Blockchain analytics and artificial intelligence

D. N. Dillenberger
P. Novotny
Q. Zhang
P. Jayachandran
H. Gupta
S. Hans
D. Verma
S. Chakraborty
J. J. Thomas
M. M. Walli
R. Vaculin
K. Sarpatwar

Blockchain records track information about financial payments, movements of products through supply chains, identity verification information, and many other assets. Analytics on this data can provide provenance histories, predictive planning, fraud identification, and regulatory compliance. In this paper, we describe analytics engines connected to blockchains to provide easy-to-use configurable dashboards, predictive models, provenance histories, and compliance checking. We also describe how blockchain data can be combined with external data sources for secure and private analytics, enable artificial intelligence (AI) model creation over geographically dispersed data, and create a history of model creation enabling provenance and lineage tracking for trusted AI.

1 Introduction

Enterprises are using blockchains, such as the Hyperledger Fabric [1, 2], to share information that provides insights into food supply chains [3], facilitates trades [4], expedites cross-border money payments [5], provides identity verification [6], secures diamond grading reports [7], and tracks shipments around the world [8]. These are just a few production blockchains in use.

The data on these enterprise blockchains are associated with an identity of the blockchain participant, timestamped, integrated, and vetted into business processes. This provides a rich warehouse of information for analytics and artificial intelligence (AI). Currently, data scientists spend 80% of their time collecting, preparing, cleaning, and organizing data for analysis [9]. On an enterprise blockchain, the data have already been identified, collected, prepared in a common format, and organized.

In the following sections, we describe how to visualize data on the blockchain (Section 3), build predictive models (Section 4), and create provenance histories (Section 5). We built a rules engine that enables users to specify conditions, which our analytics engine then uses to check the blockchain for compliance (Section 6). When blockchain data have to be combined with external data sources, we produced a set of libraries that enable data to reside where it originated, and instead our Fusion Manager combines training weights from local models to create AI models that keep data private but achieves an accuracy that is greater than just local model training (Section 7). These prior sections describe how analytics can be used to obtain insight from blockchain. We also include a section that describes how blockchain can help AI (Section 8). Currently, data scientists do not have historical, immutable records on what data were used to train AI models. Many AI models are downloaded from public websites, obtained from third parties. The models may have been trained with biased data or poisoned data that would cause the models to create predictions that are detrimental to the company using the models [10, 11].

2 Related Work

2.1 Analytics on blockchain data

Blockchain data analytics and management is an emerging area in which only a few studies have looked at the issues concerning analytics of blockchain data. The BLOCKBENCH system [12] benchmarks the popular blockchain implementations—Fabric [1], Ethereum [13], and Parity [14]—against a set of database workloads exhibiting different characteristics [IO-heavy/CPU-heavy workloads, workloads with minimal operations, online transaction processing (OLTP) workloads, etc.]. The BLOCKBENCH study concluded that current blockchain systems are not well suited for large-scale data processing workloads. Some other studies [15, 16] have also looked at performance studies of public blockchain systems, such as Bitcoin [17]. In this paper, we look at efficiently processing temporal queries on Fabric—an aspect not investigated previously. Efficient handling of temporal queries has been

Digital Object Identifier: 10.1147/JRD.2019.2900638

a very well-researched topic in the data management community [18–20].

An alternative paradigm known as "off-chain analytics" has also been investigated, wherein blockchain data are taken out, stored, and analyzed using a database. For example, Coinalytics [21] uses MemSql to analyze the blockchain data. The Blockparser system [22] generates the detailed dump of Bitcoin data in CSV format, which can be easily ingested in a database. Ron and Shamir [23] store the Bitcoin history and associated transaction graph in a database to analyze user behaviors (e.g., how users acquire and spend bitcoins). The off-chain analytics allow for efficient processing of the data, but introduce privacy problems and an additional overhead of transferring the data from the blockchain platform to an external database. In our analytical methods, we leverage both on-chain and off-chain approaches depending on the efficiency and operational characteristics required.

2.2 Leveraging blockchain for data sharing and AI

Over the last few years, there has been tremendous interest in startups targeting data and AI marketplaces with different objectives and business models. Many of them highlight their use of blockchain for decentralization but provide limited technical details.

Tracking the provenance of data and how it is shared is a central focus of Datum [24]. TraneAI [25] incentivizes users to help with tagging and labeling of data using tokens on blockchain. The Ocean protocol [26] allows data consumers and data owners to enter into contracts on blockchain, as well as supports data discovery. Users can also pledge stake on datasets that they believe will become important and that they are helping improve the quality of. As datasets get used, users are rewarded based on their stake. Computable Labs [27] focuses on crowdsourcing and incentivizing the curation of datasets using blockchain. OpenMined [28] has the privacy of data as its central focus and leverages federated learning and differential privacy. Federated training is supported using techniques such as secure multiparty computation and homomorphic encryption, whereas the owners retain private control of the data.

Machine learning and AI algorithms are executed within smart contracts in Algorithmia [29], Neureal [30], and TraneAI [25]. In Algorithmia [29], users submit a dataset and a fee for anyone who can build the model with the highest accuracy on the dataset. Models submitted by other users are tested using a smart contract, and the fee is released to the winning model. Neureal [30] provides a platform for running AI predictions on live data and leverages blockchain to incentivize accurate predictions.

3 Secure analytics service and configurable dashboard

Blockchain solutions [1–8] record immutable, curated, timestamped, agreed-upon, cross-organizational data. For typical blockchain solutions, there is a need to answer business questions that frequently translate to aggregate queries and well-defined descriptive analytics, such as trends, time series, aggregations, spatio-temporal analytics, and top-N queries. Enterprise blockchain platforms usually provide access control and a security model with capabilities, such as user and organization identity, permissioning, data access control, and often various privacy features, such as channels or private data collections in Hyperledger Fabric. Blockchain business analytics capabilities, as described before, need to follow the same access control and security model as the underlying blockchain platform. Furthermore, depending on the application, it may be undesirable to extract the blockchain data and analyze it externally since blockchain applications may have stringent extract, transform, and load (ETL) policies because of security, performance (near-real-time analytics), or replication concerns.

In this section, we describe a system that we have developed for blockchain solutions that leverage Hyperledger Fabric [1]. Our objective is to initially provide descriptive analytics service that has the following properties:

1) tightly integrated with the Hyperledger Fabric platform (no need to ETL, same security model, etc.);
2) solution-independent (i.e., can be used with any blockchain solution);
3) follows the same data abstractions (key-value store);
4) easy and fast to configure and use.

Blockchain data in Hyperledger Fabric reside in two places. The main storage is in the replicated blockchain *ledger,* which is basically an immutable store of *transactions.* A transaction comprises the following: a *header,* which captures its relevant metadata; *signature,* cryptographic signature of the client; *proposal,* specific inputs to the transaction chaincode; *response,* which contains *read-sets and write-sets* of the transaction chaincode to capture the world state before and after the transaction; and *endorsements,* obtained from different endorsing peers. Transactions are typically grouped to form *blocks* that are finally stored on the ledger.[1]

While the ledger provides basic data retrieval capabilities, its design is geared toward properties, such as immutability, privacy, and security, as support for analytics

[1] For complete details on the ledger structure, we refer the reader to https://hyperledger-fabric.readthedocs.io/en/release-1.3/ledger/ledger.html.

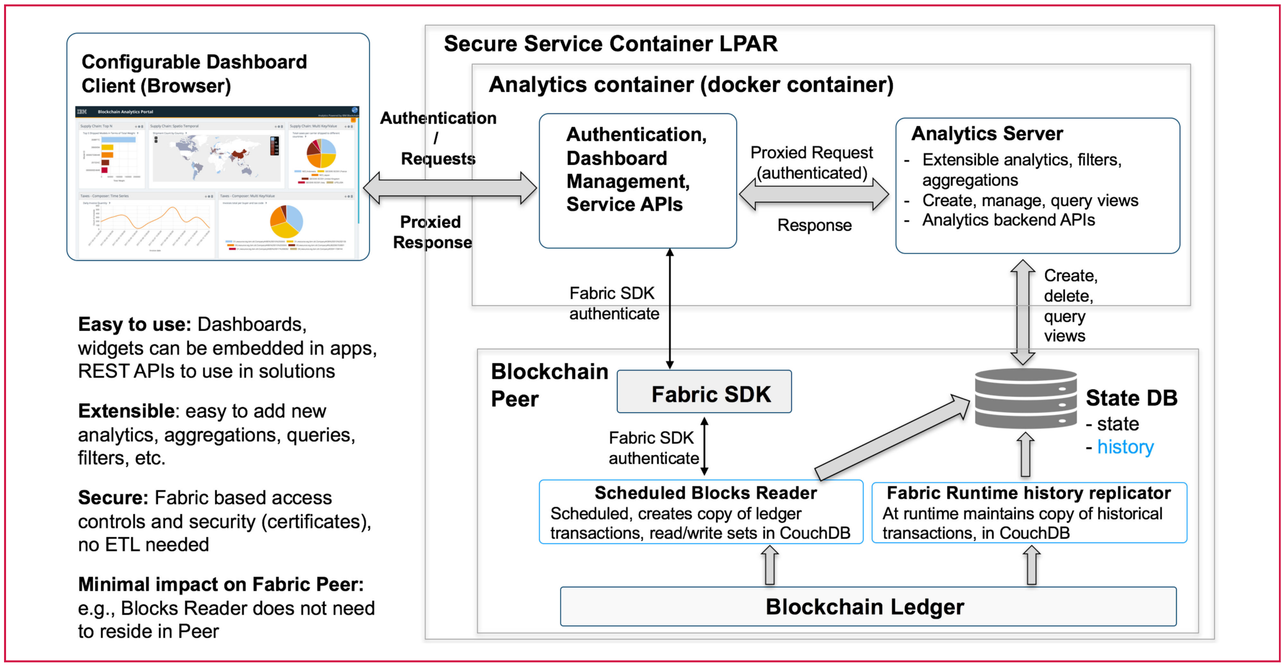

Configurable Dashboard Client (Browser)
Authentication / Requests
Proxied Response
Secure Service Container LPAR
Analytics container (docker container)
Authentication, Dashboard Management, Service APIs
Proxied Request (authenticated)
Response
Analytics Server
- Extensible analytics, filters, aggregations
- Create, manage, query views
- Analytics backend APIs
Fabric SDK authenticate
Create, delete, query views
Easy to use: Dashboards, widgets can be embedded in apps, REST APIs to use in solutions
Extensible: easy to add new analytics, aggregations, queries, filters, etc.
Secure: Fabric based access controls and security (certificates), no ETL needed
Minimal impact on Fabric Peer: e.g., Blocks Reader does not need to reside in Peer
Blockchain Peer
Fabric SDK
State DB
- state
- history
Fabric SDK authenticate
Scheduled Blocks Reader
Scheduled, creates copy of ledger transactions, read/write sets in CouchDB
Fabric Runtime history replicator
At runtime maintains copy of historical transactions, in CouchDB
Blockchain Ledger

Figure 1

Architecture of secure analytics service and configurable dashboard.

queries and workloads is not the primary design goal. Consequently, current ledger implementations in most blockchain platforms, including Hyperledger Fabric, are not well suited for analytics queries [12]. To partially address this limitation, Hyperledger Fabric also stores the latest snapshot of the data, i.e., the latest value of each key, in the secondary database, called *state database*. Hyperledger Fabric's modular architecture allows for a variety of database implementations to be used as a state database; the current release supports LevelDB [31] and CouchDB [32]. CouchDB supports storage and retrieval of JSON data and provides richer query capability that resonates with solution designers. For these two reasons, and since it is fully supported by Hyperledger Fabric, we chose CouchDB as our initial target database implementation for the analytics service. Therefore, the requirement for the analytics capability is that the data in the key-value pairs are recorded in the JSON format. As will be seen, this allowed us to take advantage the CouchDB query capabilities.

The schema of documents stored in the state database depends on the application at hand. For instance, taking a toy as an example, consider a scenario with single document type "Marble" that has attributes, such as *name, color,* and *owner.* The state database would store the current attributes of different marbles assets, say, with keys "marble1" and "marble2," as follows: {*key:* "marble1," *value:*{*"color":* "blue," *"owner*": "abc"}} and {*key:* "marble2," *value:*{*"color":* "green," "*owner*": "bca"}}.

The schema for historical data would contain different read/write sets that capture the transitions applied to the world state. For instance, to capture the transaction transferring ownership of "marble1" from owner "abc" to "cba," the historical database contains a document: {*reads*:[*key: "marble1," value*: {"*owner*": "abc"}], *writes:*:*[key:* "marble1,*" value: {"owner": "*cba*"}]}.*

3.1 Design, architecture, and implementation

Figure 1 shows an architecture overview of the analytics service. In this figure, we assume that the analytics service is codeployed with one blockchain peer running on an IBM Blockchain Platform [33]. However, the service can be deployed with more than one peer if needed; it does not rely on IBM Blockchain Platform, and it can be deployed with the peer running in any environment supported by Hyperledger Fabric.

As shown in Figure 1, our analytics system includes three main components: the *authentication and dashboard management server,* the *analytics server,* and the *ledger blocks reader/parser.* The system is exposed to external users through the *configurable dashboard web client,* which allows users to visualize the blockchain data and create and manage dynamic descriptive analytics, such as time-series aggregations, spatio-temporal, and top-N analytics.

The web client connects to an *authentication and dashboard management server*. This component registers and authenticates users with their blockchain network

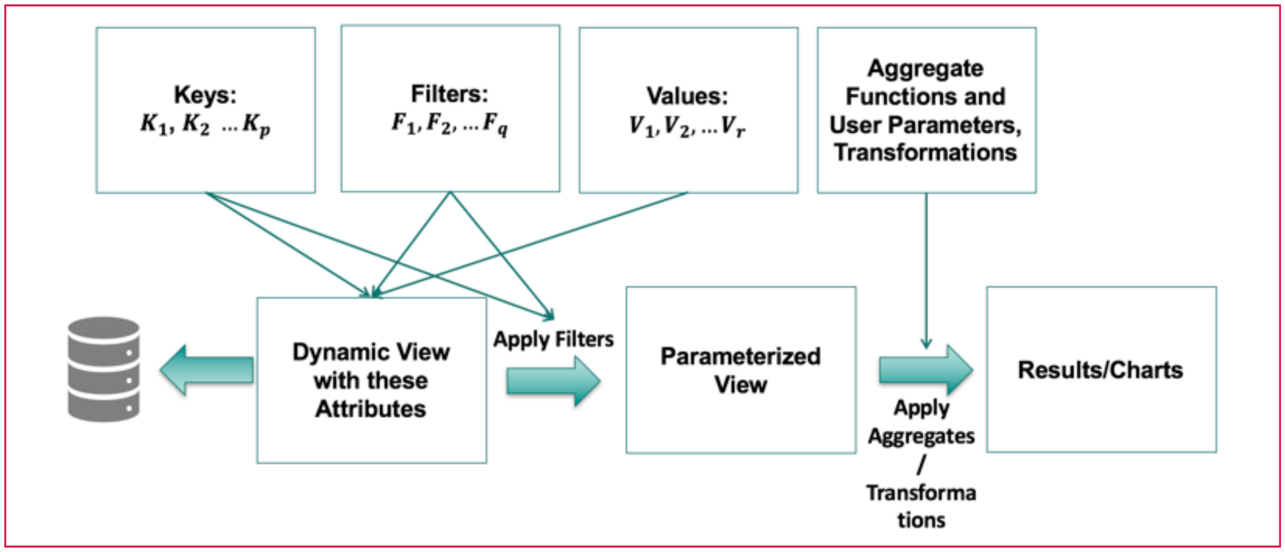

Keys: $K_1, K_2 \dots K_p$
Filters: $F_1, F_2, \dots F_q$
Values: $V_1, V_2, \dots V_r$
Aggregate Functions and User Parameters, Transformations
Dynamic View with these Attributes
Apply Filters
Parameterized View
Apply Aggregates / Transformations
Results/Charts

Figure 2

Flow of the query creation process.

credentials (private key, certificates, connection profile, blockchain organizational and channel details, etc.) using Hyperledger Fabric SDK APIs, and upon successful authentication, it provides dashboard management capabilities to support form-based interfaces for query specification.

The actual analytics requests are passed on to the *analytics server,* which is responsible for the processing of the queries against blockchain data and performing all analytics operations, such as aggregations or filtering. Additionally, the component provides functions to introspect and infer the data schema of blockchain data assets. This makes the analytics server independent of blockchain solution, as it is schema-agnostic.

The analytics service needs to be able to efficiently query both the state database and the complete ledger historical data. As explained before, in the current implementations of the ledger, complex and efficient queries are not supported. The *ledger blocks reader/parser* component addresses this problem by maintaining a copy of historical ledger data in the same CouchDB physical database where the state database resides. This allows us to leverage the query capabilities of CouchDB, such as the creation of indexes and views, which are significantly more powerful than the current ledger query capabilities. We provide two implementations of this component: *Scheduled Blocks Reader* is designed to run in an "offline fashion" as a scheduled job that queries the ledger regularly using the Fabric SDK, reads any new transactions that have been added to the ledger, decodes the read/write sets of the transactions, creates new corresponding JSON documents with the obtained key-value pairs of the data, and stores these documents in the CouchDB database. Alternatively, *Fabric Runtime History Replicator* can be used, which we built as a direct extension of Hyperledger Fabric system chaincode. This code gets activated each time a validated transaction is committed to the ledger, and similar to the offline version, the read/write sets of the transaction are added to the corresponding new JSON document, which is stored to the CouchDB database. Fundamentally, these two alternative implementations provide the same functionality, but they differ in how they can be used. The advantage of the runtime variant is that no additional query of the ledger is necessary to obtain transactions. Furthermore, the ledger and the created database are always consistent with each other. On the other hand, the scheduled variant is noninvasive of the peer's compute cycles and can exist completely independent of it.

3.2 Query creation flow

Figure 2 shows four main components of the query creation process, namely *keys, filters, values and transformations,* and *aggregators.* We now describe these steps, guided by the following simple descriptive query as an example:

"Compute the average tonnage of cargo shipped from various countries to Mexico, in the past six months."

Key attributes: In specifying the query, keys represent the "group by" attributes that allow users to partition the data into multiple segments on which aggregations can be applied. For instance, in the above-mentioned query "source country of shipment" is a key attribute; thus, before the data are processed, it is segmented based on the country of origin.

Value attributes: These represent the attributes on which the aggregation is performed. For example, the above-mentioned query has “tonnage of shipment” as a value attribute.

Filter attributes filter out data based on user-specified criteria. In the above-mentioned example, there are two filter attributes: First, the destination country name must be Mexico, and second, shipment date should be within the past six months. The data points that do not satisfy these requirements will be filtered out.

Aggregators/transformations: Aggregators are functions that are applied on top of different segments of the curated data. In the above-mentioned query, the aggregator is the function computing *average of the tonnage values.* Alternatively, an aggregator could be count, sum total, percentages, median, standard deviation, etc.

Views: The key performance-related challenge in the query process is to parse the CouchDB JSON documents and retrieve the key and value attributes, as specified before. We use CouchDB views to address this problem. Given a set of attributes, we generically construct a mapping function that retrieves the requested attributes from documents dynamically as they are added or updated, and stores the corresponding information as an “inverted table,” or a *view*. When a new query is created, we first check if a matching view exists, i.e., a view whose attributes are a superset of the query attributes. If such a view exists, it is used for efficient retrieval. If not, a new view is created.

Depending on the particular solution, the transaction volume and the corresponding peer load can vary significantly (from a small number of transactions per hour for applications, such as asset management to hundreds of transactions per second for applications such as stock exchanges). In the case of blockchain solutions with high-performance requirements (hundreds of transactions per second), the deployment of an analytics engine on one of the endorsing peers could create possible performance bottlenecks or may be impossible because of the high computational loads. In such cases, Hyperledger Fabric allows for adding extra peers, sometimes called *nonendorsing peers,* which maintain ledger and state without the overhead of participating in the process of transactions endorsement. We can add a dedicated nonendorsing peer (“analytic peer”) to the deployment, which is guaranteed to have the full copy the ledger. This reduces the impact of the analytics engine on the existing blockchain peers. Alternatively, in secure environments such as the IBM Blockchain Platform, the analytics peer can be substituted by the “secure containers” that reside at the same level of security as the blockchain solution itself, i.e., the same logical partition.

4 Predictive models with IBM Watson Studio

Data held in a blockchain can be leveraged for advanced analytics. Predictive models can be built, given there are sufficient historic data held in a blockchain. In this section, we present an example of applying machine learning (ML) to blockchain data. The use case revolves around analytics on supply chain data held on the IBM Blockchain Platform and building an ML model that predicts potential delays in shipping. The essential data science elements of exploratory analysis, feature engineering, building, training, and operationalizing the ML model are achieved using IBM’s data science tool IBM Watson Studio (previously called Data Science Experience) [34].

As a primary data source in this use case, we use the supply chain data held on a blockchain network hosted in the IBM Blockchain Platform. The supply chain information is enhanced with weather data, obtained through APIs provided by The Weather Company [35]. We also bring in location data to further enhance the supply chain data.

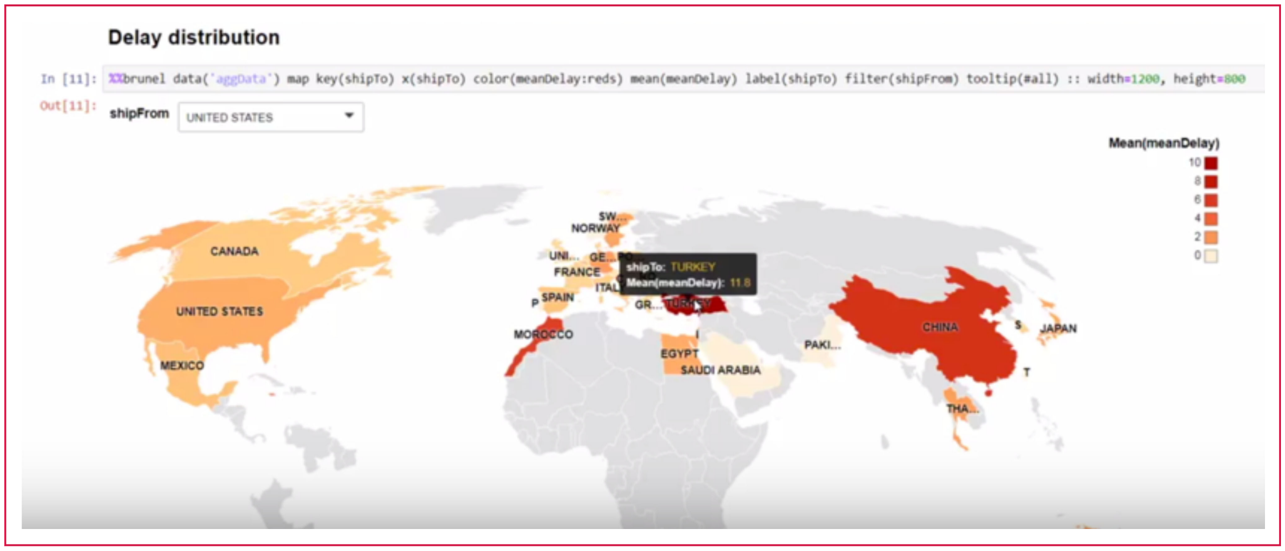
Watson Studio uses various security features for installation, data handling, and end-user access. The tool can be installed using a private SSH key file. An HTTPS connection to the client can be established using an SSL certificate and private key. Watson Studio generates a bearer token when a user signs in and securely stores information in the user’s home directory. This token expires based on defined policies. When a user signs out, the stored bearer token is cleared. An Administrator sets up authentication and authorization policies for users. Remote datasets do not create a local copy of the data. Security for local copies is ensured through encryption of the storage partition.

The first stage for most Data Science projects is the exploratory analysis of the data—looking at the data in different ways to understand which aspects of the data may be most helpful in terms of predicting an outcome, in this case, predicting shipping delays. Watson Studio allows us to connect to the Blockchain environment using a Spark [36] connector. Once connected, we pull data related to shipments from the Blockchain state database. This connection returns JSON data, a hierarchical structure. We work with this data in Spark dataframes, consisting of records and features, allowing for easy operations on the data. We join the different data sources into one dataframe that contains all the shipping, weather, and location data.

We then aggregate and summarize this data for visualization. Watson Studio supports a variety of Python- or Scala-based visualization techniques. We used the Brunel visualization library [37] to generate interactive visualizations that allow us to explore various patterns in the data.

The visualizations yielded a number of high-level observations. For example, a basic visualization shows that when we ship from the United States, the highest mean shipping delays occur when shipping to Turkey, as shown in **Figure 3**.

Another visualization showed us the impact of weather on U.S.-only shipments, where they originated from specific states. **Figure 4** summarizes delivery delays and weather conditions at the destination on the scheduled


Delay distribution

```
In [11]: %%brunel data('aggData') map key(shipTo) x(shipTo) color(meanDelay:reds) mean(meanDelay) label(shipTo) filter(shipFrom) tooltip(#all) :: width=1200, height=800
Out[11]:
```

shipFrom: UNITED STATES

shipTo: TURKEY
Mean(meanDelay): 11.8

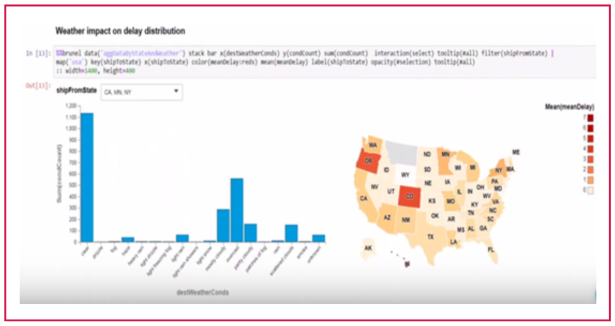
Figure 3

Visualization of mean shipping delays from the United States.

delivery date. We can drill into the interactive visualizations to explore further. For example, when we click on "Overcast," the visualization changes to show that Texas experiences a mean delivery delay of 3.14 days on overcast days. Such observations help guide further investigation by the supply chain experts.

The next stage is to use what we learned during the exploratory analysis stage to help build an ML model for predicting delays. We already have the data that we need to train the model. We may need to prepare the data further into a format suitable for model training. This step, often referred to as feature engineering, involves deciding which attributes to select as features, checking their impact on the predictive power of the model, improving these features, etc. Watson Studio provides various capabilities for feature engineering. As we are trying to predict shipment delay, a continuous variable, we chose to build a Regression [38] model. We selected "Delay" as the label to predict, and we selected our collection of features for the training data. Watson Studio allows us to experiment with a number of Regression algorithms (Random Forest, Decision Tree, Linear, etc., as examples). The tool allows us to evaluate and compare the results of these algorithms and select the best performing model we want to use, as shown in **Figure 5**.

Once we have selected the best model, we can deploy it for production use. Watson Studio allows a deployment pipeline that takes the model and any related assets through testing, staging, and production steps. We want the model to be invoked in real time as part of a shipping status inquiry. The trained and deployed ML model presents a REST [39] scoring endpoint. The model can be called by making a programmatic REST API call to this endpoint, with the relevant features as payload. In our use case, a status inquiry website was built. The website makes a REST call to the deployed ML model and informs the user whether the shipment is on time or is delayed and what the expected delay will be, as shown in **Figure 6**.

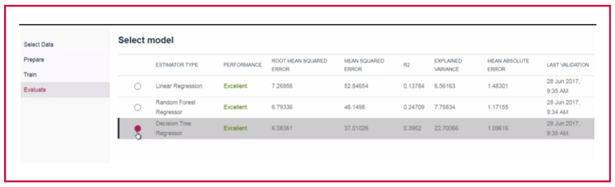

Weather impact on delay distribution

```
In [13]: %%brunel data('aggDataByStateAndWeather') stack bar x(destWeatherConds) y(condCount) sum(condCount)  interaction(select) tooltip(#all) filter(shipFromState) |
map('usa') key(shipToState) x(shipToState) color(meanDelay:reds) mean(meanDelay) label(shipToState) opacity(#selection) tooltip(#all)
:: width=1400, height=400
Out[13]:
```

shipFromState: CA, MN, NY

Figure 4

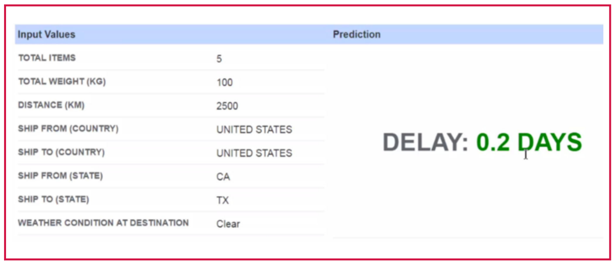
Visualization of weather impact on delays.


Select model

	ESTIMATOR TYPE	PERFORMANCE	ROOT MEAN SQUARED ERROR	MEAN SQUARED ERROR	R2	EXPLAINED VARIANCE	MEAN ABSOLUTE ERROR	LAST VALIDATION
○	Linear Regression	Excellent	7.26956	52.84654	0.13784	6.56163	1.48301	28 Jun 2017, 9:35 AM
○	Random Forest Regressor	Excellent	6.79336	46.1498	0.24709	7.75834	1.17155	28 Jun 2017, 9:34 AM
●	Decision Tree Regressor	Excellent	6.08361	37.01026	0.3962	22.70066	1.09616	28 Jun 2017, 9:35 AM

Figure 5

Select the best performing model.



Input Values		Prediction
TOTAL ITEMS	5	DELAY: 0.2 DAYS
TOTAL WEIGHT (KG)	100	
DISTANCE (KM)	2500	
SHIP FROM (COUNTRY)	UNITED STATES	
SHIP TO (COUNTRY)	UNITED STATES	
SHIP FROM (STATE)	CA	
SHIP TO (STATE)	TX	
WEATHER CONDITION AT DESTINATION	Clear	

Figure 6

Website makes a REST call to deployed ML model and displays an expected delay if any.

An important consideration in operationalizing an ML model is retraining the model. ML models tend to degrade in performance over time, as patterns in the data change. Watson Studio provides capabilities to monitor the performance of the deployed model, sets alerts when performance falls below thresholds we set, and allows us to retrain the model as needed.

5 Provenance queries

In this section, we discuss how we can efficiently execute provenance queries on Hyperledger Fabric. Fabric keeps track of all transactions executed and the associated details, e.g., new assets created, asset states modified, etc. These data can be queried to answer provenance queries related to the life-cycle of an asset. These provenance queries can support various use-cases (e.g., compliance, visualization, and reporting). Provenance queries frequently involve retrieving states of an asset in a given time range. However, retrieving such states is inefficient and cumbersome for two reasons. First, the historical transaction data are not housed in a database, but rather are distributed across a set of time-ordered blocks on file-system, with each block containing the details of a set of transactions and a link to the previous block. Second, Fabric does not maintain any temporal indexes on the blocks, and Fabric needs to access many irrelevant blocks to retrieve asset states of interest.

Notation: Let us assume that each transaction on Fabric ingests one or more key-value pairs describing a business event each. The notations are described in the following list.

$[k, (v, t)]$	key-value pair describing a business event where k represents the asset, v represents the new state of the asset k, and t represents the event time
$E(k, \theta)$	the set of pairs (events), where the key is equal to k and t lies in time interval θ
$B(k)$	the set of Fabric blocks containing details of transactions ingesting a pair with key k
$B(k, t)$	the set of Fabric blocks that contain the details of a transaction that was executed before or at timestamp t and ingested a pair with key k

Fabric provides an API GetHistoryForKey(k) (GHFK), which accesses the contents of blocks in a set B(k) and returns an iterator over the states of asset k. Fabric maintains an index, which stores the set B(k) for each key k, as well as makes use of this index to execute the GHFK call. If we want to retrieve asset-states in time interval [$t1$, $t2$], we can stop accessing the iterator once we encounter an event with a timestamp greater than $t2$. This hence requires accessing blocks in set B(k, $t2$). Note that blocks in set B(k, $t1$-1) are not relevant for the query but are still accessed. We next discuss two models that simulate temporal indexes and mitigate this limitation.

For each asset k, model INT collects events E(k, θ) for suitably chosen intervals θ and ingests new key-value pairs of the form [(k, θ), E(k, θ)], where (k, θ) is the key part and E(k, θ) is the value part. We call such intervals as indexing intervals. These pairs do not describe any business events and are generated only to accelerate the processing of temporal queries. We execute additional transactions to ingest these pairs, and these transactions are not part of the business logic either. As model INT involves executing additional transactions, there is an impact on Fabric throughput and less number of business transactions are executed per second.

We can retrieve asset k events within an indexing interval θ by executing a GHFK call on (k, θ), and this will require accessing only one block. To retrieve events in a query time-interval Ω, i.e., E(k, Ω), we first find out the set of indexing intervals θ that overlap with time-interval Ω and for each interval θ in θ, we then execute a GHFK call for the key (k, θ). If there are n indexing intervals in set θ, this requires n GHFK calls accessing one block each. This significantly reduces the number of blocks accessed and hence speeds up the retrieval of events of interest.

5.1 Indexing by Modifying Transactions (IMT)

Model IMT stores the indexing interval information along with each key-value pair ingested. An incoming key-value pair [k, (v, t)] is transformed to [(k, θ), (v, t)], wherein θ is an indexing interval and timestamp t lies within index interval θ. We discard the pair [k, (v, t)] and only store the transformed pair. Model IMT does not impact Fabric throughput, as no additional transactions are executed. The cost of model IMT is that we can no longer issue the native GHFK calls on key k. This is because we no longer have any asset identified by key k. To compute the native output of a GHFK call on key k, we need to issue a GHFK (k, θ) call for each indexing interval θ and collect the output.

The simplest indexing strategy is to take all indexing intervals of length u. Various strategies of selecting indexing intervals are possible, and we refer to [40] for a detailed discussion on these models and indexing strategies.

5.2 Experimental evaluation

Use-case: We carried out an experimental evaluation of the two models using synthetic data. Consider a blockchain-supported supply chain use-case wherein a set of shipments is placed in containers and the containers are ferried by trucks. Whenever a shipment s is placed in a container c at time t, a key-value pair [s, ((c, ''l''), t)] is inserted on Fabric. When shipment s is taken out of container c at time t, the pair [s, ((c, ''ul''), t)] is inserted. The symbols l and ul denote load and unload events. Similarly, the events [c, ((tr, ''l''), t)] and [c, ((tr, ''ul''), t)] denote the events when a container c is loaded on/unloaded from a truck tr at time t.

Fabric instance: We use Fabric release v1.0, single-peer setup running on a Lenovo T430 machine with 4 GB of RAM and dual-core Intel i5 processor. We use a single peer, but we keep the consensus mechanism turned ON. We use all default configuration settings to run our experiments.

Datasets: We generate synthetic data wherein number of shipments, containers, and trucks are taken to be 400, 100, and 20, respectively. The number of events for each shipment and container is 2K with uniform event distribution, yielding 1M of the total number of events. All events lie within time-range [0–150K]. For each transaction, we choose a batch of events to ingest with each batch being a maximal set of consecutive events s.t.: in this set, no two events share the same key. Different transactions, hence, insert a different number of events.

Temporal query: We consider the following temporal join query. Given a duration Ω: [ts; te], for each shipment s, we want to find out the set of trucks that ferried the shipment s at any time within queried duration Ω and the associated time intervals. This requires retrieving shipment and container load and unload events within time-interval Ω and taking a join over the two sets of events.

For both models, we use indexing intervals of length 2K. We vary the query interval Ω from (1–10K) to (140–150K). **Figure 7** presents the results. In the no-indexing case, the time increases from 12 to 120 seconds as the query interval moves right. This is because of a large number of accessed blocks. The timings of the INT and IMT models remain unaffected. This is because the number of blocks accessed by these models depend only on the length of the query interval, whereas the no-indexing case requires us to access all blocks in set B(k, te). The INT model performs better than the IMT model because, in the INT model, we collect pairs for key k during each successive interval of length 2K and ingest new pairs containing these collections. This approach requires access to significantly fewer blocks than the IMT model.

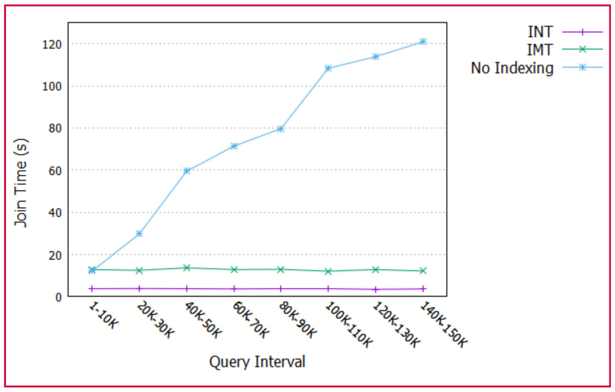


Figure 7

Join time for temporal queries using different models.

6 Compliance analysis

In this section, we focus on the compliance analysis, an instrumental task for identification of risks, violations, fraud and other problems, and a vital audit instrument. In general, compliance analysis includes a series of checks against assets stored on the blockchain ledger. In many scenarios, it is necessary to check the compliance of the assets with contractual, legal, regulatory, and other types of complex rules. The compliance may need to be repeatedly checked for newly recorded assets or due to changes in the rules, e.g., new rules not known at the time of the asset recording or rules that have changed over time. The distributed ledger available at different locations across the blockchain network provides a unique opportunity for organization-specific and workload-intensive compliance analysis of the shared data.

Data are stored on the blockchain ledger in the form of assets contained in transactions generated by a smart contract. During generation of the transaction, the smart contract checks the validity of the data with a set of rules encoded into the definition of the smart contract. The rules define the basic conditions, which must be valid about the asset before it can be recorded in the ledger. Once data pass the smart contract validation and subsequent confirmation with a consensus mechanism, the data are recorded on the immutable shared ledger and become available for analysis. The ledger contains a complete history of every data manipulation transaction and thus serves as a comprehensive source of information for compliance analysis and audit. Aside of the asset data, the transactions recorded on the ledger contain metadata essential for compliance analysis, such as the timestamp, identity of the transaction recording participant, and identity of the endorsing (transaction validating) parties.

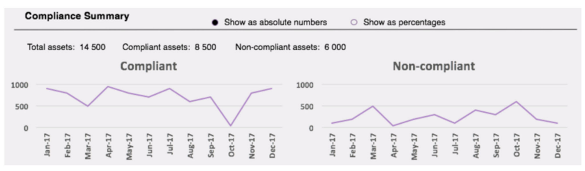
The compliance analysis may involve a wide spectrum of rules, which define the conditions of compliance of individual assets (e.g., KYC checks of financial transactions). The rules may originate from various sources, such as business contracts and agreements, regulations, laws, and treaties. The checking of the rules may require the use of off-chain data held in external sources. Additionally, complex rules may require analysis of an asset in the context of related assets or a series of related assets. For example, a single financial transaction may be independently valid but, in the context of other related transactions with certain characteristics, may indicate a violation of anti-money-laundering rules.

Compliance analysis of the blockchain ledger must address several unique challenges. First, the assets are stored in the ledger in the form of key-value pairs where the value is typically a complex document containing structured data (e.g., JSON). The schema describing the structure of the asset is implicitly defined in the smart contract and not explicitly recorded on the ledger (except for data models of Hyperledger Composer [41]. However, for data analytical tasks, the schema is critical information, which must be obtained in advance of analysis. Second, an asset can be modified and stored on the ledger multiple times in different transactions. Consequently, multiple versions of the same asset may be found in the ledger, each with a different state, timestamp, and metadata. Therefore, the analysis must consider the asset state evolution, the metadata, as well as the time-bound correlation of transactions recording related assets.

We have designed and implemented the Compliance Engine (CE), an analytical tool for the compliance analysis and audit of data held on the shared ledger of the Hyperledger Fabric. The CE allows evaluating a wide spectrum of rules against the assets stored on the ledger. The CE can be either deployed as a container connected to a peer of Hyperledger Fabric network hosted either on-premises including highly secure environments [42], in a cloud such as the IBM Blockchain Platform, or integrated with the peer and accessed by command line interface. The CE conducts the compliance analysis against the data held on ledgers of the peer. Note that the ledger may contain data the CE cannot interpret (e.g., encrypted data), and such data will be ignored in the analysis.

The compliance analysis has several steps. First, to analyze data on the ledger, the CE identifies the schema of the assets stored on the ledger. The schema can be either extracted from the data model stored along the smart contracts implemented in Hyperledger Composer or dynamically derived and extracted from the assets stored on the ledger by the native Fabric smart contracts implemented in the Go or JavaScript languages. The dynamic schema extraction method uses a clustering algorithm to identify the asset types and attributes of those assets.

In the next step, an analyst defines the compliance rules to be evaluated against the data stored on the ledger. The rules are structured as logical expressions, which upon evaluation result into a logical value (either True or False) representing either compliance or noncompliance of the asset with the rule. Each rule is assigned to one or more



Compliance Summary

● Show as absolute numbers ○ Show as percentages

Total assets: 14 500 Compliant assets: 8 500 Non-compliant assets: 6 000

Figure 8

Compliance dashboard with summary of the analysis.

asset types to which it applies. For instance, a rule that verifies a source of funds will apply to financial transactions, whereas a different rule will be applicable only to assets representing a letter of credit. The analyst has a choice to define the rules in JavaScript code or in a Controlled English-based language derived from a rule framework for the definition of smart contracts [43].

Upon execution of the analysis, the CE connects to a peer and extracts assets and metadata from blocks and transactions of the analyzed ledger and store them into a secure in-memory database. The in-memory database is either hosted in the CouchDB instance of the state database, or when highly secure configuration is required, the in-memory database is hosted in a dedicated and isolated instance of CoudDB. The assets are then classified into the schema types based on the asset data and metadata (e.g., the type of transaction that created the asset, the values of attributes of the asset, and others). The CE then sequentially iterates through the assets and analyzes them independently. For each asset, related assets required to check the compliance are retrieved from the ledger. For example, to evaluate rules against a financial transaction, the assets representing a source and destination accounts may need to be retrieved and made available to the analysis. The bundle of the asset and the related data are then evaluated against all applicable rules. The outcome of the rules evaluation is then stored in a database holding the results of the analysis.

The results of the analysis are displayed to the analyst in the compliance dashboard. The dashboard provides a summary view of the compliance status plotted as the number of assets over the time the assets were recorded on the ledger shown in **Figure 8**. The dashboard further allows for drill-down and display of details about compliance status on the level of individual rules and asset types. The dashboard also allows us to display and browse through the list of noncompliant assets and to inspect their details, such as which rules are violated. However, the noncompliant assets will typically require further analysis and appropriate actions, such as the use of ML to identify patterns of noncompliance causes. Hence, the noncompliant assets along with the rules' evaluation results are stored in the database and are made available as input into further analysis.

7 Federated learning

In this section, we focus on the integration of federated learning and blockchain. The data used in model training are often distributed across multiple geographic locations. To utilize the data for classification, regression, or other learning-based tasks, we need to rely on pooling all the data to a centralized location, and then perform model training on the aggregated data. However, communication requires hours to move large-scale data (terabytes, petabytes) with commensurate bandwidth, CPU, and networking resources. In addition, data are often proprietary, might contain sensitive information about the owner, and are subject to various regulatory, compliance, and privacy policies—further restricting their movement across the different sites. Blockchain can be leveraged as both the source of data processed with the learning algorithms as well as a mechanism to maintain the provenance of data and mechanisms executed over multiple iterations of the federated learning process.

Federated learning has recently received a lot of attention as a mechanism for model training under data parallelism (setting in which the data, as opposed to the model, are partitioned between multiple geographically separated sites) [44]. The key idea in federated learning is to create a global model to gain from the collective insights available in the local datasets while still allowing the data owners (or agents) to retain control over their data. This is typically achieved by allowing each agent to train local models using their data and share the model parameters with a central server (Fusion Manager) instead of the raw data. The Fusion Manager runs a fusion algorithm that is designed to combine the updates from the different agents into a global model. The local models, which are typically lossy and compact representations of the local data, encode information that, when combined over multiple rounds, leads to a highly accurate global model. In the following paragraphs, the model refers to a multilayer neural network (a convolutional neural network, feedforward neural network, etc.). However, the discussion is also applicable to traditional ML models, such as support vector machine (SVM), decision trees, etc. There are several aspects that need to be considered in the design of the federated-learning framework. These include but are not limited to the following:

1) distribution of the data partitions among agents;
2) choice of the model architecture;
3) model hyperparameters;
4) synchronous versus asynchronous gradient descent;
5) fusion algorithms.

Data distribution refers to how the classes (in the training data) are partitioned between the agents. Typically, the partitions are assumed to be independent and identically distributed, where each agent has a proportion of samples from every class as per the empirical data distribution. However, in practice, the distribution of labels or classes is often skewed between the agents [45–47]. This imbalance needs to be accounted for during the fusion process and typically determines the factor by which the model updates from the agents are weighed before combining. In addition, the data could also be horizontally or vertically partitioned. In horizontal partitioning, all the data features are available at every agent, whereas in vertical partitions, possibly disjoint feature subsets are available at different agents [48].

Model architecture refers to the configuration of the model in terms of the number of layers, neurons in each layer, etc. Most prior works on federated learning assume homogeneity in terms of the model architecture among agents. The training also depends on hyperparameters, such as number of training epochs, minibatch sizes over which the gradients are computed, and the frequency of model updates. These parameters, shared between the agents, are typically provided by the Fusion Manager at the start of the fusion process. Targeted learning via latent space transformation, between nonhomogenous models, has been proposed in [49].

Training algorithm refers to the mechanisms used to update the weights of a neural network model. Most networks use some variant of the *gradient descent* algorithm for parameter update. In the federated-learning setting, a distributed variant of the gradient descent algorithm is used. The algorithm can be applied in the *synchronous setting*, in which the Fusion Manager issues a blocking call and waits for all the agents to provide their update before initiating the fusion operation. The training rate is thus determined by the speed of the slowest agent.

In the *asynchronous mode*, the base model is modified as and when the updates from the agents arrive. The Fusion Manager does not wait for all the updates to initiate fusion [50].

Fusion algorithms are key to federated learning. These are mechanisms to combine the parameter updates from the agents into a single global model. Typically, the fusion algorithms are multiround protocols, where each round is initiated by sharing a global model with the agents and ends by fusing the updates from the agents into a global model—ready for the next round. The fusion algorithms are required to adapt to the other components of the framework (data distribution, training algorithm, etc.).

A parameter-sharing-based solution for distributed learning using homogenous models was proposed in [46]. At each agent, a model is trained using local data. After a fixed number of training rounds, parameters from the local models are shared with the Fusion Manager, which performs the fusion of these parameters via weighted averaging and sends the aggregated parameters back to the different agents. Each agent reinstantiates a new model using the received parameters and starts retraining it with the local data. This process (of sharing and receiving new parameters) continues until the aggregated parameters converge. As is evident, more rounds of local training reduce the amount of data that needs to be shared [51]. A variation of the scheme in [52], for highly skewed data with minibatch gradient descent, has been proposed in [47].

Similar to other learning systems, distributed learning is also vulnerable to attacks from malicious agents that share model updates designed to change the rate of convergence of the learning process or craft model poisoning attacks for targeted misclassification by the global model [53].

Blockchain facilitated distributed learning can alleviate these threats by providing a reliable and permissioned medium for the exchange of the model parameters. Blockchain can be leveraged to maintain provenance and also introduce trust into the federated-learning process. The shared ledger makes data simultaneously available at multiple locations. The ledger holds strictly structured, classified, and trusted data. Unlike other sources, blockchain allows us to verify tampering and origin of data, making blockchain a highly trusted data source. Data held on blockchain are thus a natural fit for processing by distributed learning agents. Additionally, the Blockchain permissioning constrains the agents to appropriate data partitions based on identity and granted access rights. In the environments of multiple independent blockchain ledgers, each agent is assigned to processing a subset of the total data, extracted from the ledger to which it is connected. In this approach, the data are made available for learning while the operators can enforce the required privacy, regulatory, and other requirements. Moreover, blockchain can be leveraged to maintain an association between the model parameters and the minibatches used to train those parameters. By subjecting agents to random verification (or audit), poisoning attacks can also be discouraged, leading to better trust in the global model.

Finally, orthogonal to model poisoning, the parameter updates provided by the local agents could also be used to infer valuable information about the training data. Privacy-preserving distributed learning methods are being investigated [45, 48, 52] to address some of these concerns.

8 Blockchain for trusted AI

With the increasing importance of AI in both everyday life and in critical business processes, the notions of trust in the AI data, models, learning process, and outcomes are becoming increasingly important. The need for trusted AI is elevated even more by AI becoming more distributed than ever—data and models are being processed, shared, and reused by different organizations, models can be trained in a federated way, etc. Blockchain can help to address some of the trust-related concerns of AI processes.

Blockchain, with its distributed immutable ledger, can help to track a given AI process at different granularity levels, depending on the application at hand. The access-control, privacy, and confidentiality mechanisms provide a way for confidential sharing, where no unintended party can access restricted data or models. Blockchain can record and possibly also enforce (by means of smart contracts and agreed upon consensus mechanism) the order of exchanges between parties immutably, which enables fair assessment of participant contributions. The need for such an ordering in computing contribution of participant data or model arises from the observation that "value" of any exchange of information is heavily dependent on the state, i.e., what information is already known or how far along in the training process. Finally, the immutable ledger can be used for tracing of ownership and usage across complex provenance chains.

8.1 Illustrative requirements for trusted AI

We briefly discuss several requirements that can be considered in connection with trust in an AI process.

Reuse and composition: Training of AI models is expensive in terms of time and compute resources. Hence, there is a need to reuse and compose AI assets in order to construct new AI assets. Composition also leads to challenges in attribution of value provided by various owners of the models and tracking the provenance of these models.

Confidential and trusted sharing: Oftentimes, models are trained with confidential data. It is crucial that the privacy and ownership of the models and data are preserved in complex cross-organizational compositions of models.

Verification and auditability: We must ensure that the AI process is auditable and verifiable by a third party, ideally without loss of data privacy or ownership. For instance, it is often necessary to ensure that a model is trained following certain specification or policy—e.g., using a particular dataset.

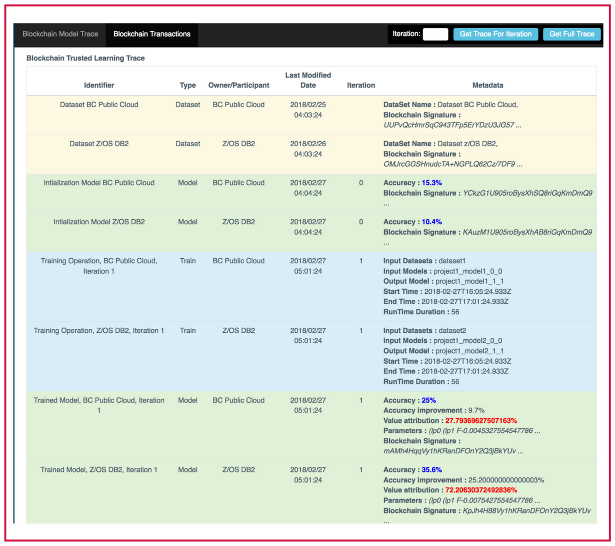
Provenance: Track the origin, ownership, and use of AI assets through an immutable and tamper-free record.

Fairness: Provide fairness guarantees and mechanisms that can help with avoiding biases.

8.2 Tracking AI process using Blockchain

We have built a blockchain library that provides key capabilities and modeling constructs needed to address some of the concerns and requirements of trusted AI processes. The key blockchain modeling constructs involved are assets representing participants, datasets, models, and operations. We now describe these constructs by means of an illustrative example of the federated-learning use case described in Section 7.

Figure 9 shows a snapshot of the blockchain transactions representing part of the trace of the training process of federated learning from the actual implementation. The trace captures the details of the process, in this case, the training process, e.g., who provides what data, how the data



Blockchain Trusted Learning Trace

Identifier	Type	Owner/Participant	Last Modified Date	Iteration	Metadata
Dataset BC Public Cloud	Dataset	BC Public Cloud	2018/02/25 04:03:24		**DataSet Name :** Dataset BC Public Cloud, **Blockchain Signature :** *UUPvQcHmrSqC943TFp5ErYDzU3JG57 ...*
Dataset Z/OS DB2	Dataset	Z/OS DB2	2018/02/26 04:03:24		**DataSet Name :** Dataset z/OS DB2, **Blockchain Signature :** *OMJrcGGSHnudcTA+NGPLQ62Cz/7DF9 ...*
Intialization Model BC Public Cloud	Model	BC Public Cloud	2018/02/27 04:04:24	0	**Accuracy : 15.3%** **Blockchain Signature :** *YCkzG1U905roBysXhSQ8riGqKmDmQ9* ...
Intialization Model Z/OS DB2	Model	Z/OS DB2	2018/02/27 04:04:24	0	**Accuracy : 10.4%** **Blockchain Signature :** *KAuzM1U905roBysXhAB8riGqKmDmQ9* ...
Training Operation, BC Public Cloud, Iteration 1	Train	BC Public Cloud	2018/02/27 05:01:24	1	**Input Datasets :** dataset1 **Input Models :** project1_model1_0_0 **Output Model :** project1_model1_1_1 **Start Time :** 2018-02-27T16:05:24.933Z **End Time :** 2018-02-27T17:01:24.933Z **RunTime Duration :** 56
Training Operation, Z/OS DB2, Iteration 1	Train	Z/OS DB2	2018/02/27 05:01:24	1	**Input Datasets :** dataset2 **Input Models :** project1_model2_0_0 **Output Model :** project1_model2_1_1 **Start Time :** 2018-02-27T16:05:24.933Z **End Time :** 2018-02-27T17:01:24.933Z **RunTime Duration :** 56
Trained Model, BC Public Cloud, Iteration 1	Model	BC Public Cloud	2018/02/27 05:01:24	1	**Accuracy : 25%** **Accuracy improvement :** 9.7% **Value attribution : 27.79369627507163%** **Parameters :** *(lp0 (lp1 F-0.0045327554547786 ...* **Blockchain Signature :** *mAMh4HqqVy1hKRanDFOnY2Q3jBkYUv ...*
Trained Model, Z/OS DB2, Iteration 1	Model	Z/OS DB2	2018/02/27 05:01:24	1	**Accuracy : 35.6%** **Accuracy improvement :** 25.200000000000003% **Value attribution : 72.20630372492836%** **Parameters :** *(lp0 (lp1 F-0.0075427554547786 ...* **Blockchain Signature :** *KpJh4H88Vy1hKRanDFOnY2Q3jBkYUv* ...

Figure 9

Snapshot of the blockchain transactions representing part of the trace of the training process of federated learning.

are consumed, which participants and operations produce the trained model instances, etc.

Specifically, there are two types of *participants* involved in this setting: *agents* and *fusion manager*. Optionally, there could exist a third type of participant known as the *auditor,* whose role is to audit the process by constructing the provenance graph and ensure different aspects of fairness and confidentiality have been maintained. Each agent uses its own *dataset* for training and testing purposes. The auditor could also own a test dataset for scoring purposes. In this setting, a *model* is represented by the weights of the underlying neural network being trained. In each iteration of the process, a new *model* is generated by each of the agents and the fusion manager (i.e., we illustrate very fine granularity of the process). In this setting, there are two types of *operations* involved: *learning operations* and *fusion operations.* As the name suggests, the *learning operations* are recorded actions of the agents and contains details, such as reference of the agent, training dataset signature, model before update (given by the fusion manager) and model after update, etc. The *fusion operation,* on the other hand, records actions of the *fusion manager* and includes details corresponding to models from various agents involved in the fusion process and the model generated by the fusion process.

9 Future work

Blockchain provides a decentralized, trusted, immutable method to share and record data. Creating analytics for data starts with providing a dashboard or a visualization of the data, what we call "descriptive" analytics. This is followed by "predictive" analytics, using machine-learning models to find patterns in data that help us answer questions of future events or trends that are not explicit in the data. AI encompasses machine-learning models with natural language understanding with intuitive visualizations to allow humans to easily interact with blockchain data. Future work would include advances in visualizations that highlight trends in an automated way that does not require explicit programming from the user. For predictive analytics, we described capabilities of Watson Studio that can help users recommend which models are best suited to the blockchain data for accurate predictions. In AI, future work would include being able to "talk" to blockchains using natural languages to query blockchain events and ask for future trends and interact with the blockchain data in "what if" scenarios.

Blockchains currently have "smart contracts" that can be used to transform, check, and manipulate blockchain data, given a set of business conditions. Going further with AI, one can advance smart contracts to "oracles" that are programmed/created and run on blockchains on behalf of users and enterprises to find the best price for products that are offered on the blockchain. Oracles can engage with each other to auction/bid on behalf of their users given rules that set limits/conditions on what range of products, volume, prices to offer/accept.

For performance and scalability, in Section 5, we have discussed how to efficiently execute temporal queries on Hyperledger Fabric. However, complex queries integrating temporal as well as multidimensional attributes (e.g., time and geographical location) are more challenging. Our future work will focus on exploration of how the data can be stored and efficiently accessed on blockchain platforms including Hyperledger Fabric, and how to accelerate specific types of these queries. Moreover, in cases when the data are in the form of graphs, we will investigate efficient on-chain methods for storage and access of the graph nodes, or application of graph algorithms such as traversal. These problems are especially challenging due to the restrictive APIs of Fabric and of other blockchain platforms.

10 Conclusion

We have presented a series of business-ready tools that analyze data recorded on blockchain by supply chains, financial payment systems, various business processes, and other transactional sources. The tools are centered on the Hyperledger Fabric platform and leverage the blockchain specific characteristics of the data, such as the transactional history, identity of participants, and timestamps among others, and significantly accelerate the complex tasks of analysts, data scientist, and business specialists. We have also highlighted the importance of the use of trusted data in analytical tasks and outlined how blockchain can help in the federated learning. Finally, as the blockchain technologies, data science methods, and AI models are rapidly evolving, so do we continue to advance the capabilities of the analytical tools, and we have highlighted some of the key future directions.

References

1. Linux Foundation - Hyperledger Fabric. Jan. 2019. [Online]. Available: https://www.hyperledger.org/projects/fabric
2. N. Gaur, L. Desrosiers, P. Novotny, et al., *Hands-On Blockchain With Hyperledger: Building Decentralized Applications With Hyperledger Fabric and Composer*. Birmingham, U.K.: Packt Publishing, 2018
3. R. Wolfson, "Understanding how IBM and others use Blockchain Technology to track global food supply chain," *Forbes*, Jul. 11, 2018.
4. we.trade. Jan. 2019. [Online]. Available: https://we-trade.com/
5. J. J. Roberts, "IBM and stellar are launching blockchain banking across multiple countries," *Fortune*, Oct. 16, 2017.
6. E. E. Wood, "SecureKey announces blockchain based identification service," Feb. 8, 2018. [Online]. Available: itbusiness.ca
7. Gemological Institute of America, "Chow Tai Fook and GIA bring diamond grading reports to consumers via blockchain," May 23, 2018. [Online]. Available: https://www.gia.edu/gia-news-press/chow-tai-fook-gia-blockchain
8. M. D. Castillo, "IBM-Maersk blockchain platform Adds 92 clients as part of global launch," *Forbes*, Aug. 2018.
9. G. Press, "Cleaning big data: Cost time-consuming, least enjoyable data science task, survey says," *Forbes*, Mar. 2016.
10. L. Muñoz-González, B. Biggio, A. Demontis, et al., "Towards Poisoning of Deep Learning Algorithms with Back-gradient Optimization," in *Proc. 10th ACM Workshop on Artificial Intell. and Security (AISec '17)*, ACM, New York, NY, USA, 2017, pp. 27–38. doi: https://doi.org/10.1145/3128572.3140451
11. M. Jagielski, A. Oprea, B. Biggio, "Manipulating machine learning: poisoning and countermeasures for regression learning," in *Proc. 39th IEEE Symp. Secur. Privacy*, Apr. 2018, pp. 19–35.
12. T. T. A. Dinh, Ji Wang, Gang Chen, et al., "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. ACM Int. Conf. Manage. Data*, 2017, pp. 1085–1100.
13. Ethereum Blockchain App Platform. Accessed: Jan. 2019. [Online]. Available:https://www.parity.io/
14. Ethcore. *Parity: Next Generation Ethereum Browse*. Accessed: Jan. 2019. [Online]. Available: https://www.parity.io/
15. C. Decker and R. Wattenhofer, "Information propagation in the Bitcoin network," in *Proc. IEEE P2P*, 2013, pp. 1–10.
16. K. Croman, C. Decker, I. Eyal, et al., "*On Scaling Decentralized Blockchains - (A Position Paper)*," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2016, pp. 106–125.
17. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Accessed: Jan. 2019. [Online]. Available: https://bitcoin.org/bitcoin.pdf
18. D. Gao, C. S. Jensen, R. T. Snodgrass, et al., "Join operations in temporal databases," *Int. J. Very Large Data Bases*, vol. 14, no. 1, pp. 2–29, 2005
19. M. F. Mokbel, T. M. Ghanem, and W. G. Aref, "Spatio-temporal access methods," *IEEE Data Eng. Bull.*, vol. 26, pp. 40–49, 2003
20. B. Chawda, H. Gupta, S. Negi, et al., "Processing interval joins on map-reduce," in *Proc. 17th Int. Conf. Extending Database Technol.*, 2014.
21. Coinalytics. Accessed: Jan. 2019. [Online]. Available: https://www.crunchbase.com/organization/coinalytics-co
22. BlockParser. Accessed: Jan. 2019. [Online]. Available: https://github.com/mcdee/blockparser
23. D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Proc. Int. Conf. Financial Cryptography Data Secur.*, 2013, pp. 6–24.
24. Datum. Accessed: Jan. 2019. [Online]. Available: https://datum.org/
25. TraneAI, "Decentralized network for artificial intelligence at scale, white paper. Accessed: Jan. 2019. [Online]. Available: https://github.com/TraneAI/Whitepaper
26. Ocean Protocol, white paper. Accessed: Jan. 2019. [Online]. Available: https://oceanprotocol.com/tech-whitepaper.pdf
27. Computable Labs. Accessed: Jan. 2019. [Online]. Available: https://www.computable.io/
28. OpenMined. Accessed: Jan. 2019. [Online]. Available: https://www.openmined.org/
29. A. B. Kurtulmus and K. Daniel, "Trustless machine learning contracts; Evaluating and exchanging machine learning models on the Ethereum blockchain," 2018, *arXiv:1802.10185*. [Online]. Available: http://arxiv.org/abs/1802.10185
30. Neureal, "Open-source, peer-to-peer, AI supercomputing, Live data stream prediction powered by blockchain, Infinitely scalable." Accessed: Jan. 2019. [Online]. Available: https://docs.google.com/document/d/1kOJx7clG2V4TevhgwndRDievXpVaAciPzjmqGxI0CtA/view#
31. LevelDB. Accessed: Jan. 2019. [Online]. Available: http://leveldb.org/
32. CouchDB. Accessed: Jan. 2019. [Online]. Available: http://couchdb.apache.org/
33. IBM Corp., IBM Blockchain Platform. Accessed: Jan. 2019. [Online]. Available:https://www.ibm.com/blockchain/platform
34. Watson Studio. Accessed: Jan. 2019. [Online]. Available: https://www.ibm.com/cloud/watson-studio
35. The Weather Company. Accessed: Jan. 2019. [Online]. Available: https://business.weather.com/
36. Apache Spark. Accessed: Jan. 2019. [Online]. Available: https://spark.apache.org/
37. Brunel Visualizations. Accessed: Jan. 2019. [Online]. Available: https://dataplatform.cloud.ibm.com/docs/content/analyze-data/brunel-visualization.html
38. Regression. Accessed: Jan. 2019. [Online]. Available:https://en.wikipedia.org/wiki/Regression_analysis
39. REST. Accessed: Jan. 2019. [Online]. Available:https://en.wikipedia.org/wiki/Representational_state_transfer
40. H. Gupta, S. Hans, S. Mehta, et al., "On building efficient temporal indexes on Hyperledger Fabric," in *Proc. IEEE 11th Int. Conf. Cloud Comput.*, 2018, pp. 294–301.
41. Hyperledger Composer. Accessed: Jan. 2019. [Online]. Available: https://www.hyperledger.org/projects/composer
42. A. Nuñez Mencias, D. Dillenberger, P. Novotny, et al., "An optimized blockchain solution for the IBM z14," *IBM J. Res. & Dev.*, vol. 62, no. 2/3, pp. 4:1–4:11, 2018.
43. T. Astigarraga, X. Chen, Y. Chen, et al., Business-level blockchain users with a rules framework for smart contracts, in *Proc. 16th Int. Conf. Serv.-Oriented Comput.*, 2018, pp. 111–128.
44. J. Dean, G. S. Corrado, R. Monga, et al., "Large scale distributed deep networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.
45. N. Papernot, M. Abadi, U. Erlingsson, et al., "Semi-supervised knowledge transfer for deep learning from private training data," in *Proc. Int. Conf. Mach. Learn.*, 2017.
46. H. B. McMahan, E. Moore, D. Ramage, et al., "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017.
47. D. Verma, S. Chakraborty, S. Calo, et al., "An algorithm for model fusion for distributed learning," *Proc. SPIE*, vol. 10635, 2018, Art. no. 106350O.
48. S. Hardy, W. Henecka, H. Ivey-Law, et al., "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," 2017, *arXiv:1711.10677*.
49. T. Xing, S. S. Sandha, B. Balaji, et al., "Enabling edge devices that learn from each other: Cross modal training for activity recognition," in *Proc. 1st Int. Workshop Edge Syst., Anal. Netw.*, 2018, pp. 37–42.
50. X. Lian, W. Zhang, C. Zhang, et al., "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3049–3058.
51. S. Wang, T. Tuor, T. L. Salonidis, et al., "When edge meets learning: Adaptive control for resourceconstrained distributed machine learning," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2018, pp. 63–71.
52. K. Bonawitz, V. Ivanov, B. Kreuter, et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191.
53. E. V. Bagdasaryan, A. Veit, and Y. Hua, et al., "How to backdoor federated learning," 2018, *arXiv:1807.00459*.

Received May 31, 2018; accepted for publication January 25, 2019.

Donna N. Dillenberger *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (engd@us.ibm.com).* Ms. Dillenberger is an IBM Fellow working on enterprise solutions, including blockchain, analytics, and artificial intelligence solutions to detect counterfeit products.

Petr Novotny *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (p.novotny@ibm.com).* Dr. Novotny is a Research Staff Member working on blockchain technologies including Hyperledger Fabric platform, analytical and compliance tools, and seminal blockchain use cases.

Qi Zhang *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (q.zhang@ibm.com).* Dr. Zhang is a Research Staff Member with interests in blockchain systems, cloud computing, big data processing, and distributed systems.

Praveen Jayachandran *IBM Research, Bengaluru 560045, India (praveen.j@in.ibm.com).* Mr. Jayachandran is a Senior Technical Staff Member and Master Inventor with IBM Research-India. His work spans different aspects of blockchain technology, including developing an enterprise-grade blockchain platform and reimagining industry use cases in a blockchain world.

Himanshu Gupta *IBM Research, New Delhi 110070, India (higupta8@in.ibm.com).* Mr. Gupta is a Senior Researcher with the AI Engineering department, IBM Research-India. His research interests include data management, data mining, information integration, Spark/map-reduce-based processing, and blockchain analytics.

Sandeep Hans *IBM Research, New Delhi 110070, India (shans001@in.ibm.com).* Mr. Hans is a Researcher with the AI Engineering department, IBM Research-India. His research interests include distributed computing, data management in blockchain, and blockchain analytics.

Dinesh Verma *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (dverma@us.ibm.com).* Dr. Verma is an IBM Fellow working on technology areas at the intersection of Internet of Things, artificial intelligence, and distributed systems.

Supriyo Chakraborty *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (supriyo@us.ibm.com).* Mr. Chakraborty is a Research Staff Member working on information privacy, applied cryptography, and adversarial machine learning.

John J. Thomas *IBM Corporation, Armonk, NY 10504 USA (jothomas@us.ibm.com).* Mr. Thomas is an IBM Distinguished Engineer and the Director in IBM's Data Science Elite team. He works with IBM clients on applying machine learning and DL to various use cases.

Matthew M. Walli *IBM Corporation, Lexington, KY 40575 USA (mwalli@us.ibm.com).* Mr. Walli is an Architect with the IBM Analytics Data Science Elite Team collaborating with IBM clients to integrate machine learning and deep learning technologies into enterprise environments.

Roman Vaculin *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA* (*vaculin@us.ibm.com*). Mr. Vaculin is a Research Manager of the Blockchain Solutions Research team. He works on blockchain innovations including blockchain, artificial intelligence, and Internet of Things.

Kanthi Sarpatwar *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA (sarpatwa@us.ibm.com).* Dr. Sarpatwar is a Research Staff Member working on artificial intelligence and blockchain solutions.