci 2022.
13. Tjoa E, Guan C. A survey on explainable artificial intelligence (XAI): Toward medical XAI. IEEE Trans Neural Netw Learn Syst 2020;32(11):4793–4813.
14. Hu VC, Ferraiolo D, Kuhn R, et al. Guide to Attribute Based Access Control (ABAC) Definition and Considerations. NIST Special Publication; 2014.
15. Yang Q, Liu Y, Chen T, et al. Federated machine learning: Concept and applications. ACM Trans Intell Syst Technol 2019;10(2):1–19.
16. OASIS committee (Organization for the Advancement of Structured Information Standards). eXtensible Access Control Markup Language (XACML). (Rissanen E. ed.) OASIS Open; 2013. Available from: www.oasis-open.org/committees/xacml/ [Last accessed: January 8, 2024].
17. Nieles M, Dempsey K, Pillitteri V. An Introduction to Information Security. NIST Special Publication; 2017.
18. Goyal V, Pandey O, Sahai A, et al. Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. CCS'06. Association for Computing Machinery; New York, NY, USA; 2006; pp. 89–98. Available from: https://doi.org/10.1145/1180405.1180418
19. Boyen X, Waters B. Anonymous hierarchical identity-based encryption (without random oracles). In: Advances in Cryptology—CRYPTO 2006. (Dwork C. ed.) Springer Berlin Heidelberg: Berlin, Heidelberg; 2006; pp. 290–307.
20. Armknecht F, Boyd C, Carr C, et al. A guide to fully homomorphic encryption. IACR Cryptology ePrint Archive; 2015.
21. Kohonen T. Self-organized formation of topologically correct feature maps. Biol Cybern 1982;43(1):59–69.
22. Kohonen T, Honkela T. Kohonen Network. Scholarpedia; 2007.
23. Hsu CC. Generalizing self-organizing map for categorical data. IEEE Trans Neural Netw 2006;17(2):294–304.
24. Kader GD, Perry M. Variability for categorical variables. J Stat Educ 2007; 15(2).
25. Corizzo R, Ceci M, Pio G, et al. Spatially-aware autoencoders for detecting contextual anomalies in geo-distributed data. In: Discovery Science: 24th International Conference, DS 2021, Halifax, NS, Canada, October 11–13, 2021, Proceedings. Berlin, Heidelberg: Springer-Verlag; 2021; pp. 461–471. Available from: https://doi.org/10.1007/978-3-030-88942-5_36
26. Mignone P, Malerba D, Ceci M. Anomaly detection for public transport and air pollution analysis. In: 2022 IEEE International Conference on Big Data (Big Data); 2022; pp. 2867–2874.
27. Ceci M, Corizzo R, Japkowicz N, et al. ECHAD: Embedding-based change detection from multivariate time series in smart grids. IEEE Access 2020; 8:156053–156066.
28. Al-Badi A, Tarhini A, Khan AI. Exploring big data governance frameworks. Proc Comput Sci 2018;141:271–277.
29. Ben Aissa MM, Sfaxi L, Robbana R. DECIDE: A new decisional big data methodology for a better data governance. In: Information Systems. (Themistocleous M, Papadaki M, Kamal MM. ed.) Springer International Publishing: Cham; 2020; pp. 63–78.
30. Anisetti M, Ardagna CA, Berto F. An assurance process for Big Data trustworthiness. Future Gener Comput Syst 2023;146:34–46. Available from: https://www.sciencedirect.com/science/article/pii/S0167739X23001371
31. Lv Z, Song H, Basanta-Val P, et al. Next-generation Big Data analytics: State of the art, challenges, and future research topics. IEEE Trans Industr Inform 2017;13(4):1891–1899.
32. Gupta M, Patwa F, Sandhu R. An attribute-based access control model for secure Big Data processing in Hadoop ecosystem. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control. ABAC'18. New York, NY, USA: Association for Computing Machinery; 2018; pp. 13–24. Available from: https://doi.org/10.1145/3180457.3180463
33. Gupta M, Patwa F, Sandhu R. Object-tagged RBAC model for the Hadoop ecosystem. In: Data and Applications Security and Privacy XXXI. (Livraga G, Zhu S. ed.) Springer International Publishing: Cham; 2017; pp. 63–81.

34. Chabin J, Ciferri CDA, Halfeld-Ferrari M, et al. Role-based access control on graph databases. In: SOFSEM 2021: Theory and Practice of Computer Science: 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25–29, 2021, Proceedings. Berlin, Heidelberg: Springer-Verlag; 2021; pp. 519–534. Available from: https://doi.org/10.1007/978-3-030-67731-2_38
35. Gupta E, Sural S, Vaidya J, et al. Enabling attribute-based access control in NoSQL databases. IEEE Trans Emerg Top Comput 2022;11(1):208–223.
36. Preuveneers D, Joosen W. Towards multi-party policy-based access control in federations of cloud and edge microservices. In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW); 2019; pp. 29–38.
37. de Matos E, Tiburski RT, Amaral LA, et al. Providing context-aware security for IoT environments through context sharing feature. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE); 2018; pp. 1711–1715.
38. Sofuoglu SE, Aviyente S. GLOSS: Tensor-based anomaly detection in spatiotemporal urban traffic data. Signal Processing. 2022. Available from: https://www.sciencedirect.com/science/article/pii/S0165168421004072 [Last accessed: January 8, 2024].
39. Zhang M, Li T, Shi H, et al. A decomposition approach for urban anomaly detection across spatiotemporal data. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19. International Joint Conferences on Artificial Intelligence Organization; 2019; pp. 6043–6049. Available from: https://doi.org/10.24963/ijcai.2019/837
40. Riveiro M, Lebram M, Elmer M. Anomaly Detection for Road Traffic: A Visual Analytics Framework. IEEE Trans Intell Transp Syst 2017;18(8): 2260–2270.
41. Kraiman JB, Arouh SL, Webb ML. Automated anomaly detection processor. In: Enabling Technologies for Simulation Science VI. vol. 4716. (Sisti AF, Trevisani DA. eds.) International Society for Optics and Photonics. SPIE; 2002; pp. 128–137. Available from: https://doi.org/10.1117/12.474940
42. Zhang M, Li T, Yu Y, et al. Urban anomaly analytics: Description, detection, and prediction. IEEE Trans Big Data 2022;8(3):809–826.

Cite this article as: Polimeno A, Mignone P, Braghin C, Anisetti M, Ceci M, Malerba D, Ardagna CA (2024) Balancing protection and quality in big data analytics pipelines. *Big Data* 3:X, 1–17, DOI: 10.1089/big.2023.0065.

Abbreviations Used

ABAC	= attribute-based access control
DAG	= Direct Acyclic Graph
ETL	= Extract, Transform, Load
GDPR	= General Data Protection Regulation
GHSOM	= Growing Hierarchical Self-Organizing Maps
UDFs	= user-defined functions