Machine learning-driven automatic storage space recommendation for object-based cloud storage system

Anindita Sarkar Mondal¹ · Anirban Mukhopadhyay² · Samiran Chattopadhyay^{1,3}

Received: 12 April 2021 / Accepted: 26 August 2021 / Published online: 25 September 2021
© The Author(s) 2021

Abstract

An object-based cloud storage system is a storage platform where big data is managed through the internet and data is considered as an object. A smart storage system should be able to handle the big data variety property by recommending the storage space for each data type automatically. Machine learning can help make a storage system automatic. This article proposes a classification engine framework for this purpose by utilizing a machine learning strategy. A feature selection approach wrapped with a classifier is proposed to automatically predict the proper storage space for the incoming big data. It helps build an automatic storage space recommendation system for an object-based cloud storage platform. To find out a suitable combination of feature selection algorithms and classifiers for the proposed classification engine, a comparative study of different supervised feature selection algorithms (i.e., Fisher score, F-score, L1L21) from three categories (similarity, statistical, sparse learning) associated with various classifiers (i.e., SVM, *K*-NN, Neural Network) is performed. We illustrate our study using RSoS system as it provides a cloud storage platform for the healthcare data as experimental big data by considering its variety property. The experiments confirm that L1L21 feature selection combined with *K*-NN classifier provides better performance than the others.

Keywords Cloud storage system · Classification engine · Automatic storage space recommendation · Energy efficient data centers · Healthcare data

Introduction

Nowadays, the application areas of a cloud storage system are not limited to digital data deposits. Many machine learning approaches [2, 12, 16] are associated with renowned cloud storage systems to make them computationally advanced. Storage location prediction is one of the crucial applications

of cloud storage systems. In this article, we have utilized health-related data to illustrate this task.

Health data comes from various organizations, like sensor provider, hospital manager, account manager, patients relatives, etc. and the structure of the generated data is varied with the source. Eventually, more information is added to the existing one. With time, they may feel to store additional information on the currently stored objects [11]. Therefore, they follow the unstable structure format, transmit towards the unstable structure schema. All of them make the storage space prediction job more complex.

Different cloud storage systems are available in the literature to support health data. These include Amazon S3 [28] that provides bucket-oriented object storage architecture, Openstack Swift that [27] supports account-container based object storage architecture, and Object based schema oriented data storage System (RSoS System) [24] that follows schema-based account-container oriented object storage architecture. Some time-series databases are also included to handle health sensor datasets. For example, PhilDB [20] uses metadata tracking architecture for identifying every time-

✉ Anindita Sarkar Mondal
sarkar.anindita5@gmail.com

Anirban Mukhopadhyay
anirbanbuba@yahoo.com

Samiran Chattopadhyay
samirancju@gmail.com

¹ Department of Information Technology, Jadavpur University, Kolkata, India

² Department of Computer Science and Engineering, Kalyani University, Nadia, India

³ Institute for Advancing Intelligence, TCG Centres for Research and Education in Science and Technology, Kolkata, India

series data type, and SciDB [38] follows native array data model for handling time-series datasets. The challenge is that health data is not only composed of time-series data or big chunks of data file, it is a combination of different types of datasets (time-series data, graph-based, file, etc.). Among all these database model variants, only schema-based architecture can provide the variant storage location to the different health data format. In such a way, the characteristics of this schema-based storage architecture not only supports the big data variety property, but also reduces the database operation execution time as shown in [24]. Due to this reason, we have considered RSoS system as a prototype design in this article.

RSoS may use different database models for different types of data to be stored. It is required to separate the storage space for different types of data. Identifying the types of data manually takes time and constant human intervention. Therefore it is required to automate the process of identifying the type of the data before sending it to its designated storage space. Machine learning can help us do this by predicting the storage space corresponding to each type of incoming data automatically.

Hybrid technologies are used in the healthcare domain to deal with different dataset formats like wiki-health platform [18], PolyglotHIS framework [15] and RSoS storage system [24]. All of them use more than one database model under a single storage layer. Cassandra [4], MongoDB [25] and HDFS [3] are widely used database models for big data. They are different in data type, storage capacity, storage architecture and data storage process.

RSoS system is a schema oriented object-based cloud storage system. Assigning object storage space requires enough database knowledge. In most of the cases, users do not have enough knowledge and time to define this object storage space. Also, if the user makes any mistake for defining the contents of object storage space then the data will not be stored in the proper place. To remove such complexity, we move towards designing an automated system to identify the corresponding storage space of the incoming dataset. Machine learning technology is one way to predict the storage space for the automated system to make it convenient for any human being.

In articles [23,24], the author discussed the detailed architecture, working procedure and performance comparisons of object-based schema oriented data storage system (RSoS system). To communicate with RSoS system (object based schema oriented data storage system), users use three queries in JSON format, viz., READ, WRITE and DELETE. Before storing any data to RSoS, a user needs to notify the related information like account, attribute name, attribute status (key, nonkey) and storage location (database name). The hypergraph data model maintains this information in a graphml.xml file, as presented in Listing 1. According to this listing, account values are U101, U102, U103 and

U104; attribute names are Time, Temperature, PatientID, PatientDetails, fileextension, remotefiledata and id. These storage devices are named by considering the used database model in that storage device. Here, three database models are used viz., Cassandra, MongoDB, and HadoopDFS. The mentioned Cassandra, MongoDB, and HadoopDFS are the name of storage devices. These storage locations are accessed using unique and individual URI, mentioned in Listing 1.

This hypergraph data model known as graphml.xml update is mandatory in every little change of data schema or the new dataset entry. This task is done manually, and it is a laborious and time-consuming job for human beings. Hence, in this article, our primary goal is to design a classification engine for the RSoS system that can reduce human trouble by predicting storage space automatically.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns/graphml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/graphml">
<graph edgedefault="directed">
<node id="1" userID="U101" name="Time" status="key"/>
<node id="2" userID="U101" name="Temperature" status="nonkey"/>
<node id="3" userID="U101:U104" name="PatientID" status="nonkey"/>
<node id="4" userID="U103" name="PatientID" status="key"/>
<node id="5" userID="U103" name="PatientDetails" status="nonkey"/>
<node id="6" userID="U102" name="fileextension" status="nonkey"/>
<node id="7" userID="U102" name="remotefiledata" status="nonkey"/>
<node id="8" userID="U102" name="id" status="key"/>
<edge source="1" target="2" HyperEdgeName="L1" HyperHyperEdgeName=
"Cassandra" URI="http://localhost:8080/cassandra/rest" />
<edge source="2" target="3" HyperEdgeName="L2" HyperHyperEdgeName=
"Cassandra" URI="http://localhost:8080/cassandra/rest" />
<edge source="3" target="1" HyperEdgeName="L3" HyperHyperEdgeName=
"Cassandra" URI="http://localhost:8080/cassandra/rest" />
<edge source="5" target="4" HyperEdgeName="L4" HyperHyperEdgeName=
"MongoDB" URI="http://localhost:8080/MongoRestServer/webresources
/rest"/>
<edge source="7" target="8" HyperEdgeName="L5" HyperHyperEdgeName=
"HadoopDFS" URI="http://localhost:8080/HadoopRestServer/webresources
/rest"/>
<edge source="6" target="7" HyperEdgeName="L6" HyperHyperEdgeName=
"HadoopDFS" URI="http://localhost:8080/HadoopRestServer/webresources
/rest"/>
<edge source="6" target="8" HyperEdgeName="L7" HyperHyperEdgeName=
"HadoopDFS" URI="http://localhost:8080/HadoopRestServer/webresources
/rest"/>
</graph>
</graphml>
```

Listing 1 Example of data contents presented in hypergraph data model

Based on the solicit, functionalities of the classification engine depend upon the associated applicant [22,40–42]. Here, the purpose of the classification engine is different from the existing ones. It motivates us to build a new classification engine framework for the RSoS system. RSoS system is a cloud storage system where data reaches within its own query format. Due to unique structure of every query for the different types of the datasets, generated datasets from each and every query act as the input to the designed classification engine. This generated dataset holds the system sub-parts status values of the server corresponding to the query. The input data looks like a two-dimensional matrix, $M = [m_{ij}] \in \mathbb{R}^{m \times n}$, m and n being the number of queries and the number of system sub-parts treated as features, respectively. The training dataset consists of one more column named class which holds the data type value of the corresponding query

with input data. The test dataset includes only the feature values of the input data.

A comparative study of feature selection approaches wrapped with classifiers is conducted to find suitable and efficient machine learning technologies for this framework. Firstly, the feature selection algorithms applied to the training dataset are responsible for selecting the relevant features. Next, the trained classifier determines the class value of the test dataset by considering the selected features. For the comparison purpose, three feature selection algorithms, viz., L1L21, Fisher score, F-score, and three classifiers, viz., K -NN, neural network, SVM with three kernel functions, viz., linear, polynomial, radial basis function (RBF) are considered here.

The ultimate aim of this machine learning technologies is to devise a feature selection algorithm along with a classifier to find proper object storage space for the incoming data presented in the write query. The experiment of this article is limited to the storage device URI prediction as the part of object storage space and the other parts viz., attribute list is constrained to the supporting storage device.

The major contributions of this article are listed below.

- A classification engine framework is proposed to predict the probable object storage space architecture from input data characteristics.
- The detailed architecture of the RSoS system associated with the proposed classification engine framework is presented.
- The brief workflow model of the classification engine framework is described. The detailed description about the involved sub units with their internal communication is also presented.
- A comparative analysis is performed to find out the proper combination of feature selection algorithm and classifier for the designed classification engine framework.

In the next section, we describe the existing classification engines. The overall information of RSoS system is explained in the next section. The following section describes the proposed classification engine framework with the workflow model. The main focus of the next section is to determine the components of the proposed classification engine, i.e., a feature selection algorithm and a classifier. An experimental comparative study is made here. Finally, the last section concludes the article with a discussion on future work.

Related work

In this section, we discuss some reputed classification engines and point out how they perform the tasks based on associated application's demand.

Development of the classification engine as machine learning technology for cloud storage system can broadly be categorized into two research areas, viz., the provided services of existing classification engines and machine learning approach associated with different cloud storage systems.

The main aim of varonis classification engines [40] is to classify, manage and protect the sensitive datasets from cyber-attacks. For this purpose, it monitors the user's behaviour and decides who will have access to which type of data. Moreover, it helps prevent unwanted data access.

ACE (autonomous classification engine) provides a framework to the user, where by giving feature vectors, the user can optimize the classifier, classifier parameters and reconfigure the problem [22]. By using this framework, the authors in [35] classify beatboxing sounds and find out that adaBoost classifier with C4.5 decision trees provides more accurate results than the others. Here, a genetic algorithm is used for feature vector generation.

Advanced classification engine (ACE) is maintained and designed by Websense security lab [security overview websense ACE (advanced classification engine)]. It is designed for providing real-time websense gateway and cloud security services. It works by classifying web page contents, URL and protocols inline based on the presence or absence of hidden malicious messages in page data contents [8,42]. PSIGEN releases the ACE (accelerated classification engine) [29]. It is able to custom classification of documents based on user requirements.

ACE is a framework, consisting of three classification engines, viz., data classification engine, storage classification engine and data placement engine for ILM (information life-cycle management) [34]. Here, storage location of a data item is shifted based on business value of the data. The authors in [10] proposed a classification engine which is used to identify the name of the social network platform of an image. This engine is built up using K -nearest neighbor (K -NN) classification and decision tree mechanism.

Integrated classification engine has been proposed by Veritas in [41]. Veritas is dedicated for multicloud data management where integrated classification engine is used to achieve the data protection requirements by classifying the risky and sensitive data worldwide.

Google cloud machine learning engine is a computing platform that provides training and prediction services [2]. Applying this platform, developers can build advanced complex machine learning models. One can use of google cloud machine learning engine in google cloud storage system as data processing purpose. Here, it auto-scales the number of server clusters to generate the result within the access time limit.

Amazon machine learning provides services to the developer for doing predictive analysis by drawing machine learning models [1,12]. For this reason, through the AWS

account, a user can select the amazon machine learning standard setup and the necessary data can be selected from amazon S3 and Redshift. After performing predictive analysis on the selected data set, amazon machine learning generates predictive or machine learning models.

Microsoft provides easy data access facility to the clients without hard code connection information (like, subscription ID and token authorization) using Azure machine learning datastores [6,16]. When a client registers azure storage solutions as a datastore, then azure machine learning will be responsible for creation and registration of the necessary datastores. The datastore stores all the related connection information of the corresponding storage account.

Table 1 compares some popular smart storage solutions which use machine learning approaches to make them intelligent. Archivist [30] is a mechanism to select the storage device unit from solid-state drives (SSD), conventional hard disk drives (HDD) and non-volatile RAM (NVM) for file placement. Adaptive Resource Management (ARM) system developed by [26] is another machine learning based technology which enables object based storage cluster to be a self-managing and self-adaptive system. On the other hand, AIOps (artificial intelligence for IT operations) platforms designed by Gartner [9,17] is a combination of big data and machine learning. One can use AIOps to detect the failure parameter of cloud object storage service at IBM (IBM COS [13]). Smart object storage system (SOSS) [45] is a machine level storage system which considers machine level technologies to make it intelligent.

Our proposed approach in this article differs in two ways from the existing approaches as shown in Table 1. Here we develop a machine learning unit (called as classification engine). It works as a part of the cloud storage system unlike the discussed ones that provide a platform where clients can build their own machine learning models using the facilities of that storage system to make the storage system presuming services more advanced. Secondly, the objective of the proposed classification engine unit is different from the discussed ones and also the working process and architecture are varied with the demands of associated applications.

In this article, our main aim is to predict the storage space for the dataset presented in client's write query request where multiple database models are involved. In the research world, there have some popular recent research work which uses traditional strategy rather than machine learning for handling multiple databases. Such as PolyglotHIS [15] uses the query mapper to divide a query into subqueries for accessing data from multiple data storage systems which come from different database models. Data mapper [39] is used to map the dataset to the corresponding database system by considering data object relationship with database system. polyglot persistence mediator (PPM) [32] selects the database at run time on the basis of tenant's defined information where it

consists with the database schema with the nodes values, SLA information with the functional (ACID transactions, Joins, updation etc.), continuous (e.g., availability, latency, throughput etc.), and non-functional (e.g., scalability, elasticity, durability, replication etc.) parameters as the inputs. Wiki-Health platform [18] considers the ontology engine is used to describe the storage systems by referring to the semantic information of storage systems. These mentioned four research work focus on different direction for working with multiple database models. However, none of the above works is focused on automatic storage space prediction using machine learning. The approach proposed in this work is thus novel to the best of our knowledge.

RSoS system overview

For doing the experimental analysis, “object based schema oriented data storage system” (RSoS System) platform is chosen due to its data storage procedure and architecture. Before drawing the desired unit for calculating storage space automatically, here we discuss the related information of RSoS system. The detailed information of RSoS system is distributed in two articles. The first article [23] presents a conceptual view of RSoS system and the second article [24] discusses the detailed architecture, functional prototype, and comparative analysis. The abbreviation of this storage system is shown in [24].

Object storage procedure: RSoS system treats data as an object. This object makes encapsulation of data with three overlays, viz., account, container and object. The account is used to identify each user uniquely. The container is the user-defined identifier. The information of these overlays is passed to the RSoS system by the query.

The main task of data storage procedures is to generate “object storage space”. To create object storage space it is required to know account, container, local schema and database. These information are object-dependable. After receiving account-container information with an object, the next task is to find out the database and local schema from this object received as a data query. At this time it takes the help of “hyper-graph data model”, the contents are human written.

Hyper-graph data model makes up with three parts, viz., node, hyperedge, hyperhyperedge. Node is the representation of the attribute, account and status. The attribute is presented in data passed by the query. Status informs that this is key or non-key. The link between two edges is hyperedge and a combination of different hyperedges are hyperhyperedge. Hyperhyperedge holds information of the database, i.e., database name and device URI where this database is presented. All of this information is present in graphml.xml of RSoS system.

Table 1 Comparative analysis on systems overview of ML based storage solutions

Name	Objective	Challenges	Working principle	Disadvantages
Archivist [30]	Selection of data placement location in storage system for reducing file access latency	Run-time status and diverse properties of both data and storage systems	ML based mechanism to optimize access pattern prediction for data file placement at runtime in storage system	Data movement cost and data storage capacity as negligible by using HDD
ARM system [26]	Develop a self managing and self adaptive storage cluster	In the presence of diverse workload patterns and resource bottlenecks	ML based system adaptation technique to track hotspot performance to take correct decision for optimizing future performance	Consider each and every node in the cluster has same configuration.
AIOps [17]	Design an automated system to discover failure parameters in IBM COS	Detect, predict and prevent failures and performance slowdowns that could impact users	A data driven ML based platform to find out the root cause of performance slowdown by data analysis	Background jobs, load imbalance, and resource bottleneck have not been considered which may effects the system performance
SOSS system [45]	Design an intelligent storage system to satisfy the changing attribute pattern	Object, composed of application data and data attribute, is considered as storage unit to accomplish the predicting job	Adopting the storage system to achieve intelligence such as pattern recognition, predicting object properties and an adaptive cache replacement policy	Complex nature of the storage system makes it difficult to handle
RSoS system [23]	Storage space prediction for health data based on it's data structure	Handling big data variety issue of health data	ML based framework analyse the system behavior corresponding to input data and match the related storage device	Background job effects the value of system monitoring output when an input hits the system

Architecture: This storage system has three-layer architecture, viz., client API, object manipulation unit and storage device unit. Through the client API, users can communicate with RSoS system. RSoS system receives three queries, viz., read, write and delete in JSON format.

Storage device unit consists of several workstations with huge disk size. These workstations run by database management systems. To support the health data variety property, more than one DBMS is involved. RSoS system uses three database management systems, viz., Hadoop distributed file system, Cassandra, MongoDB. This unit is enclosed with a RESTful web service. Each individual database management system has separate unique URI.

The sub-components of RSoS system placed between client API and storage device units receive instruction from client API as query and decides which service needs to be provided. To provide the instructed service, it needs to know the corresponding object storage space. Here, it takes help from graphml.xml and the respective query. At last, it generates the corresponding query and URI to hit the particular storage device unit.

Classification engine framework

The data classification engine uses machine learning technologies on a set of training data to acquire knowledge about this dataset. This knowledge helps predict the property of a similar type of dataset. To do such operations, a set of units is works parallelly, so we called it engine. The ultimate goal or work process of it varies with the system demands to which it is associated.

Overview

Because of the application dependency, RSoS cloud storage platform is used to describe the framework of the desired classification engine. The overall architecture of the designed classification engine unit associated with the RSoS system is shown in Fig. 1. This classification engine framework is built up with four major components: (1) server monitoring unit, (2) artificer, (3) decision maker and (4) computation unit.

Server monitoring unit collects the information of cloud storage server (e.g., RSoS system) subparts (viz., load, memory, input/output, processor) when a write query hits this system. Artificer is responsible to run the data type prediction jobs. If it finds the storage space information of the incoming data is present in the hypergraph data model then it stops

Fig. 1 Architecture of RSoS system for automatic storage space prediction

the job otherwise it continues further operations. The sub-components (viz., feature selection approach and classifier) of decision maker are dedicated for predicting the data type of this arrived dataset. Computation unit collects the corresponding information of storage space (viz., database name, storage device URI, attribute names) for this dataset which are required to update the metadata (e.g., hypergraph data model).

This classification engine is connected with three components of RSoS system, viz., Client API, storage device API generator and hypergraph data model. The received query from consumers is passed to this classification engine unit through the client API. It also acts as a communication medium between RSoS and consumers. The classification engine informs storage device API generator about the storage space of the dataset placed within the arrived query. This storage device API generator then converts this query to the device reachable query and generates the corresponding storage device URI which varies with the storage space value and query type (viz., read, write, delete). At last, the classification engine is responsible to update the hypergraph data model with the generated storage space information which is used further by classification engine to find out storage space for similar type datasets.

Workflow

To perform the prediction of object storage space of the incoming write query, the classification engine follows a workflow model presented in Fig. 2 by considering RSoS cloud storage platform. The proposed classification engine workflow model is designed for predicting the storage space of the arrived client's write request which hits the storage system at that time. For the experimental purpose healthcare dataset is considered that consist with mainly three types of data, viz., document, file, and sensor. We find few differences between sensor, file, and document data requests. Sensor data is tuple structured dataset consists of data value which is very small in size compared to others and the time gap between two data requests is a few seconds or few minutes. On the other hand, document type data is smaller in size than file type data and consists of a set of key-value pairs and the time gap may vary from a few minutes to hours or even days. File type data is larger or same in size compared to document type data request. Also, between two sensor type data requests, database servers can receive document type or file type data requests. No doubt these characteristics make an effect on the system properties (i.e., input/output, network, CPU, and memory) values.

Fig. 2 Workflow model of proposed classification engine

At the initial stage, when write query hits the classification engine, two parallel tasks are performed, viz., monitor the server status by the server monitoring unit and check the contents of graphml.xml file by Artificer. Graphml.xml is the written format of the hypergraph data model in the file structure.

The server monitoring unit observes the server status value related to resource performance metrics (e.g., memory, processor, Input/output and load) [31] when a JSON query hits the RSoS server. The generated server-status value set corresponding to the same query is not the same every time but the fluctuation value is very negligible in different queries. These characteristics help us decide the class of the unknown query.

Artificer acts as an investigator to find out the object storage space of coming write query in graphml.xml file. If it achieves the desired aim then the classification engine stops its tasks. Otherwise, it starts decision maker activity to move forward. Decision maker is the combination of two activities, learning technique and classifier.

The classifier communicates with learning technique to decide the class value (i.e., data type) of the dataset, present in this unknown write query. This process goes through certain steps. Firstly, the learning technique accomplishes its jobs offline. It trains a set of server monitoring datasets whose class values are known and finds out a set of features for doing the classification. Here, we consider three class values: (1) file, (2) sensor, (3) document. Secondly, Using the selected features, the classifier decides the class value of received monitoring dataset in real-time.

The computation unit generates the object storage space for the unknown arrived write query. For generating object storage space, the object key-value, attribute names and database name are needed to be known. The “received time” value is varied with every upcoming query. So, by default,

time is considered as an object key. After receiving the write query, the computation unit can know the attribute names. From the class value, collected from decision maker, the database name is known. Each class value corresponds to a database name such as “HadoopDFS” for “File”, “Cassandra” for “Sensor” and “MongoDB” for “Document”. The computation unit updates the graphml.xml file by the corresponding predicted object storage space information and at the same time, it sends this information to the Artificer.

Experiments

When a dataset hits the RSoS server, it comes as a query request. These queries are of three types, viz., write, read and delete. Here, for the experimental purpose, the write query is considered.

During the experiments, three types of data are used viz., sensor, document and file, as shown in Table 3. Real health sensors send tuple structured datasets within write query, presented as sensor type data. These data come in an infrequent manner. Here, we consider five types of sensors, Temperature, ECG, Pulse, Airflow and Oxygen. Consumers are mainly health workers. Consumers send documents or file type datasets to the RSoS server in write query format. For practical simulation, three consumers, who send medical images (as file type), patient’s photo (as file type) and patient’s details information (as document type) are considered individually.

Table 2 shows a comparison of different technical parameters between our RSoS system and other four machine learning based storage solutions. It can be seen that none of the solutions handles more than one kind of data format as considered datasets and three different types of database models under a single platform like RSoS. Also, two recent

Table 2 Comparative study on technical parameters of ML based storage solutions

Systems	Data store used	Data considered	Technology considered	Benchmark used for experiment
Archivist [30]	HDDs, SSDs, and NVM	Log files, Configuration files	LRU, baseline and optimized algorithm	Filebench Benchmark including WebProxy, WebServer, Fileserver, Varmail
ARM system [26]	Ceph [43]	Data file	Linux SAR utility, Ceph default, Baseline, DLR, ARM	NSF Cloud's Chameleon testbed, COSBench Benchmark version 0.4.2, sysbench CPU Benchmark
AIOPS [17]	IBM Cloud Object Storage [13], HDFS [3]	Log file (access logs, connectivity logs)	Apache search, Spark, Apache Grafana, Slack	N. A.
SOSS system [45]	Object storage system (OSS) [46]	Application data, attribute data	Switch, RAM, CPU, EPROM, Disk	N. A.
RSoS system	Cassandra, MongoDB, HDFS	File data, document data, sensor data	Ganglia, RestAPI	N. A.

technological considerations, viz., Ganglia and RestAPI have been used to setup the experiments by RSoS while others are used classical technologies like RAM, LRU, DLR etc. Some storage solutions use benchmark to set up their experimental parameters to measure their task performance. For our case, parameters are chosen according to the components selected for the classification engine unit. These elements make us rethink about the designing of experimental components.

RSoS server monitoring

To observe the experimental scenario, Ganglia 3.6.0 [21] plug-ins are used. As an example, we run the scenario for 1 h which generates 50 KB of data as sample data for the experimental execution. However, shorter or longer runs on the scenario will generate a smaller or larger amount of sample data but the ratio (1:1:1) of the involved three types data (file, document, and sensor) is same. Also, the training model is fixed here. The effect of training data size variation on the performance of the classification engine (accuracy, precision, and recall) is found to be very small provided we have good representation of all data types in the training set. RSoS system runs in our local workstation in this experiment. Ganglia starts monitoring the server status of RSoS system in four parts, viz., Processor, Memory, Input/Output and Load [31] from the first query request and stops at the last query request. Each and every time when ganglia finds a query request, it generates the corresponding monitored value matrix.

The dimension of this matrix is (*query number* $\times$ 18). This 18 is the number of properties which shows the corresponding status values of system sub parts of the RSoS server node (memory, input-output, load, CPU) of that query. Each query has different sets of property values but the property names are same in every query. It acts as the set of properties

or features and helps the classification engine distinguish the dataset according to their types. If we analyze the property value set from query to query, we can see a minor difference.

Monitored dataset processing

After collection of RSoS server monitoring dataset, we need to make the classification engine learn so that it can predict the object storage space of incoming data query. For experimental reason, we divide the collected dataset into two parts, training dataset and test dataset. Both training and test datasets are in .csv format and one more column, representing the class is added.

Three types of write queries are used for each type of data viz, file data, sensor data, and document data. After the task of monitored dataset processing, a set of rows with 18 columns are generated corresponding to each type of write query. One more column is added with this data as 'class'. The corresponding data type of the write query is written under the class attribute. For example, for file data, class value is file, for sensor data, class value is sensor, and for document data, it is document. For training dataset, class value is known but for test dataset it is to be predicted.

Feature selection

Feature selection is the first step of the classification engine. Here, it is needed to select such features which help to distinguish different classes well. These techniques totally run on the training dataset. There are 18 property values but each property value is not equally necessary to differentiate the classes. For this reason, we take the help of some feature selection algorithms to select the relevant properties.

Table 3 Example of query set for different users

Data source	Example	Query structure
Sensor	Pulse	<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "patientid": "anita123", "pulse": "74", "Time": "Tue May 08 16:06:53 GMT+05:30 2018" }}</pre>
	Temper-ature	<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "patientid": "anita123", "temperature": "33.2", "Time": "Tue May 08 17:06:53 GMT+05:30 2018" }}</pre>
	Oxygen	<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "patientid": "anita123", "spo2": "99", "Time": "Tue May 08 18:06:53 GMT+05:30 2018" }}</pre>
	Airflow	<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "patientid": "anita123", "airflow": "12", "Time": "Tue May 08 19:06:53 GMT+05:30 2018" }}</pre>
	ECG	<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "patientid": "anita123", "ecg": "10", "Time": "Tue May 08 20:06:53 GMT+05:30 2018" }}</pre>
		<pre>{ "container": "U009_ecg_12348", "account": "U101", "operation": "Store", "datagroups": { "familyHistory": { "father": { "history": ["High blood pressure", "Diabetes"] }, "mother": { "history": ["Diabetes", "Asthma"] }, "generalObservation": ["In pain", "Looks unwell", "Appears very thin"], "habit": ["Smoking", "Tobacco chewing"], "medicalHistory": ["High blood pressure", "Stroke", "Diabetes"], "patientDemographicInfo": { "address": "xyz", "age": "31 years, 25 days", "email": "xyz@xyz.in", "gender": "male", "occupation": "business", "patient_id": "6", "patient_name": "xyz", "ph": "xyz", } } } }</pre>
Consumer1	Patient Details	

Table 3 continued

Data source	Example	Query structure
Consumer2	Patient Medical Image	<pre>{ "account": "U104", "operation": "Store", "container": "ecg", "datagroups": { "id": "BenchmarkGain1", "remotefiledata": "xyz....", "fileextension": "png", } }</pre>

In this experiment, we apply such feature selection algorithms which select features (or properties) based on label information. Here, three feature selection algorithms are used from three different sub categories, (1) Fisher score, (2) F-score and (3) L1L21. Fisher score [44] is a supervised similarity-based feature selection algorithm which is fast and inexpensive. Here, each feature holds the score value individually using Fisher criterion and higher score value determines the better relevant feature compared to others. At the same time, it selects those properties whose values are similar for the same class and dissimilar for different classes. F-score [5] is a supervised statistical feature selection criterion which is based on the correlation between the input variables. It is fast and efficient because this process selects those properties which have a strong statistical relationship with the target variable. L1L21 [19] is the supervised sparse learning-based feature selection algorithm. It selects properties using sparsity-induced regularization terms by making feature coefficients small or zero. This feature selection algorithm considers the concept of multiple classes or multiple targets at the time of feature selection. This model is cheaper and can perform on the smaller set of input variables.

Dataset classification

In the next step, our target is to divide the collected dataset on the basis of their data types or class values. For the training dataset, the class values are known. Three data types, viz., sensor, file and document represent three class labels. Classifier learns from the training dataset which is divided into three classes. This knowledge is applied to predict the class value of test dataset.

We consider three supervised classification algorithms to do the experiments. These are K-nearest neighbors (K-NN), support vector machine (SVM), and neural network (NN).

K-NN [7] follows the “closest point search” mechanism which assigns unknown data points to the class in which the majority of their K nearest neighbors fall. It is a popular supervised non-parametric classifier used for prediction. In our experiment 5 is chosen as the K’s value because the odd value of K shows a clear majority prediction output.

SVM [33] is also known as “Discriminative Classifier” where it calculates the optimal hyperplane separating the classes from the training dataset and using this hyperplane finds out the class of unknown data points. It performs efficiently on a multi-dimensional dataset. Here, three types of SVM are used, linear SVM, radial basis function (RBF) SVM, polynomial (Poly) SVM. In this experiment, linear SVM, RBF SVM and Poly SVM consider the value of C (regularization parameter) as 1.0. Whereas, RBF SVM and Poly SVM consider 2.0 as the gamma value and poly SVM assigned 2.0 as the degree value.

Neural network [37] follows the human mind mechanism. From the training dataset it generates some rule or functionality using a multi-layer way and applies these rules to find out the class of unknown data points. It is able to work with incomplete information. According to our experiment, a neural network uses three hidden layers with thirty neurons in each layer. The used activation function is ReLU and weight optimization algorithm is Adam for hidden layers. Two hundred epochs are used here.

Dataset prediction

After collecting the knowledge of classification, the next target of classification engine is to predict the data type of incoming dataset. For this purpose, SVM, neural network and K-NN classifiers are used. The dataset prediction procedure goes through in certain steps.

In the first step, the classifiers identify the class value (sensor, file and document) of the dataset which is present in the incoming query request. After that, the classification engine generates the corresponding object storage space of this query and writes down the related information into the hypergraph data model if it is not present there.

The task of deciding the probable object storage space structure corresponding to the predicted class value is done by the computation unit of the classification engine shown in Fig. 2. To generate the object storage space, the computation unit needs to have little information such as, account value, list of attributes of dataset, name of the storage device and the corresponding storage device URI according to the structure of the hypergraph data model shown in Listing 1. Three

Table 4 Comparative study on experimental parameters of ML based storage solutions

Systems	ML framework used	ML algorithm used	ML algorithm compared	Input parameter	Comparison parameters
Archivist [30]	Offline classifier	Neural Network (16-Neuron, 32-Neuron)	Naive Bayes Classifier and Support Vector Machine	Offsetw, offsetr, lengthw, lengthr, spanw, spanr	Classifier accuracy, file access latency
ARM System [26]	Reinforce-ment learning	TensorFlow Neural Network	N. A.	System-level performance metrics of Ceph Storage such as state and reward information	I/O and network interference (Average read and write response time)
AIOps [17]	Big data analysis platform	Smart group by method, multivariate anomaly detection algorithm, feature isolation technique, map-reduce programming model	N. A.	Operation type, bucket and object names of access logs, HTTP return code, start and end times of the operation, and various latency statistics	Anomaly score (Z-score) of the aggregated latency
SOSS system [45]	Object classifier	Pattern classification	N. A.	Object attribute, access-based object attribute (including application assistance and existing user input), inter-object relationships	N. A.
RSoS system	Classification engine	K nearest neighbor (K-NN) classifier	Linear SVM, Poly SVM, RBF SVM, neural network	proc_run, load_five, disk_free, proc_total, load_one, mem_cached, bytes_in, bytes_out, mem_free, mem_buffers, pkts_in, load_fifteen, cpu_idle, cpu_user, cpu_nice, cpu_system, cpu_wio, pkts_out	Classifier accuracy, precision, recall

storage devices are used in this article, viz, Cassandra, MongoDB, and HadoopDFS. These storage devices consist of their own database model. According to the supporting characteristics, Cassandra is able to handle large scale structured data, mongoDB for document data, and HadoopDFS for file data. In this experiment, some predefined constraints are followed. Sensor data has three attributes: patientid, time, and data value; document data has time and document attributes; and file data has id and file attributes. As mentioned in Listing 1, each storage device has an unique URI with their URI values. The task of the computation unit is to collect the above information against the predicted class values and write them into the graphml.xml file as a form of object storage space value.

Table 4 shows that different storage technologies use individual machine learning approaches and parameters to improve the performance of their storage solutions. It can be seen that mainly classification based technologies got

much attention to this kind of prediction task. The sharpness of the RSoS system also depends upon the performance of the designed classification engine unit. Hence, three performance parameters of the classification engine are considered that provide better analysis comparing with the other storage technologies because they do not bother about the performance of the building elements. Input parameters or selection of feature set are the important parameters for any kind of classification task. It can be seen from the table that RSoS system is using many important features as input parameters compared to any other storage solutions. This helps design the classification engine for RSoS system. For this important task, we need to consider a suitable feature selection algorithm and the corresponding classifier. As per the results shown in the next section, it can be seen that L1l21 feature selection algorithm and K-NN classifier are the most suitable components for this classification engine.

Results and discussion

Two main things are needed to design the classification engine. One is a feature selection algorithm and another is a classifier. So, we make a comparative study of feature selection algorithms and classifiers to find out suitable combination of feature selection algorithms and classifiers for this classification engine. The results are shown in Table 6.

According to the results, the input of the feature selection algorithm is the generated output of ganglia. When RSoS server receives any query request then it passes through ganglia and the feature selection algorithm selects some features from these outputs. Here, top 5 features are selected from 18 features. The selected feature sets by the three mentioned feature selection algorithms are presented in Listing 9 separately.

A selected feature set contains the attributes of the data set that are chosen to be important by a feature selection algorithm for distinguishing the different classes of data types. Among the set of selected features by Fisher score and F-score algorithm, `proc_total` and `proc_run` signify the processor status such as total number and running processors, respectively. The features `load_five` and `load_one` determine the load status during last 5 and 1 minutes, respectively. Also the feature `disk_free` denotes the total free disk size. On the other hand, the selected feature set by L1l21 algorithm includes `mem_cached`, `mem_buffer` and `mem_free` which describe memory status such as cache, buffer and free memory size, respectively along with `bytes_in` and `byte_out` for representing the number of input and output bytes, respectively. These features are used by the classifiers to identify the class value of the test dataset.

```
Fisher score: {proc_total, proc_run, load_five, load_one, disk_free}
F-Score: {proc_total, proc_run, load_five, load_one, disk_free}
L1l21: {mem_cached, mem_free, mem_buffers, bytes_in, bytes_out, }
```

Listing 9 Selected Feature-set by the specified feature selection algorithms

The results are shown in two separate tables. Table 5 is for the test dataset where it holds only feature values. Table 6 consists of original class values with the predicted class values. Pointer column of Tables 5 and 6 are used to make bridge between these two tables. If we analyse Tables 5 and 6 then we find out some rows are marked and some are unmarked. Input feature values of marked data rows are present in training dataset.

The class column of Table 6 represents the original class values of the input dataset. In Table 6, we present outputs of three feature selection algorithms, viz., Fisher Score, FScore and L1l21 after applying 3 different SVMs (linear SVM, Poly SVM, RBF SVM), Neural Network and K -NN classifier as `Pfisher_Linear`, `Pfisher_Poly`, `Pfisher_RBF`, `Pfisher_Neural`, `Pfisher_K-NN`, `PFScore_Linear`, `PFScore_Poly`, `PFScore_RBF`, `PFScore_Neural`, `PFScore_K-NN`, respec-

tively. For L1l21 feature selection algorithm, we only consider Linear SVM, Neural Network and K -NN classifier represented as `PL1l21_Linear`, `PL1l21_Neural` `PL1l21_K-NN`, respectively.

Accuracy, precision and recall are used to judge the performance of the designed classification engine [14,36], shown in Table 7. These metrics are measured based on the test data prediction. The values of these metrics are varied by the applied algorithms (the combination of feature selection algorithm and classifier). Accuracy measures the correctness of prediction. precision is used to measure the positive predictive value and recall for calculating the true positive rate. These two are computed based on the class items.

Decision Maker is the main backbone of the classification engine unit. The whole system may break down by the wrong components selection for the decision maker. Figure 3 presents the visualization of performance metrics values which helps to make a fair decision. Here, three sub-plots present the comparison of precision (Fig. 3a), recall (Fig. 3b), and accuracy (Fig. 3c) values between the 13 feature selection algorithms and classifier algorithms. Precision and recall are calculated based on three different data types (document, file, and sensor).

The pair of L1l21 feature selection algorithm and K -NN classifier give a 100% accuracy rate. Fisher-score feature selection algorithm gives a very low accuracy rate (13.33%) with a neural network classifier. The combination of the L1l21 feature selection algorithm and K -NN classifier has the highest precision and recall values, i.e., 100% on the respect of three class items.

Conclusions and future work

In this article, we have proposed an idea to convert any object based storage system like RSoS into an automated system by automatically identifying the storage spaces for the incoming queries. For this purpose, we have built a classification engine which is able to categorize data based on their structure format. The RSoS system stores data based on its structure by implementing object storage space, a combination of the database name, data attributes name and object key. The storage system considered here can accommodate three types of data (i.e., sensor, document, file) under a single storage platform. The association of this classification engine with the RSoS system enables it to predict the corresponding object storage space for the incoming queries without manual intervention.

The efficacy of the classification engine depends upon the performance of its two main building blocks which are a feature selection approach and a classifier. Three supervised feature selection algorithms from three categories, viz., Fisher score from similarity-based, F-score from statistical-based, L1l21 from sparse learning-based approaches have

Table 5 Example of feature values used for testing purpose

Pointer	proc_ run	load_ five	disk_ free	mem_ cached	pkts_ in	bytes_ in	bytes_ out	mem_ free	load_ fifteen	cpu_ idle	cpu_ user	cpu_ nice	mem_ buffers	cpu_ system	cpu_ wio	proc_ total	pkts_ out	load_ one
Test1	1	0.53	860.35	22568560	1.22	198.14	301.85	611476	0.67	82.2	14	0	617340	2.8	1	917	2.17	0.49
Test3	0	0.59	860.37	22487480	7.4	1221.02	939.57	587852	0.67	79.5	13.5	2.6	617972	3.3	1.1	842	7.97	0.41
Test5	1	0.7	860.37	22545304	0.85	122.24	221.25	350468	0.69	85.6	11.8	0	618544	2.2	0.3	835	1.65	0.63
Test7	3	0.91	860.37	22537512	0.72	79.77	163.7	301044	0.77	84.5	11.6	0	618992	3.3	0.7	831	1.37	1.26
Test9	1	0.62	860.37	22451576	0.75	94.33	176.4	277464	0.71	86.3	11.4	0	619500	2.2	0.2	830	1.45	0.28
Test10	0	0.56	860.37	22439988	1.62	334.98	299.58	308536	0.67	86.4	10.9	0	619700	2.6	0.1	832	2.27	0.41
Test11	0	0.52	860.37	22378904	1.15	146.03	276.75	251528	0.62	86.2	10.8	0	620436	2.4	0.6	829	1.93	0.36
Test13	0	0.45	860.37	22285800	1.55	277.24	335.15	289580	0.56	86.5	10.9	0	620920	2.4	0.2	838	2.15	0.48
Test14	1	0.56	860.37	22252728	0.65	85.9	120.95	307456	0.58	84.8	12	0	621160	3.1	0.1	831	1.5	0.71
Test15	2	0.48	860.37	22249184	1.03	182.88	261.73	263672	0.55	86.5	10.9	0.1	621416	2.5	0.1	830	1.73	0.39
Test17	2	0.45	860.37	22152916	0.8	144.44	205.16	289816	0.51	84.3	11.6	0	621892	3.3	0.7	832	1.55	0.5
Test19	1	0.55	860.37	22132340	0.92	198.04	215.01	306696	0.52	86.5	10.8	0	622380	2.6	0.1	835	1.67	0.71
Test21	2	0.35	920.50	3281772	29.1	41004.89	1061.7	212866440.41	0.41	90.7	5.9	0	589016	3.3	0.2	859	12.67	0.34
Test26	1	1.34	920.49	3234444	28.67	38324.01	715.25	213752401.04	0.4	68.8	16.5	0	604732	12.6	2.1	864	6.9	1.31
Test27	1	1.34	920.49	3234444	28.67	38324.01	715.25	213752401.04	0.4	72.2	14.5	0	604732	11.4	1.8	864	6.9	1.31
Test28	1	1.34	920.49	3234444	28.67	38324.01	715.25	213752401.04	0.4	72.2	14.5	0	604732	11.4	1.8	864	6.9	1.31

The row consisting of bold values also present in the training dataset.

Table 6 Comparative resultset shows the test output of feature selection algorithms with classification algorithm where Sensor = S, Document = D, File = F

Pointer	Class	Pfisher _Linear	Pfisher _Poly	Pfisher _RBF	Pfisher _Neural	Pfisher _K-NN	PFScore _Linear	PFScore _Poly	PFScore _RBF	PFScore _Neural	PFScore NN	PFScore_K - NN	PLI21 _Linear	PLI21 _Neural	PLI21 _K-NN
Test1	D	S	F	D	S	F	S	F	D	S	F	D	D	D	D
Test3	D	S	D	D	S	D	D	D	D	F	D	F	D	D	D
Test5	D	S	D	D	S	D	S	D	D	S	D	D	D	D	D
Test7	D	D	D	D	S	D	S	D	D	S	D	D	D	D	D
Test9	D	D	D	D	S	D	S	D	D	S	D	D	D	D	D
Test10	D	S	D	D	S	D	D	D	D	D	D	D	D	D	D
Test11	F	D	D	S	S	D	S	D	S	S	F	F	F	F	F
Test13	F	F	D	F	S	F	S	D	F	F	F	F	D	D	F
Test14	F	S	D	S	S	D	D	D	S	S	D	F	D	D	F
Test15	F	S	D	S	S	F	D	D	S	S	F	D	D	D	F
Test17	F	S	D	F	S	D	S	D	F	S	D	D	F	F	F
Test19	F	S	D	D	S	D	S	D	D	S	D	D	D	D	F
Test21	S	D	F	S	F	S	S	F	S	S	S	S	S	S	S
Test26	S	D	S	S	F	S	F	S	S	F	S	F	D	D	S
Test27	S	S	S	S	S	S	S	S	S	F	S	F	S	S	S
Test28	S	D	S	S	S	S	D	S	S	S	S	D	S	S	S

The row consisting of bold values also present in the training dataset.

Table 7 Performance matrix of the combined feature selection algorithm and classifier

Feature Selection Algorithm_Classifier	Accuracy (%)	Precision_Document (%)	Precision_File (%)	Precision_Sensor (%)	Recall_Document (%)	Recall_File (%)	Recall_Sensor (%)
Pfisher_LinearSVM	26.67	25	50	22.22	20	20	40
Pfisher_PolySVM	46.67	47.37	0	100	90	0	50
Pfisher_RBF SVM	70	75	75	64.28	90	30	90
Pfisher_Neural	13.33	0	0	16.67	0	0	40
Pfisher_K-NN	70	53.3	60	100	80	30	100
PFScore_LinearSVM	23.33	25	33.33	21.05	20	10	40
PFScore_PolySVM	46.67	47.36	0	100	90	0	50
PFScore_RBF SVM	70	75	75	64.28	90	30	90
PFScore_Neural	26.67	25	33.33	25	10	20	50
PFScore_K-NN	70	57.14	60	90.9	80	30	100
PLI21_LinearSVM	53.33	45.45	42.85	100	50	60	50
PLI21_Neural	63.33	52.94	50	100	90	30	70
PLI21_K-NN	100	100	100	100	100	100	100

Fig. 3 Comparative study of the performance matrix values of the components of the decision maker

been compared. Side-by-side, three classifiers, viz., SVM, K -NN, and Neural Network are used with the mentioned feature selection algorithms. After analyzing the results, L121 feature selector combined with K -NN classifier is found to provide the best service.

In future, our next aim is to extend the functionality of the classification engine in such a way that it can provide more prominent results in less time. Also, in the current situation, the training part of the engine is done offline which creates a deficiency in some situations. Whereas, if the system is trained with wrong information or the system predicts wrong data at test time, there is no chance to rectify on the later situation. We plan to explore the possibilities of alleviating these problems through online training when needed.

Author contributions The framework and experimental dataset are design by A.S. Mondal. The preparation of experimental platform is done by A. Mukhopadhyay. The overall article preparation is done under the observation of S. Chattopadhyay. Later, all the 3 authors have worked together on it to finalize the menu script. The submitted manuscript is studied and approved by all.

Funding Not applicable.

Declarations

Conflict of interest We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and material The dataset is given in the paper as paper content.

Code availability Code is uploaded and available in the GitHub and will be made available for public after the acceptance of the article, if required.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bahrami M, Singhal M (2015) The role of cloud computing architecture in big data. In: Information granularity, big data, and computational intelligence. Springer, pp 275–295
- Bisong E (2019) Google cloud machine learning engine (cloud ml). In: Building machine learning and deep learning models on Google Cloud Platform. Springer, pp 545–579
- Borthakur D (2008) Hdfs architecture guide. Hadoop Apache Project 53
- Cassandra. <http://cassandra.apache.org/>
- Chen Y-W, Lin C-J (2006) Combining svms with various feature selection strategies. In: Feature extraction. Springer, pp 315–324
- Collins E (2014) Big data in the public cloud. *IEEE Cloud Comput* 1(2):13–15
- Cunningham P, Delany SJ (2007) k-nearest neighbour classifiers. *Mult Classif Syst* 34(8):1–17
- forcepoint (2019) Forcepoint advanced classification engine (ace). https://www.forcepoint.com/product/add-on/advanced-classification-engine-ace?utm_source=Websense&utm_medium=Redirect&utm_content=websense-advanced-classification-engine%3Fcmpid%3Dslblog. Accessed 19 Nov 2019
- Gartner (2020) Aiops (artificial intelligence for it operations). <https://www.gartner.com/en/information-technology/glossary/aiops-artificial-intelligence-operations>. Accessed 29 June 2020
- Giudice O, Paratore A, Moltisanti M, Battiatto S (2017) A classification engine for image ballistics of social data. Springer, pp 625–636
- Han PKJ, Klein WMP, Arora NK (2011) Varieties of uncertainty in health care: a conceptual taxonomy. *Med Decis Mak* 31(6):828–838
- Herbrich R (2017) Machine learning at amazon. In: WSDM, p 535
- IBM (2020) Ibm cloud object storage. <https://www.ibm.com/cloud/object-storage>. Accessed 29 June 2020
- Japkowicz N (2006) Why question machine learning evaluation methods. In: AAAI workshop on evaluation methods for machine learning, pp 6–11
- Kaur K, Rani R (2015) Managing data in healthcare information systems: many models, one solution. *Computer* 48(3):52–59
- Klein S (2017) Azure data factory. Apress, pp 105–122
- Levin A, Garion S, Kolodner EK, Lorenz DH, Barabash K, Kugler M, McShane N (2019). Aiops for a cloud object storage service. *IEEE*, pp 165–169
- Li Y, Guo L, Wu C, Lee C-H, Guo Y (2014) Building a cloud-based platform for personal health sensor data management. *IEEE*, pp 223–226
- Liu J, Ji S, Ye J (2009) Multi-task feature learning via efficient l_2 , l_1 -norm minimization. *AUAI Press*, pp 339–348
- MacDonald A (2016) Phildb: the time series database with built-in change logging. *PeerJ Comput Sci* 2:e52
- Massie ML, Chun BN, Culler DE (2004) The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Comput* 30:817–40
- McKay C, Fiebrink R, McEnnis D, Li B, Fujinaga I (2005) Ace: a framework for optimizing music classification. In: ISMIR, pp 42–49
- Mondal AS, Chattopadhyay S, Neogy S, Mukherjee N (2016) Object based schema oriented data storage system for supporting heterogeneous data, pp 1025–1032
- Mondal AS, Neogy S, Mukherjee N, Chattopadhyay S (2019) Performance analysis of an efficient object-based schema oriented data storage system handling health data, pp 1–15
- Mongodb. <https://www.mongodb.org/>
- Noel RR, Mehra R, Lama P (2019) Towards self-managing cloud storage with reinforcement learning. *IEEE*, pp 34–44
- Openstack swift. https://www.swiftstack.com/docs/introduction/openstack_swift.html
- Palankar MR, Iamnitchi A, Ripeanu M, Garfinkel S (2008) Amazon s3 for science grids: a viable solution? New York
- PSIGEN (2019) Psigen releases accelerated classification engine. <https://www.psigen.com/?s=Accelerated+Classification+Engine>. Accessed 19 Nov 2019
- Ren J, Chen X, Tan Y, Liu D, Duan M, Liang L, Qiao L (2019) Archivist: a machine learning assisted data placement mechanism for hybrid storage systems. *IEEE*, pp 676–679
- Sarkar A, Pant K, Chattopadhyay S (2018) Drsq-a dynamic resource service quality based load balancing algorithm. In: International conference on computational intelligence, communications, and business analytics. Springer, pp 97–108
- Schaarschmidt M (2015) Gessert F (2015) Ritter Norbert (2015) Towards automated polyglot persistence. *Datenbanksysteme für Business, Technologie und Web (BTW)*
- Scholkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press
- Shah G, Voruganti K, Shivam P, Alvarez M (2006) Ace: classification for information lifecycle management
- Sinyor E, Rebecca CM, Mcennis D, Fujinaga I (2005) Beatbox classification using ace. *Music Information Retrieval, Citeseer*
- Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. *Inform Process Manag* 45(4):427–437
- Specht DF (1991) A general regression neural network. *IEEE Trans Neural Netw* 2(6):568–576
- Stonebraker M, Brown P, Zhang D, Becla J (2013) Scidb: a database management system for applications with complex analytics. *IEEE Ann Hist Comput* 15(03):54–62
- Trivedi K, Shah S, Srivastava K (2020) An efficient e-commerce design by implementing a novel data mapper for polyglot persistence. In: Advanced computing technologies and applications. Springer, pp 149–156
- Varonis (2019) Varonis, data classification engine. <https://www.varonis.com/products/data-classification-engine/>. Accessed 19 Nov 2019
- Veritas (2019) Veritas introduces new classification engine for intelligent data management across its portfolio. <https://www.veritas.com/news-releases/2017-07-25-veritas-introduces-new-classification-engine-for-intelligent-data-management-across-its-portfolio>. Accessed 19 Nov 2019
- websense (2019) Advanced analysis using real-time classification. https://www.websense.com/content/support/library/web/hosted/bsky_help/content_analysis.aspx. Accessed 19 Nov 2019
- Weil SA (2007) Ceph: reliable, scalable, and high-performance distributed storage. PhD thesis. University of California Santa Cruz
- Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V (2001) Feature selection for svms. In: Advances in neural information processing systems, pp 668–674
- Zeng L-F, Feng D, Qin LJ (2004) Soss: smart object-based storage system. In: Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826), vol 5. *IEEE*, pp 3263–3266
- Zeng L-F, Feng D, Wang F, Zhou K (2005) Object replication and migration policy based on oss, vol 1. *IEEE*, pp 45–49

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.