Computer Modeling in
Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.020128

ARTICLE

Optimizing Big Data Retrieval and Job Scheduling Using Deep Learning Approaches

Bao Rong Chang[1], **Hsiu-Fen Tsai**[2,*] **and Yu-Chieh Lin**[1]

[1]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan

[2]Department of Fragrance and Cosmetic Science, Kaohsiung Medical University, Kaohsiung, Taiwan

*Corresponding Author: Hsiu-Fen Tsai. Email: sftsai@kmu.edu.tw

Received: 05 November 2021 Accepted: 28 March 2022

ABSTRACT

Big data analytics in business intelligence do not provide effective data retrieval methods and job scheduling that will cause execution inefficiency and low system throughput. This paper aims to enhance the capability of data retrieval and job scheduling to speed up the operation of big data analytics to overcome inefficiency and low throughput problems. First, integrating stacked sparse autoencoder and Elasticsearch indexing explored fast data searching and distributed indexing, which reduces the search scope of the database and dramatically speeds up data searching. Next, exploiting a deep neural network to predict the approximate execution time of a job gives prioritized job scheduling based on the shortest job first, which reduces the average waiting time of job execution. As a result, the proposed data retrieval approach outperforms the previous method using a deep autoencoder and Solr indexing, significantly improving the speed of data retrieval up to 53% and increasing system throughput by 53%. On the other hand, the proposed job scheduling algorithm defeats both first-in-first-out and memory-sensitive heterogeneous early finish time scheduling algorithms, effectively shortening the average waiting time up to 5% and average weighted turnaround time by 19%, respectively.

KEYWORDS

Stacked sparse autoencoder; Elasticsearch; distributed indexing; data retrieval; deep neural network; job scheduling

1 Introduction

In recent years, the rapid growth of the amount of data coupled with the declining cost of storage equipment, the evolution of software technology, and the maturity of the cloud environment led to the rapid development of big data analytics [1]. When the amount of data is enormous, and the speed of data flow is fast, traditional methods are no longer for dealing with data storage, computing, and analysis in time, and excessively large-scale access will also cause I/O severe delay in a system. Faced with such explosive growth of large amounts of data, implementing tremendous distributed computing [2] with large-scale data clustering and type of not-only-SQL (NoSQL) storage technology has become a popular solution in recent years. Apache Hadoop or Spark [3] is currently the most widely known big data analytics platform in business intelligence with decentralized computing capabilities. Each of them is very suitable for typical extract-transform-load (ETL) workloads [4] due to its large-scale scalability and relatively low cost. Hadoop uses first-in-first-out (FIFO) [5] scheduling by default to prioritize jobs in the order in which they arrive. Although this kind of scheduling is relatively fair, it is likely to cause low system throughput. On the other hand, the response time of data retrieval using a traditional decentralized system is lengthy, causing inefficient job execution. Therefore, how to improve the efficiency of data retrieval and job scheduling becomes a crucial issue of big data analytics in business intelligence.

In terms of artificial intelligence development, deep learning [6] has rapidly developed in recent years. AlphaGo [7], developed by Google DeepMind in London, UK, defeated many Go masters in 2014. AI has become a hot research topic once again. The use of machine learning [8] or deep learning in various aspects of research and related applications is constantly prosperous. IBM developed a novel deep learning technology that mimics the working principle of the human brain, which can significantly reduce the speed of processing a large amount of data. Deep learning is a branch of artificial intelligence, and today's technology giants Facebook, Amazon, and Google focus on its related development for many innovations. In this era, the explosive growth of data will exist, and how to process and analyze a large amount of data effectively has become an important topic.

Nowadays, data retrieval and job scheduling research mainly focuses on using Hadoop and Spark open-source big data platforms in business intelligence systems to improve efficiency. Generally speaking, this study considers encoding and indexing technologies to realize fast data retrieval to increase the system throughput in big data analytics. Developing high-performance job scheduling in big data analytics reduces the time of job waiting for execution in a queue. The objective of this paper is to develop an advanced deep learning model together with a high-performed indexing engine that can beat the previous method, integrating deep autoencoder [9] (DAE) and Solr indexing [10] (abbreviated DAE-SOLR), over the speed of big data retrieval significantly. Based on deep learning model, this paper explored integrating stacked sparse autoencoder (SSAE) [11] and Elasticsearch indexing [12], abbreviated SSAE-ES, to create a fast approach of data searching and distributed indexing, which can reduce the search scope of the database and dramatically speeds up data searching. Implementing the proposed SSAE-ES can outperform the previous method DAE-SOLR with higher data retrieval efficiency and system throughput for big data analytics in business intelligence. On the other hand, this paper tried to exploit deep neural networks (DNN) [13] to predict the approximate execution time of jobs and give prioritized job scheduling based on the shortest job first (SJF) [14], which can reduce the average waiting time of job execution and average weighted turnaround time [15].

2 Related Work

2.1 Literature Review

The real-time data processing and analysis are getting higher and higher for big data analytics in business intelligence. To pursue better performance of data analysis, the job schedule plays a vital role in improving the performance of big data analytics. For example, Yeh et al. [16] proposed the user can dynamically assign priorities for each job to speed up the execution speed of the job. However, it lacks automatic functions to get the job started and must get permission from the user every time. Thangaselvi et al. [17] mentioned the user could import self-adaptive MapReduce (SAMR) algorithm into Hadoop, which can adjust the parameters recalling the historical information saved on each node, thereby dynamically finding slow jobs. But SAMR works based on the model established by K-means [18]. A K-means approach usually has specific assumptions about the data distribution, while DNN with hierarchical feature learning has no explicit assumptions about the data. Therefore, DNN can establish a more complex data distribution than K-means to get better prediction results.

Many studies even apply deep learning to the distributed computing nodes. For example, Marquez et al. [19] used the concept of deep cascade learning to combine Spark's distributed computing using multi-layer perceptrons to build a model that can perform large-scale data analysis in a short time [20]. However, there will still be a longer waiting time for job execution due to this model's, i.e., FIFO, lack of job scheduling optimization. Likewise, Lee et al. [21] gave data clustering with deep autoencoder and Solr indexing to significantly improve the query performance. However, the system lacks suitable job scheduling. Besides, Chang et al. [22] proposed the memory-sensitive heterogeneous early finish time (MSHEFT) algorithm for improving job scheduling. Unfortunately, when encountering the same size as data in the different jobs, it performs scheduling just like a FIFO, and nothing can be improved at all.

Nature-inspired optimization can give some searching advantages by using meta-heuristic-like approaches rather than deep learning models. L. Abualigah et al. proposed a novel population-based optimization method called Aquila Optimizer (AO) [23], which the Aquila's behaviors have inspired nature while catching the prey. Abualigah et al. [24] proposed a novel nature-inspired meta-heuristic optimizer, called Reptile Search Algorithm (RSA), motivated by the hunting behavior of Crocodiles. Regarding the arithmetic optimization algorithm used for searching the target data, Abualigah et al. [25] presented a comprehensive survey of the Internet of Drones (IoD) and its applications, deployments, and integration. Integration of IoD includes privacy protection, security authentication, neural network, blockchain, and optimization based-method. Instead, Time-consuming will be a crucial problem about the cost when people consider meta-heuristic-like approaches or the arithmetic optimization algorithm for searching in significant data retrieval issues.

2.2 Encoder for Data Clustering

Deep autoencoder (DAE) is an unsupervised learning model of neural networks. Its model architecture includes two fully connected feedforward networks, called encoder and decoder, which perform compression and decompression. After training AE, the user reserves the encoder and can input the data into the encoder. The encoder outputs a point located in the three-dimensional space, as shown in Fig. 1. Three-dimensional space divides itself into eight quadrants according to the X, Y, and Z axes, and the quadrants numbering from 1∼8 are inserted into the last column of the table, as shown in Fig. 2.

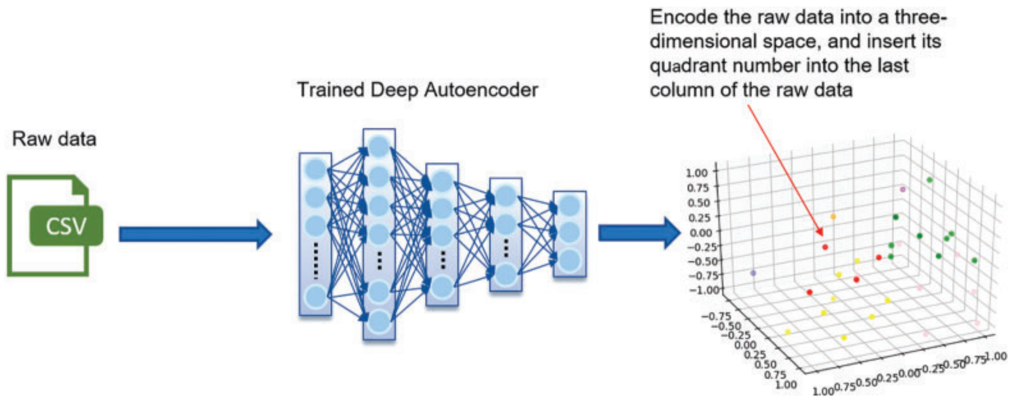
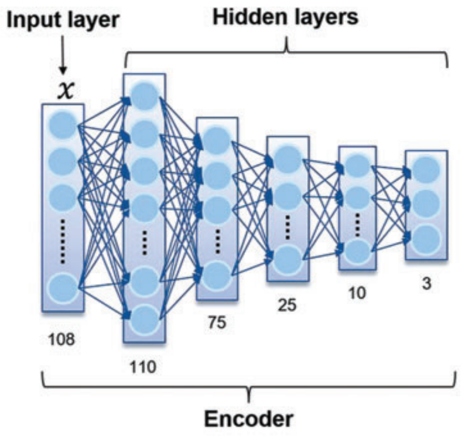
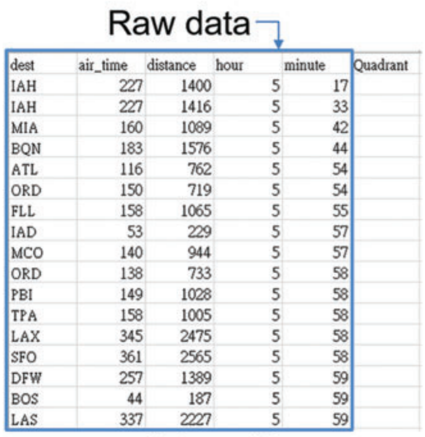

Raw data
CSV
Trained Deep Autoencoder
Encode the raw data into a three-dimensional space, and insert its quadrant number into the last column of the raw data

Figure 1: Data set mapping to 3D coordinates

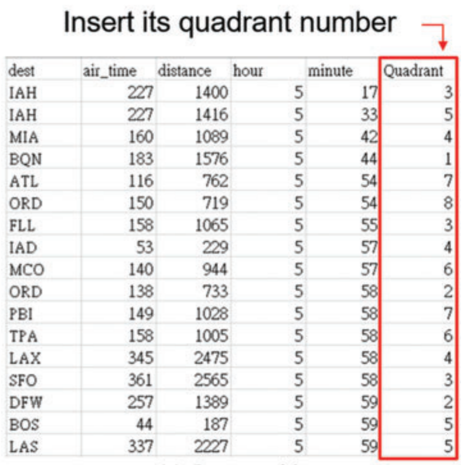


(a) Deep autoencoder architecture



dest	air_time	distance	hour	minute	Quadrant
IAH	227	1400	5	17	
IAH	227	1416	5	33	
MIA	160	1089	5	42	
BQN	183	1576	5	44	
ATL	116	762	5	54	
ORD	150	719	5	54	
FLL	158	1065	5	55	
IAD	53	229	5	57	
MCO	140	944	5	57	
ORD	138	733	5	58	
PBI	149	1028	5	58	
TPA	158	1005	5	58	
LAX	345	2475	5	58	
SFO	361	2565	5	58	
DFW	257	1389	5	59	
BOS	44	187	5	59	
LAS	337	2227	5	59	

(b) Input table


Insert its quadrant number

dest	air_time	distance	hour	minute	Quadrant
IAH	227	1400	5	17	3
IAH	227	1416	5	33	5
MIA	160	1089	5	42	4
BQN	183	1576	5	44	1
ATL	116	762	5	54	7
ORD	150	719	5	54	8
FLL	158	1065	5	55	3
IAD	53	229	5	57	4
MCO	140	944	5	57	6
ORD	138	733	5	58	2
PBI	149	1028	5	58	7
TPA	158	1005	5	58	6
LAX	345	2475	5	58	4
SFO	361	2565	5	58	3
DFW	257	1389	5	59	2
BOS	44	187	5	59	5
LAS	337	2227	5	59	5

(c) Output table

Figure 2: Encoder for data clustering

2.3 Stacked Sparse Autoencoder (SSAE)

Since training AE will activate the neurons of a hidden layer too frequently, AE may easily result in overfitting due to the degree of freedom being more considerable. A sparse constraint is added to the encoder part to reduce the number of activations of each neuron for every input signal. A specific input signal can only activate some neurons and will probably inactivate the others. In other words, each input signal of the sparse autoencoder cannot activate all of the neurons every time during the training phase, as shown in Fig. 3.

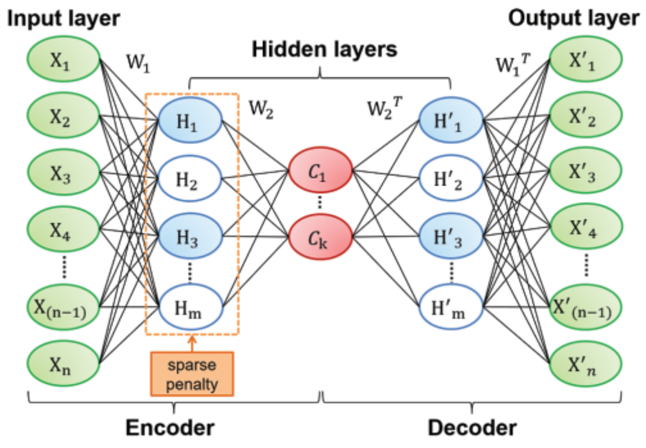


Figure 3: Sparse autoencoder model

The autoencoder with sparse constraints is called sparse autoencoder (SAE) [26]. SAE defines loss function on Eq. (1), where L represents the loss function without sparse constraint, *KL* stands for *KL* divergence [27], ρ denotes the expected activation degree of neurons in the network (if the activation function is a Sigmoid function, and SAE sets its value to 0.05, which means that most of the neural cell is not activated), and $\hat{\rho}_j$ is the average activation degree of the jth neuron. Here, people defined *KL* on Eq. (2) and use *KL* divergence to measure the similarity between the average activation output of hidden layer nodes and the sparsity ρ. Eq. (3) definds $\hat{\rho}_j$ the average activation degree on the training sample set in which m represents the number of training samples, and a_j and x^i stand for the response output of the jth node in the hidden layer to i samples.

$$L_{sparse} = L + \beta \sum_j KL(\rho||\hat{\rho}_j) \tag{1}$$

$$KL(\rho||\hat{\rho}_j) = \rho log\frac{\rho}{\hat{\rho}_j} + (1-\rho)\, log\frac{1-\rho}{1-\hat{\rho}_j} \tag{2}$$

$$\hat{\rho}_j = \frac{a_j}{m}\sum_{i=1}^{m}[x^i] \tag{3}$$

After training multiple SAEs in which the user tuned each SAE layer by layer, the user finally stacked multiple SAEs up and called it stacked sparse autoencoder (SSAE), as shown in Fig. 4. The encoded output of the previous stage acts as the input of the next stage. After the training of the first SAE in Stage 1, the user can obtain the first hidden layer with m neurons denoted Hidden 1. Likewise, by cloning Hidden 1 to the second SAE in Stage 2, the user can obtain the second hidden layer with n neurons denoted Hidden 2. Finally, the user stacks two SAEs to form a stacked sparse autoencoder (SSAE). Based on the ahead of the hidden layer, the current hidden layer can generate a new set of features. Hopefully, the user can get more hidden layers according to layer-by-layer training in the different stages.

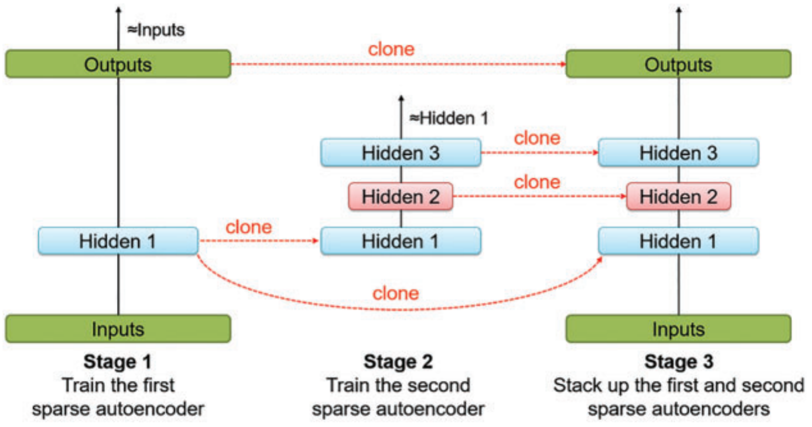


Figure 4: Layer-wise pre-training for two SAEs and stack them up

2.4 Data Retrieval by Indexing

According to the quadrant value of the last field, the data is divided into different files and then sent to Solr for data indexing. To cope with large-scale data retrieval, the longer it takes when the index volume is enormous, distributed indexes can speed up the search time. SolrCloud reduces the pressure of single-machine processing, and multiple servers need to complete indexing together. Its architecture is shown in Fig. 5.

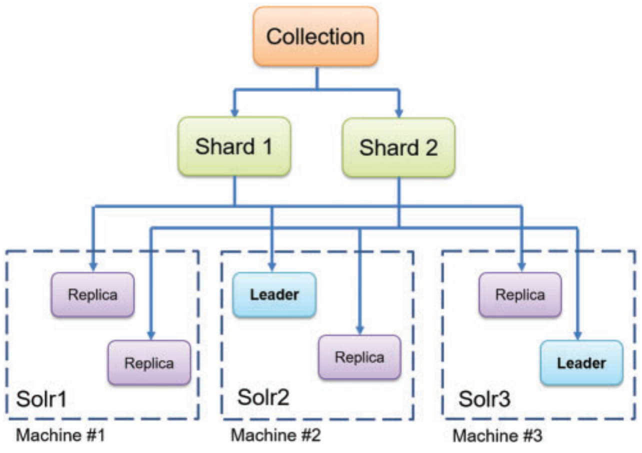


Figure 5: Solr architecture

After creating an index (collection) of data, Solr divided index into multiple shards, and each shard has a leader for job distribution, as shown in Fig. 6. When the replica completes the job, it will send the result back to the leader, and then the leader will send the result back to SolrCloud for the final result. The user can upload the file to any replica when the data is uploaded. If it is not the leader, it will forward the request to the leader of the same shard, and the leader will give the file path to each replica in the same shard. If the file does not belong to the same shard, the leader will transfer it to another leader in its corresponding shard. The corresponding leader will also give the file path to each replica in the same shard. The user manually uploaded the data to Solr after completing data clustering using the encoder process for subsequent node use. The flowchart of data preprocessing illustrates how to prepare the data clustering together with indexing, as shown in Fig. 7.

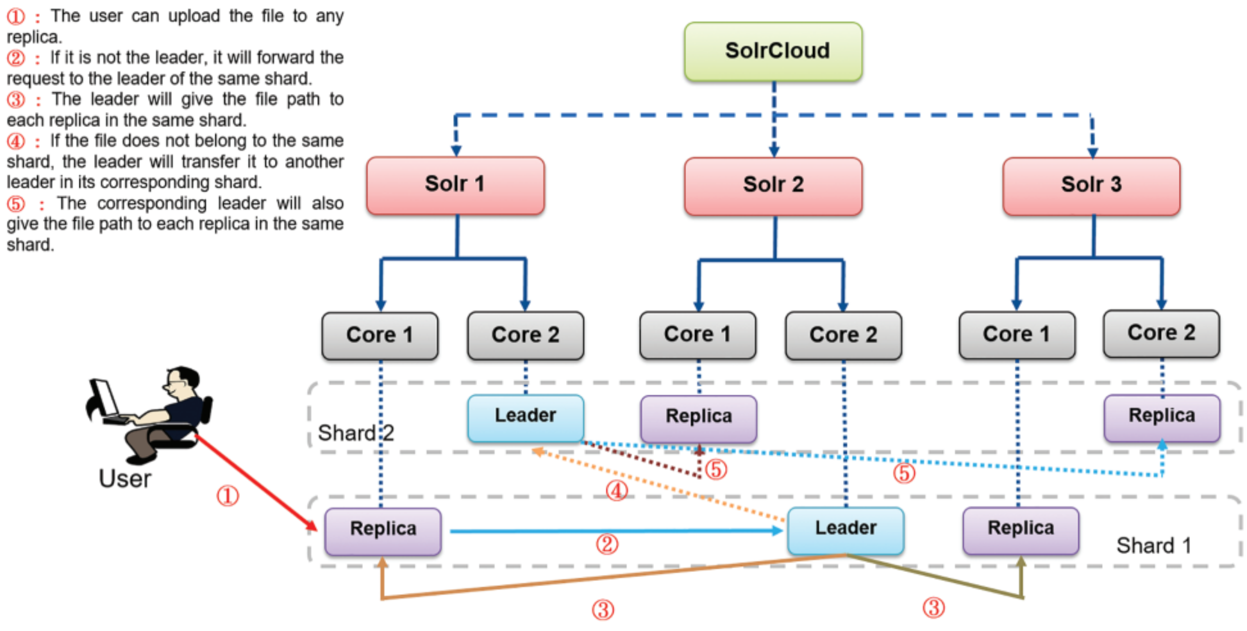

①: The user can upload the file to any replica.
②: If it is not the leader, it will forward the request to the leader of the same shard.
③: The leader will give the file path to each replica in the same shard.
④: If the file does not belong to the same shard, the leader will transfer it to another leader in its corresponding shard.
⑤: The corresponding leader will also give the file path to each replica in the same shard.
SolrCloud
Solr 1
Solr 2
Solr 3
Core 1
Core 2
Leader
Replica
Shard 2
Shard 1
User

Figure 6: SolrCloud architecture

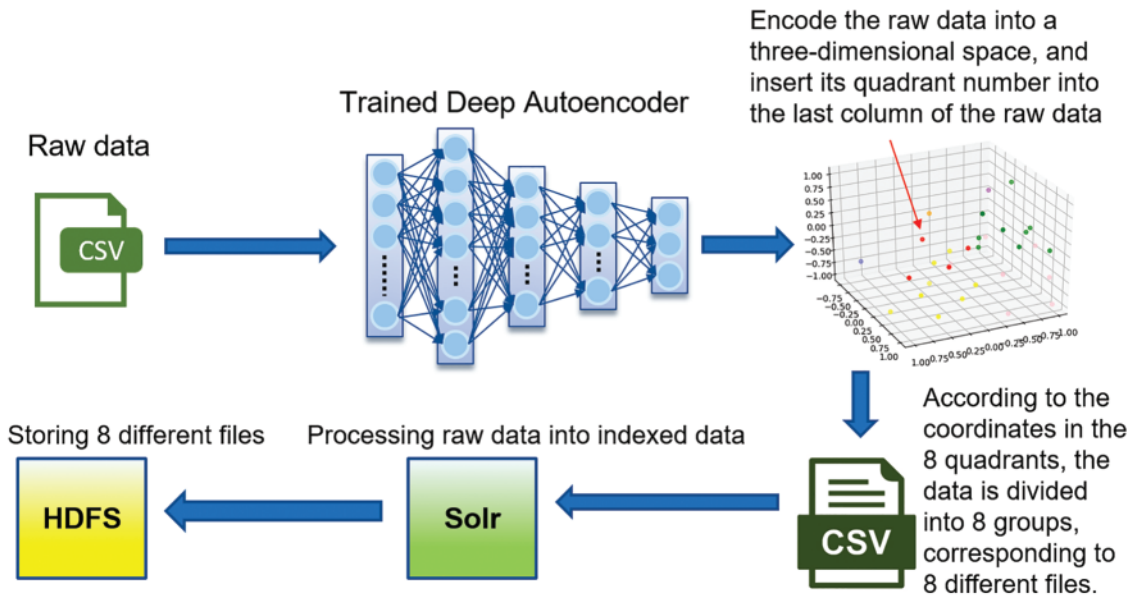

Raw data
CSV
Trained Deep Autoencoder
Encode the raw data into a three-dimensional space, and insert its quadrant number into the last column of the raw data
According to the coordinates in the 8 quadrants, the data is divided into 8 groups, corresponding to 8 different files.
CSV
Processing raw data into indexed data
Solr
Storing 8 different files
HDFS

Figure 7: Data preprocessing flow

2.5 *SQL Interface Selection*

Developing an automatic SQL interface selection lets users choose an appropriate SQL interface [28], such as Hive, Impala, or SparkSQL, for performing SQL commands with the best efficiency. The appropriate SQL interface selection is valid according to the remaining memory size given in a cluster node. After trial and error, we found two critical points labeled as L1 and L2, where L1 represents the amount of memory around 5 GB and L2 10 GB. In this study, we set each node the amount of memory with 20 GB. Thus it defines three memory zones ranging from 0∼5 GB, 5 GB∼10 GB, and 10 GB∼20 GB, as shown in Fig. 8. When the remaining memory size is less than L1, the system will automatically select the Hive interface to perform SQL commands. The system will use the Impala interface between L1 and L2. If it is more than L2, the system will choose the SparkSQL interface, as shown in Fig. 8. We note that the Impala and SparkSQL interfaces need more memory to run the job having a large amount of data. Given sufficient memory, SparkSQL with in-memory computing capability can run the best efficiency in big data analytics. On the contrary, the Hive interface can run any job at a low level of memory size but is time-consuming.

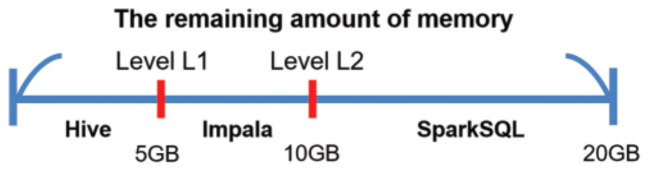


Figure 8: SQL interface selection

2.6 Job Scheduling Algorithm MSHEFT

Heterogeneous early finish time (HEFT) [29] is a heuristic way to schedule a set of dependent jobs onto a network of heterogeneous workers that takes communication time into account. HEFT works on a list out of the scheduling algorithm that establishes a priority list first. HEFT will assign each job to the appropriate CPU to follow the sorted priority list to complete the job as soon as possible. People modified the HEFT algorithm to the memory-sensitive heterogeneous early finish time (MSHEFT) algorithm [22] that first considers the job's priority, then the data size, and, finally, the remaining memory size to choose the appropriate SQL command interface automatically. The flow chart of job processing with the MSHEFT algorithm and automatic SQL interface selection, as shown in Fig. 9.

2.7 Two-Class Caching in Data Retrieval

Two-class caching is usually used to maximize the distributed system's performance and optimize data retrieval and job scheduling. Two-class caching consists of in-memory and in-disk caches to save excessive hardware resource consumption caused by repeated searches. Two-class caching describes the flow in the following, as shown in Fig. 10.

2.8 In-Memory Cache Mechanism

In terms of in-memory cache design, Memcached [30], a distributed high-speed fast memory system, temporarily stores data. Memcached can store the data in the memory in the form of key-value. A colossal hash table is constructed in the memory so that users can quickly query the corresponding data value through the key-value. Therefore, it is often used as a cache function to speed up the access performance of the system. However, the Memcached system has specific restrictions on its use. The default maximum length of the key-value is only 250 characters, and the acceptable storage data size cannot exceed 1 MB. Therefore, it is necessary first to divide and store the search results when storing the cache.

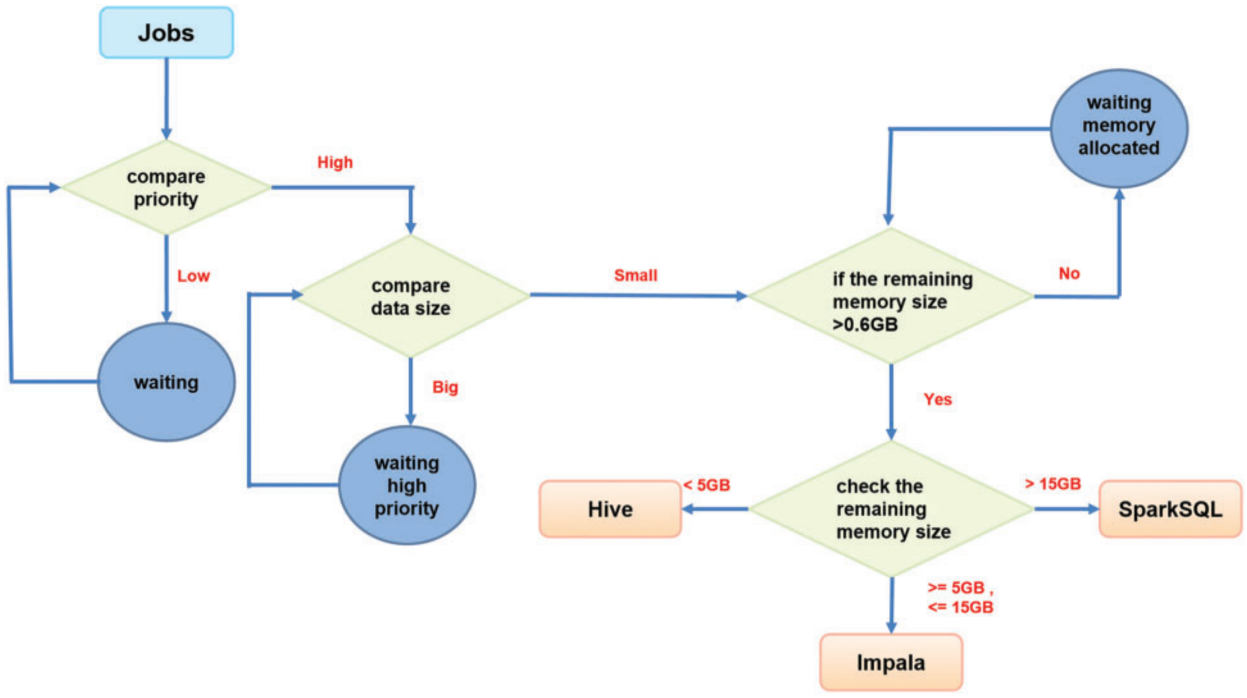


Figure 9: Job scheduling with MSHEFT algorithm and SQL interface selection

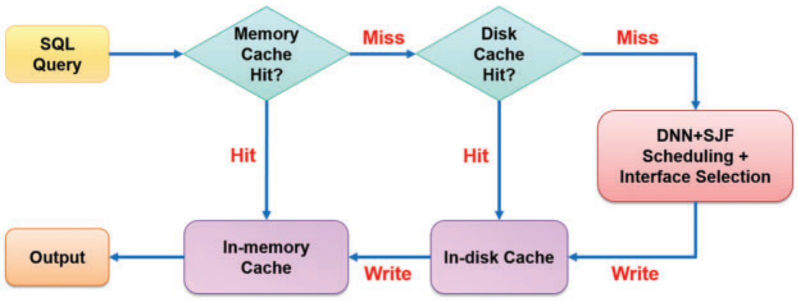


Figure 10: The flow of two-class caching in data retrieval

After the user enters the SQL command, the system will first use the MD5 algorithm [31] to convert the command into 16 bytes of hash value results and use the hash value as the unique identification code (i.e., key-value) for this search job, as shown in Fig. 11. If there is the exact SQL requirement later, it still results in the same hash value through the MD5 algorithm. The system can obtain the data through this unique identification code and realize the mechanism of cache design.

2.9 In-Disk Cache Mechanism

As for in-disk, this paper uses HDFS distributed file system as the platform for in-disk cache storage. Since HDFS does not have storage capacity limitations like the Memcached system, it is only necessary to save a copy of the search result and upload it to HDFS. Similarly, the exact unique identification code as applied in the in-memory cache system names its file for identification, as shown in Fig. 12. Since files in HDFS will not become invalid automatically, users must periodically delete them manually in HDFS. Therefore, this paper also has the function of active clearance. The user only needs to enter the "purge x" command in the CLI, and the system can automatically delete the cache data that the user has not accessed within x days.

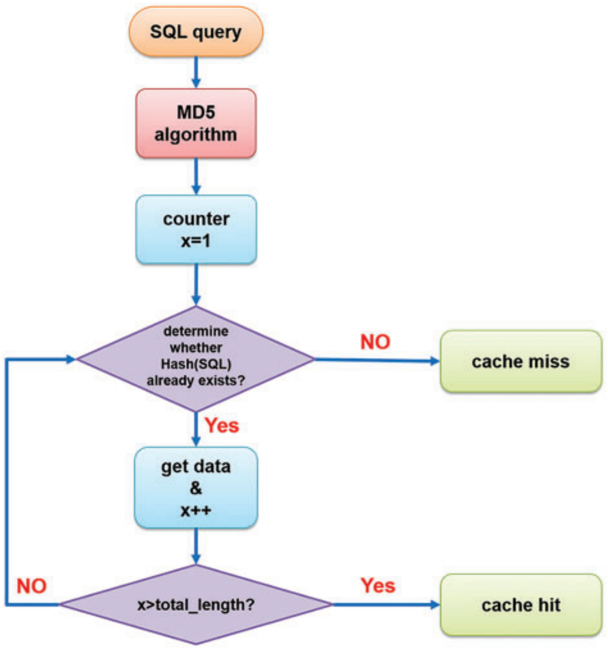
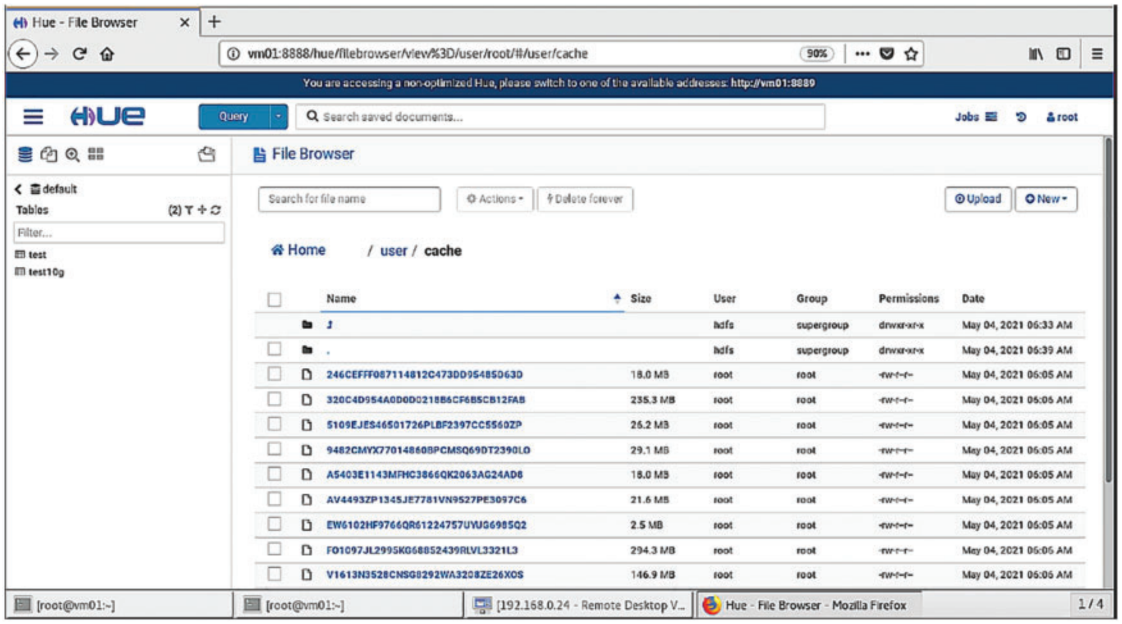


Figure 11: In-memory cache flow


Hue - File Browser
vm01:8888/hue/filebrowser/view%3D/user/root/#/user/cache
90%
You are accessing a non-optimized Hue, please switch to one of the available addresses: http://vm01:8889
Query
Search saved documents...
Jobs
root
File Browser
default
Tables (2)
Filter...
test
test10g
Search for file name
Actions
Delete forever
Upload
New
Home / user / cache
[root@vm01:~]
[root@vm01:~]
[192.168.0.24 - Remote Desktop V...
Hue - File Browser - Mozilla Firefox
1 / 4

Name	Size	User	Group	Permissions	Date
↥		hdfs	supergroup	drwxr-xr-x	May 04, 2021 06:33 AM
.		hdfs	supergroup	drwxr-xr-x	May 04, 2021 06:39 AM
246CEFFF087114812C473DD954850630	18.0 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
320C4D954A0D0D0218B6CF6B5CB12FAB	235.3 MB	root	root	-rw-r--r--	May 04, 2021 06:06 AM
5109EJES46501726PLBF2397CC5560ZP	26.2 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
9482CMYX77014860BPCMSQ69DT2390LO	29.1 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
A5403E1143MFHC3866QK2063AG24AD8	18.0 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
AV4493ZP1345JE7781VN9527PE3097C6	21.6 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
EW6102HF9766QR61224757UYUG6985Q2	2.5 MB	root	root	-rw-r--r--	May 04, 2021 06:05 AM
FO1097JL2995KG68852439RLVL3321L3	294.3 MB	root	root	-rw-r--r--	May 04, 2021 06:06 AM
V1613N3528CNSG8292WA3208ZE26XOS	146.9 MB	root	root	-rw-r--r--	May 04, 2021 06:06 AM

Figure 12: Cached files saving in HDFS

3 Method

3.1 Improved Big Data Analytics

In Fig. 13, improving the performance of big data analytics in a business intelligence system concerns three aspects: data retrieval, job scheduling, and different SQL command interfaces. The first one aims to speed up the data searching in big data. Therefore, this study must reduce the scope of data searching in the database and then implement data retrieval as fast as possible. This study introduced an approach to reduce the searching scope using data clustering with stacked sparse autoencoder, followed by a fast query response using quickly distributed data indexing with Elasticsearch, then stored tables into HDFS. The speed of data retrieval is relative to SQL command interfaces, so we will explore how to find the appropriate interface for the current situations of a node. This study has introduced three SQL command interfaces: Hive, Impala, and SparkSQL [28].

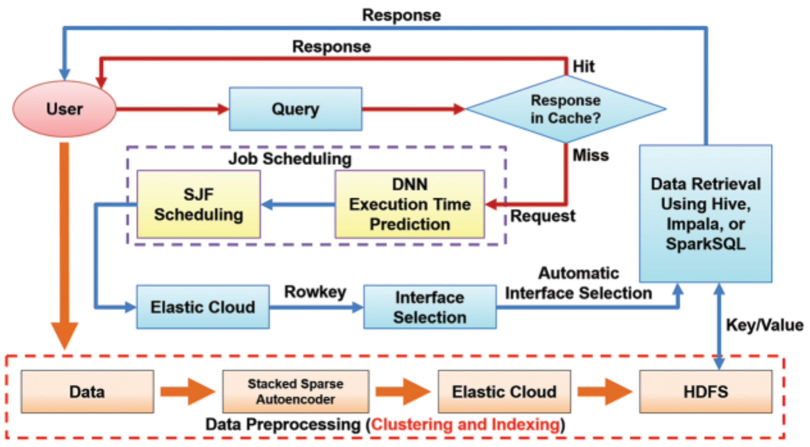
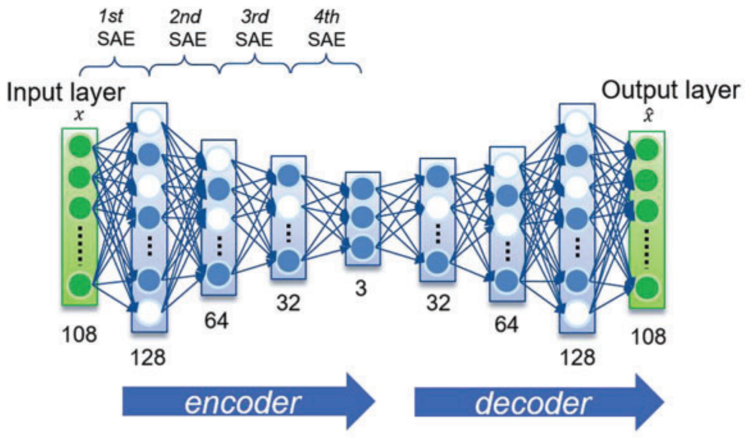

Response
Response
Hit
User
Query
Response in Cache?
Miss
Job Scheduling
SJF Scheduling
DNN Execution Time Prediction
Request
Data Retrieval Using Hive, Impala, or SparkSQL
Automatic Interface Selection
Elastic Cloud
Rowkey
Interface Selection
Key/Value
Data
Stacked Sparse Autoencoder
Elastic Cloud
HDFS
Data Preprocessing (Clustering and Indexing)

Figure 13: The optimization of big data analytics flow

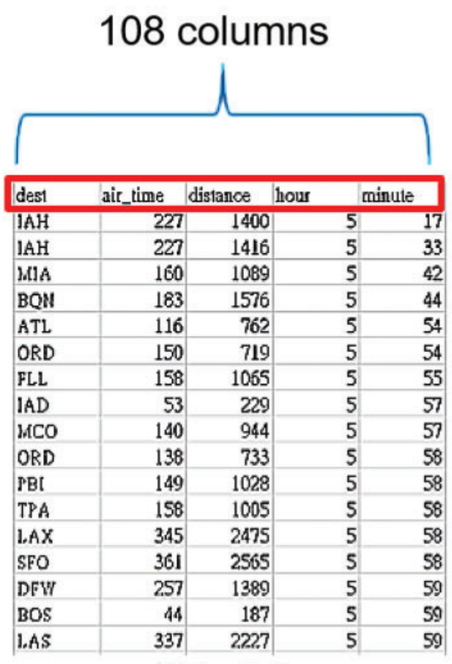
On the other hand, the deep neural network carried out the time prediction of job execution to optimize job scheduling in big data analytics. According to the job with the shortest time for execution as the higher priority in scheduling, it can reduce the average waiting time in a queue. Finally, users only need to enter SQL commands through the Command-Line Interface (CLI), and the system will automatically detect the remaining memory size of a node and select the appropriate SQL interface to carry out the jobs, which can also speed up the job execution and is called the automatic interface selection. In particular, a SQL query can save the searching result into an in-memory or in-disk cache. Therefore, you can directly cache the data without launching the SQL command interfaces if you want to retrieve the same data repeatedly in a short time. The cache indeed speeds up the data retrieval dramatically.

3.2 Data Clustering by Mapping

The Stacked Sparse Autoencoder (SSAE) model can apply to a data set clustering. The dimensional vector of the SSAE input and output layers is the number of initial data columns denoted m. SSAE constructs the l-layer neurons as an encoder and the mirror layers as a decoder with unsupervised learning. The output of each layer is connected to the input of followed layer to reduce the data dimension and finally obtain an output of n-dimensional vector in the middle layer, and then run the decoder of an SSAE in reverse order to restore the initial input data column. There is an example of SSAE, as shown in Fig. 14. In Fig. 14, the loss function is MSE, the activation function is Sigmoid, and the optimizer is Adam.

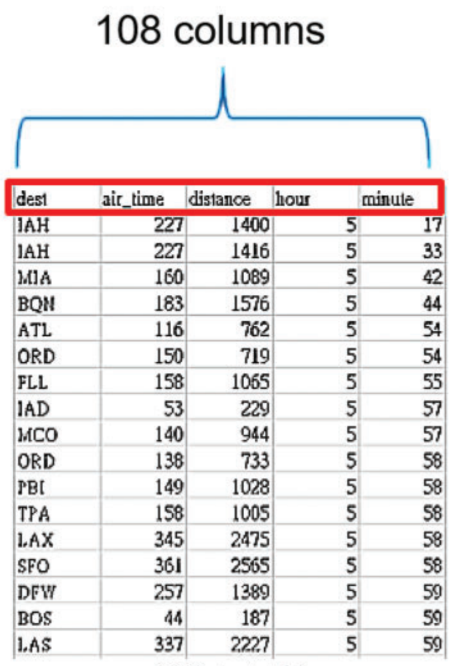


(a) Stacked sparse autoencoder architecture


108 columns

dest	air_time	distance	hour	minute
IAH	227	1400	5	17
IAH	227	1416	5	33
MIA	160	1089	5	42
BQN	183	1576	5	44
ATL	116	762	5	54
ORD	150	719	5	54
FLL	158	1065	5	55
IAD	53	229	5	57
MCO	140	944	5	57
ORD	138	733	5	58
PBI	149	1028	5	58
TPA	158	1005	5	58
LAX	345	2475	5	58
SFO	361	2565	5	58
DFW	257	1389	5	59
BOS	44	187	5	59
LAS	337	2227	5	59

(b) Input table

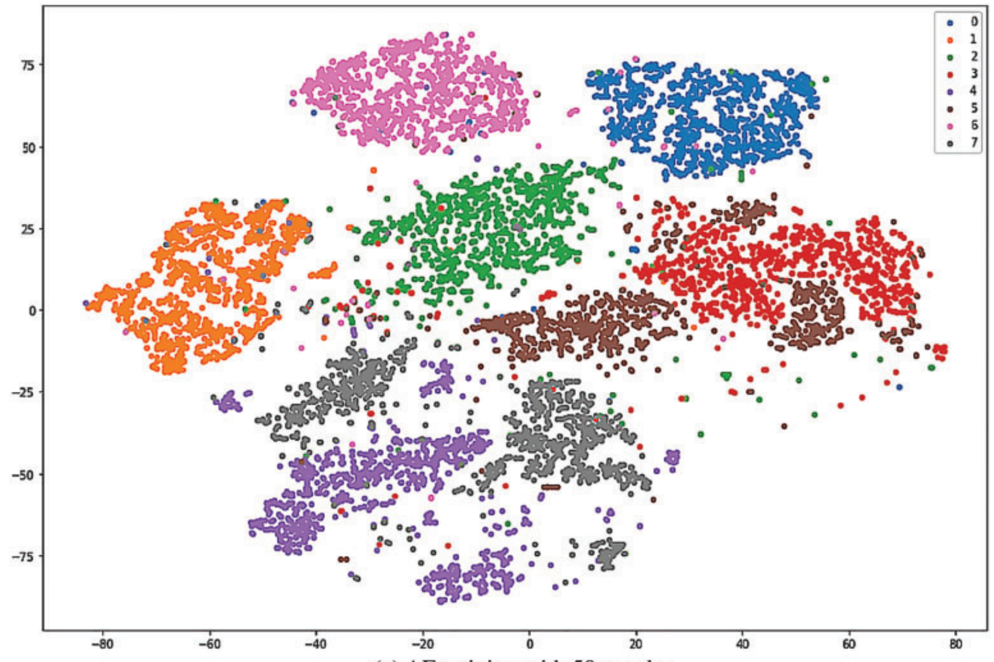

108 columns

dest	air_time	distance	hour	minute
IAH	227	1400	5	17
IAH	227	1416	5	33
MIA	160	1089	5	42
BQN	183	1576	5	44
ATL	116	762	5	54
ORD	150	719	5	54
FLL	158	1065	5	55
IAD	53	229	5	57
MCO	140	944	5	57
ORD	138	733	5	58
PBI	149	1028	5	58
TPA	158	1005	5	58
LAX	345	2475	5	58
SFO	361	2565	5	58
DFW	257	1389	5	59
BOS	44	187	5	59
LAS	337	2227	5	59

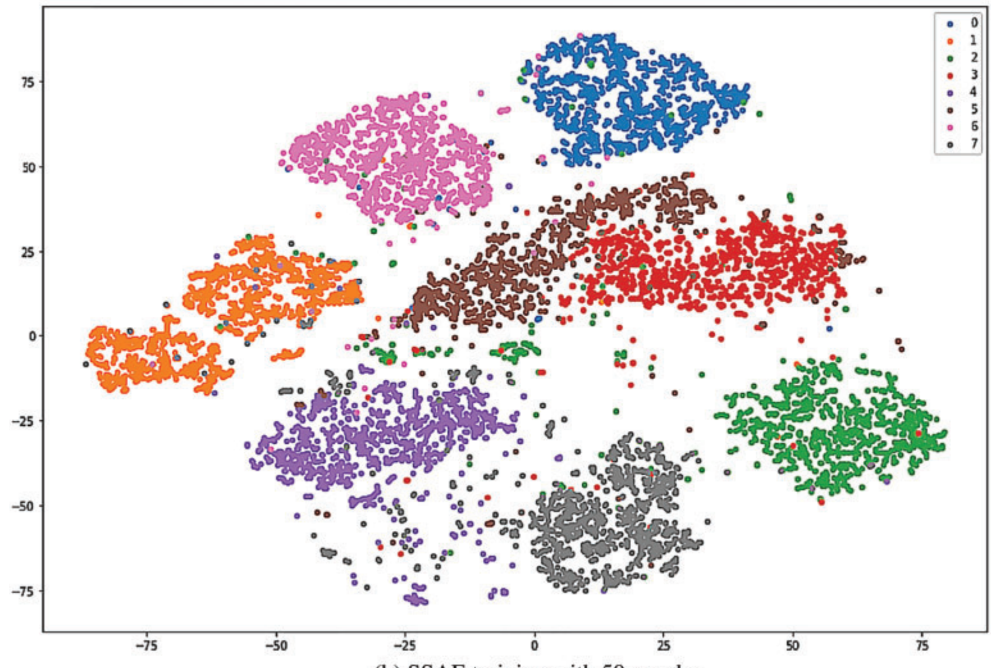
(c) Output table

Figure 14: Data clustering using stacked sparse autoencoder

The results of running 50 epochs between AE and SSAE training show their visualized results in Fig. 15. Since AE does not have sparsity restriction, it turns out to be some of the different classes of data entangled around the same area, resulting in uneven data distribution and worse the effect of data clustering. In Fig. 15a, two data sets may be mapped to the same area and have entangled, where there is no apparent separation between the classes, such as gray-class tangled with purple-class, and red-class tangled with brown-class. The SSAE with sparse constraints can effectively solve this problem, making the sampled data of the same class closer and the separation between different classes more pronounced, as shown in Fig. 15b.



(a) AE training with 50 epochs



(b) SSAE training with 50 epochs

Figure 15: Data visualization results between AE and SSAE training

Here, we separate data into eight classes for a specific data set. Therefore, the output of the last hidden layer of the encoder from a trained SSAE model represents the mapping of each row of data to a point in the 3-dimensional space. According to the X, Y, and Z axes in the 3-dimensional space, SSAE automatically divided the data into eight classes. The corresponding eight quadrants numbering from 1 to 8 are inserted into the last column of the original data table, as shown in Figs. 1 and 2. In this study, the user applies the SSAE to data clustering instead of AE. After data clustering, the user can upload the tables to Elasticsearch for distributed indexing in them.

3.3 Data Retrieval by Quick Indexing

The system will generally spend longer I/O time to cope with large-scale data retrieval. Distributed indexing using Elasticsearch can speed up the data search speed and resolve the bottleneck problem about single-machine processing with time-consuming. Multi-server processing in a cluster can implement an indexing function to run the data search as fast as possible. When started each node, the system will automatically join this node and designate one of them to act as the master node. Accordingly, Elasticsearch, as shown in Fig. 16, indexed the data and then stored the data into the distributed file system HDFS.

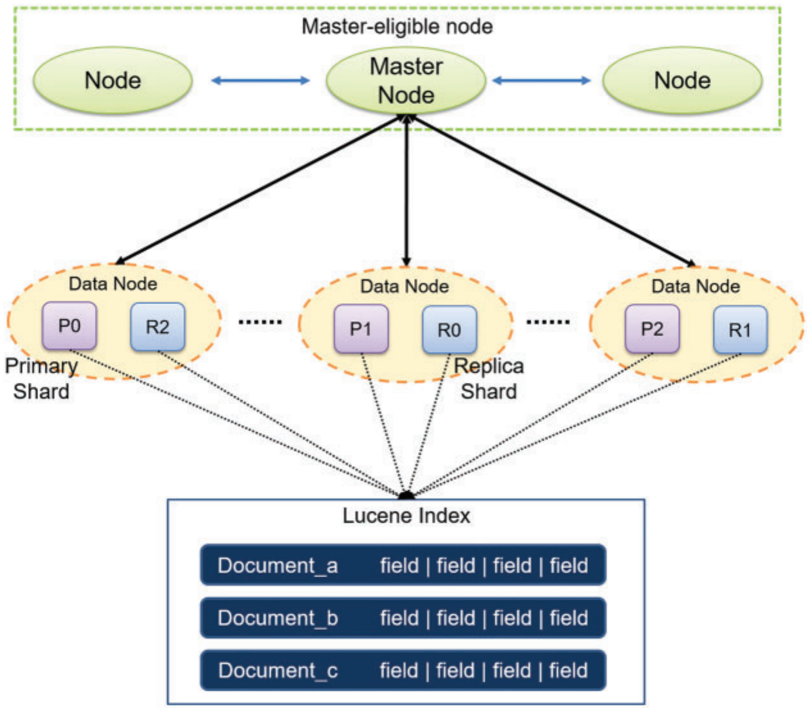

Master-eligible node
Node
Master Node
Node
Data Node
P0
R2
Primary Shard
Data Node
P1
R0
Replica Shard
Data Node
P2
R1
Lucene Index
Document_a field | field | field | field
Document_b field | field | field | field
Document_c field | field | field | field

Figure 16: Cluster by Elasticsearch

After clustering input data into three-dimensional space, the user can manually upload this data set to Elasticsearch for distributed indexing, as shown in Fig. 17. When an input data set needs to be indexed, the indexing tool starts if the current node is the master node; otherwise, it will forward it to the master node. In Fig. 18, Elasticsearch first checks whether the data set was indexed before. If not, Elasticsearch will start building the index, write the results into the Lucene index, and use the hash algorithm to ensure that it makes indexed data evenly distributed and stored in the designated primary shard and replica shard. Meanwhile, Elasticsearch will create a corresponding version number and store it in translog. Elasticsearch supposes comparing the existing version numbers with the new ones to check any conflict if the data set is indexed early. If not, it can start indexing. If yes, Elasticsearch returns an error result that writes to translog. Finally, indexed data stored in HDFS can apply for the job with SQL command, as shown in Fig. 13.

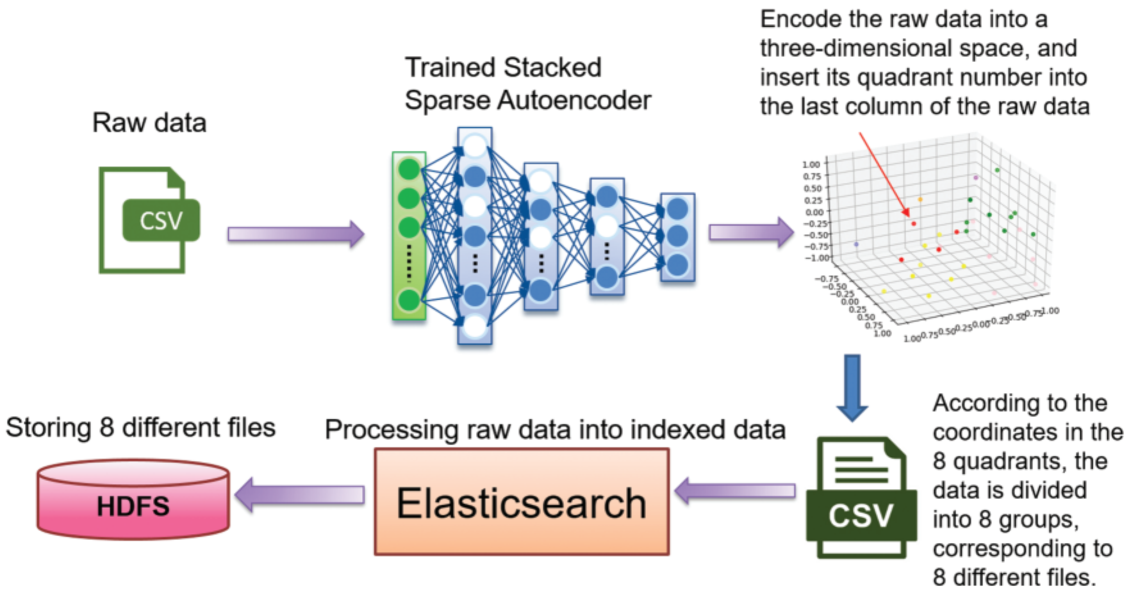

Raw data
CSV
Trained Stacked Sparse Autoencoder
Encode the raw data into a three-dimensional space, and insert its quadrant number into the last column of the raw data
According to the coordinates in the 8 quadrants, the data is divided into 8 groups, corresponding to 8 different files.
CSV
Processing raw data into indexed data
Elasticsearch
Storing 8 different files
HDFS

Figure 17: Data clustering and indexing flow

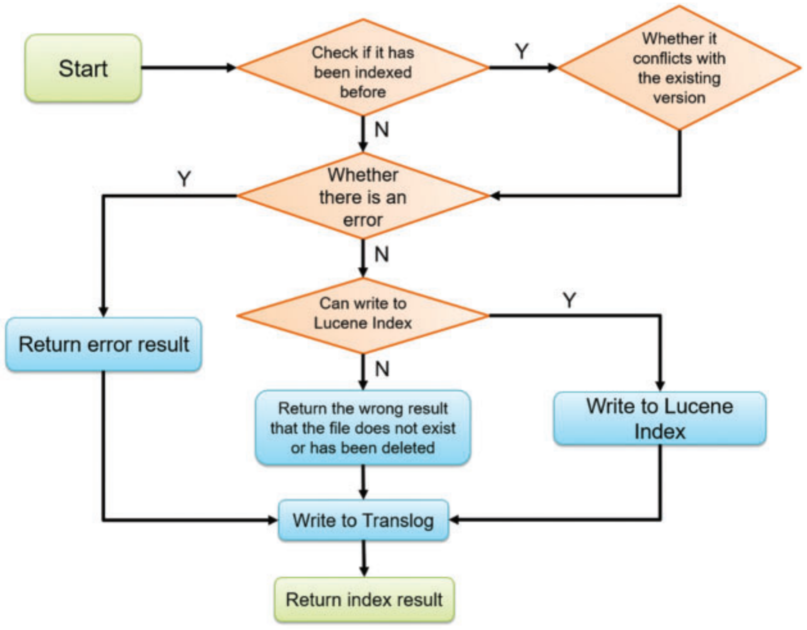
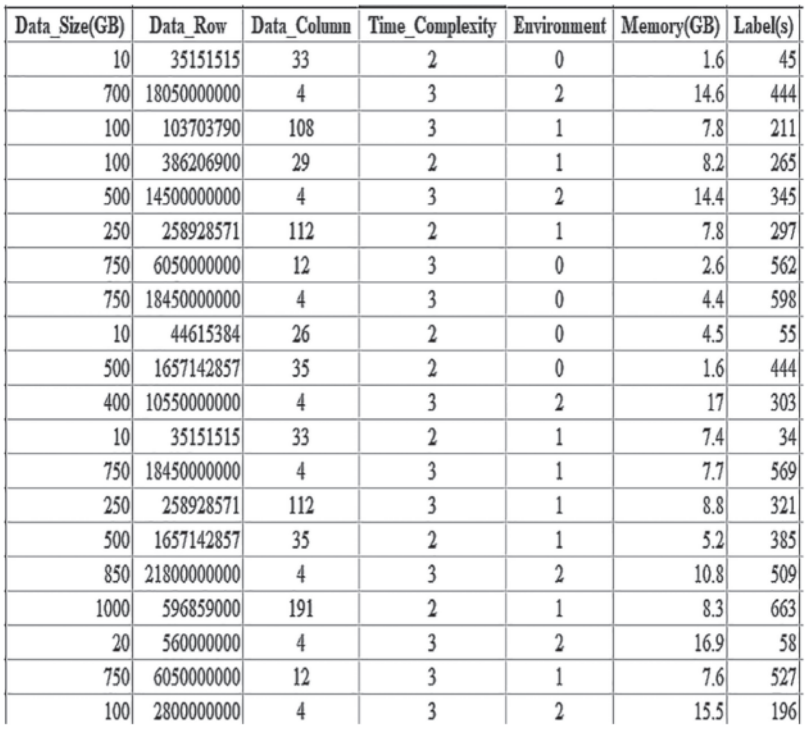

Start
Check if it has been indexed before
Y
Whether it conflicts with the existing version
N
Whether there is an error
Y
N
Can write to Lucene Index
Y
N
Return error result
Return the wrong result that the file does not exist or has been deleted
Write to Lucene Index
Write to Translog
Return index result

Figure 18: Indexing flow of Elasticsearch

3.4 *Deep Neural Network Prediction for Job Scheduling Optimization*

According to the SJF job scheduling algorithm, users can train a deep neural network (DNN) to predict the approximate job execution time. The input layer of this DNN has a six-dimensional vector, including data size, the number of data rows, the number of data columns, the time complexity of program execution, the SQL interface environment, and the remaining memory size. The output layer has only a one-dimensional vector (Label) as a time prediction of performing data retrieval, as shown in Fig. 19. Users can collect real data from the web to train a DNN model. Three sets of SQL interfaces—Hive, Impala, and SparkSQL—are used to test a DNN time prediction applied to the data retrieval function under the different remaining memory sizes given. The activation function in a DNN is ReLU, the loss function MSE, and the optimizer Adam [32]. We will give the DNN model architecture and loss curve during the training, as shown in Figs. 20 and 21, respectively.



Data_Size(GB)	Data_Row	Data_Column	Time_Complexity	Environment	Memory(GB)	Label(s)
10	35151515	33	2	0	1.6	45
700	18050000000	4	3	2	14.6	444
100	103703790	108	3	1	7.8	211
100	386206900	29	2	1	8.2	265
500	14500000000	4	3	2	14.4	345
250	258928571	112	2	1	7.8	297
750	6050000000	12	3	0	2.6	562
750	18450000000	4	3	0	4.4	598
10	44615384	26	2	0	4.5	55
500	1657142857	35	2	0	1.6	444
400	10550000000	4	3	2	17	303
10	35151515	33	2	1	7.4	34
750	18450000000	4	3	1	7.7	569
250	258928571	112	3	1	8.8	321
500	1657142857	35	2	1	5.2	385
850	21800000000	4	3	2	10.8	509
1000	596859000	191	2	1	8.3	663
20	560000000	4	3	2	16.9	58
750	6050000000	12	3	1	7.6	527
100	2800000000	4	3	2	15.5	196

Figure 19: Column "Label(s)" indicating the output of a deep neural network (DNN)

In addition, we used a combination of deep neural network prediction and the shortest job priority scheduling to optimize the job scheduling. When a job enters the queue, the system first considers its execution priority and then predicts its approximate execution time to view as a job scheduling condition. Finally, the system will consider the remaining memory size to select the appropriate SQL interface to carry out the input SQL command. In such a way, the system can implement the significant throughput. Fig. 22 describes the flow of the proposed method mentioned above.

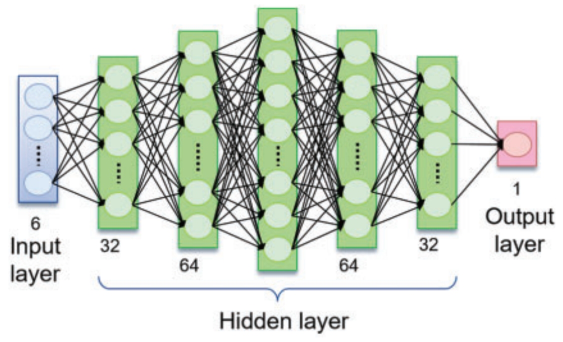


Figure 20: Deep neural network (DNN) architecture

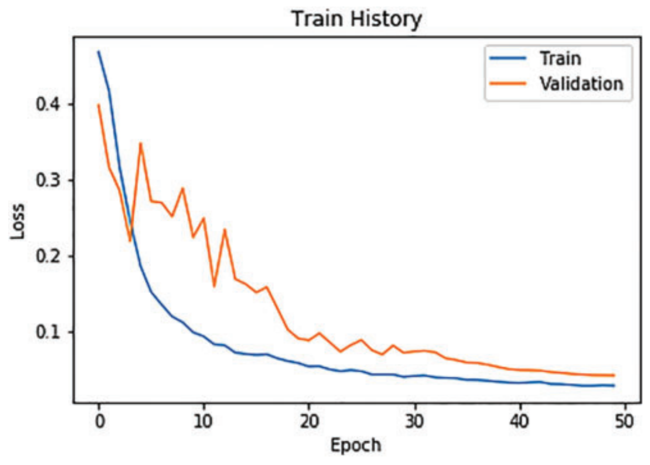


Figure 21: Loss curve during DNN training phase

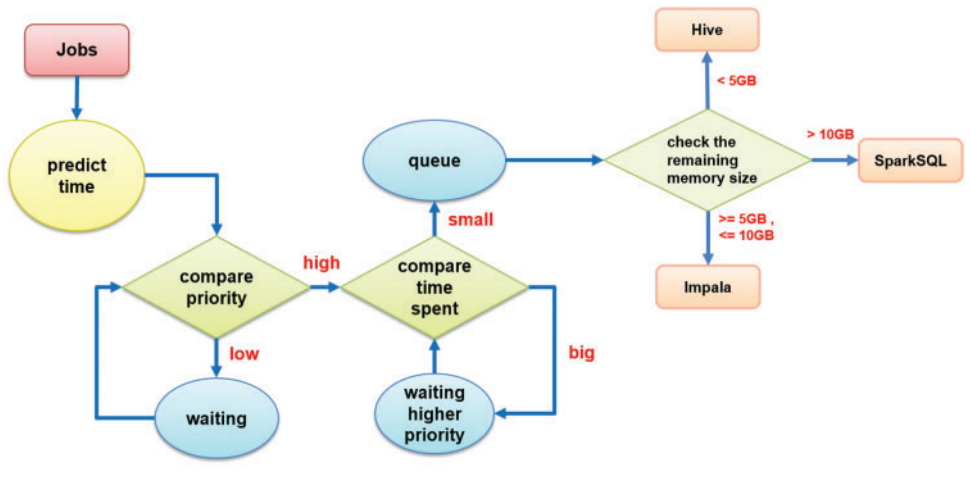


Figure 22: The flow of job scheduling optimization

3.5 Execution Commends and Its Flow

The designed programs in this study contain many application functions. Users can issue commands through the command-line interface (CLI) to input SQL commands. All commands of the programs describe their functions, as listed in Table 1. Fig. 23 gives the execution flow.

Table 1: Execution commands

Command	Function
sql [priority][sql_cmd]	Enter the SQL query commands with priority, and if the input priority is not set, the default value is 5 (the lowest priority)
status	Displays the current cluster status
flush	Clear all in-memory cache
purge [days]	Clear cache not access in the specified number of days
display [on|off]	Whether to display search results
enforced [name|auto]	Forced to use specified analysis interface
set [number]	Set the input SQL query command number

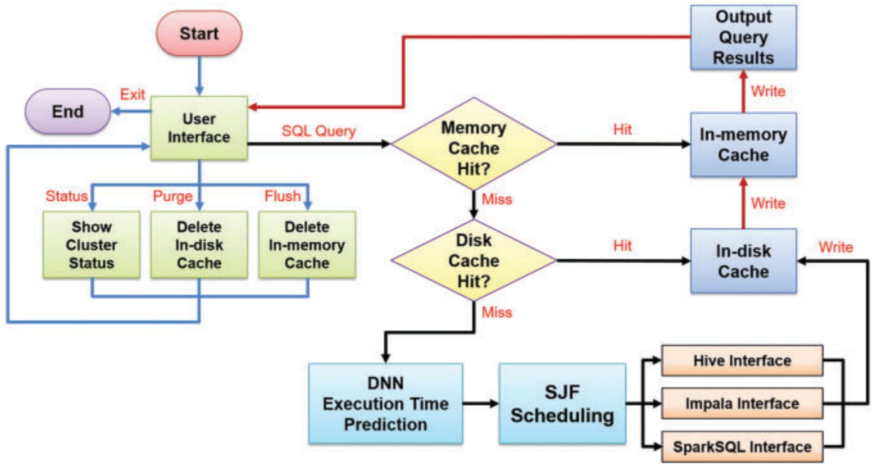


Figure 23: Job execution flow

4 Experiment Results and Discussion

4.1 Experimental Environment

The experiment uses the dynamic and adjustable resource characteristics of Proxmox VE to set up experimental environments with different memory sizes for the nodes in a cluster. The test environment is listed in Table 2.

Table 2: Test environment

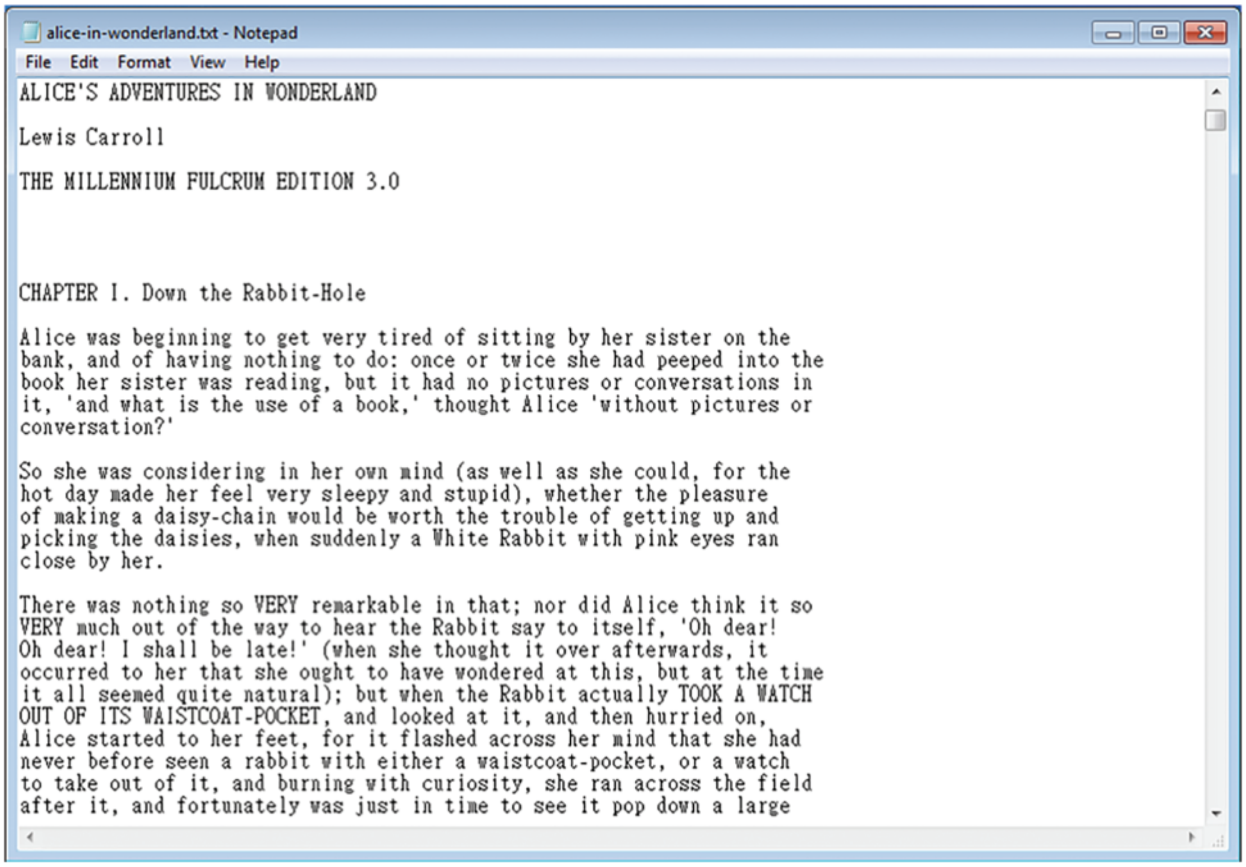
Test environment	Description
Environment I	Configure the memory space of the cluster virtual machine to 5 GB
Environment II	Configure the memory space of the cluster virtual machine to 10 GB
Environment III	Configure the memory space of the cluster virtual machine to 20 GB

4.2 Data Sets

The experiments proceed with all jobs with real data sets collected from the web to verify that the proposed approaches can effectively improve business intelligence performance in big data analytics. User inputs the SQL commands related to a certain real data set and measures the job execution time consumed in a specific SQL interface. The real data sets collected from the web includes (1) world-famous books [33], (2) production machine load [34], (3) semiconductor product yield [35], (4) temperature, rainfall, and electricity consumption related to people's livelihood [36,37], (5) forest flux station data [38], (6) Traffic violations/accidents [39], (7) Analysis of obesity factors [40], (8) Airport flight data [41]. The detailed information about real data sets describes as follows:

(1) World-famous books

This test is first to read the full text of the world-famous books as follows: Alice's Adventures in Wonderland, The Art of War, Adventures of Huckleberry Finn, Sherlock Holmes, and The Adventures of Tom Sawyer. After that, the word count task counts the number of occurrences of words, and sorts them from high-hit to low-hit in order. An example of plain text file of Alice's Adventures in Wonderland is shown in Fig. 24.

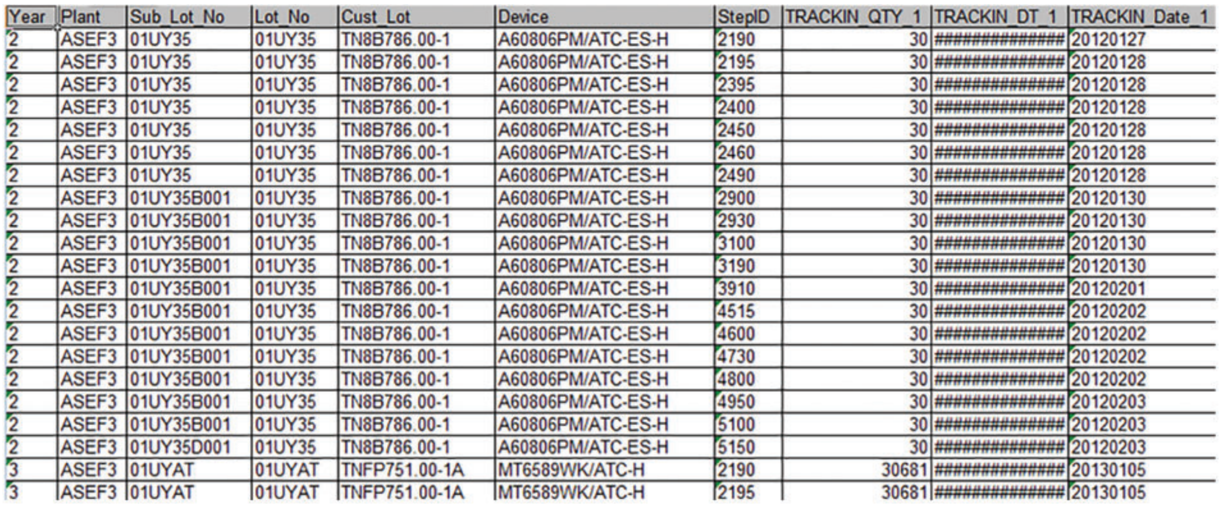


```
ALICE'S ADVENTURES IN WONDERLAND

Lewis Carroll

THE MILLENNIUM FULCRUM EDITION 3.0



CHAPTER I. Down the Rabbit-Hole

Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do: once or twice she had peeped into the
book her sister was reading, but it had no pictures or conversations in
it, 'and what is the use of a book,' thought Alice 'without pictures or
conversation?'

So she was considering in her own mind (as well as she could, for the
hot day made her feel very sleepy and stupid), whether the pleasure
of making a daisy-chain would be worth the trouble of getting up and
picking the daisies, when suddenly a White Rabbit with pink eyes ran
close by her.

There was nothing so VERY remarkable in that; nor did Alice think it so
VERY much out of the way to hear the Rabbit say to itself, 'Oh dear!
Oh dear! I shall be late!' (when she thought it over afterwards, it
occurred to her that she ought to have wondered at this, but at the time
it all seemed quite natural); but when the Rabbit actually TOOK A WATCH
OUT OF ITS WAISTCOAT-POCKET, and looked at it, and then hurried on,
Alice started to her feet, for it flashed across her mind that she had
never before seen a rabbit with either a waistcoat-pocket, or a watch
to take out of it, and burning with curiosity, she ran across the field
after it, and fortunately was just in time to see it pop down a large
```

Figure 24: Plain text file of Alice's Adventures in Wonderland

(2) Production machine load

Fig. 25 is a product production process report provided by Taiwan's large packaging and testing factory. The content includes the product number, the responsible employee, the name of the production machine, etc. The file format is .xls format. The purpose is to find the production machines used too frequently or too low in the production schedule and provide the decision-making analysis of the person in charge of the future schedule. Count the number of times the production machine is used and calculate the overall sample standard deviation. According to the concept of normal distribution, find out the data outside the "mean ± 2 ∗ standard deviation" range and treat it as a machine that may be Overloading or Underloading.

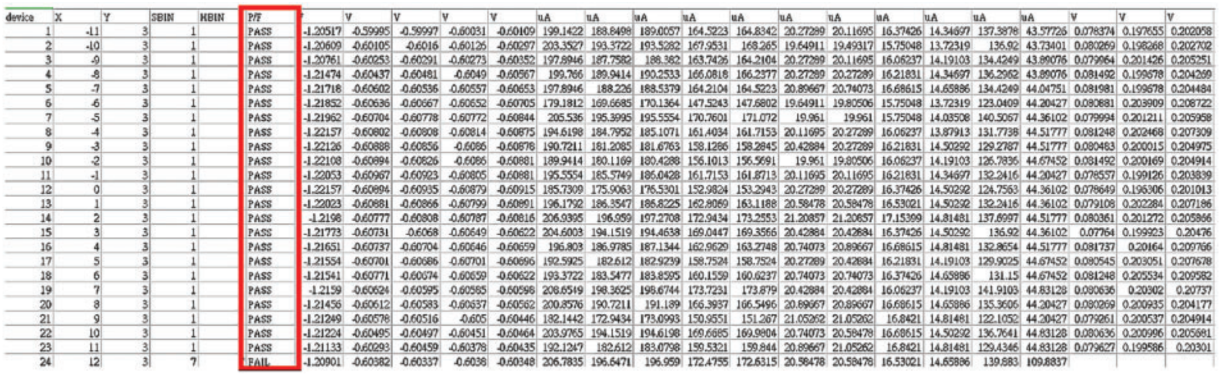


Year	Plant	Sub_Lot_No	Lot_No	Cust_Lot	Device	StepID	TRACKIN_QTY_1	TRACKIN_DT_1	TRACKIN_Date_1
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2190	30	##############	20120127
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2195	30	##############	20120128
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2395	30	##############	20120128
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2400	30	##############	20120128
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2450	30	##############	20120128
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2460	30	##############	20120128
2	ASEF3	01UY35	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2490	30	##############	20120128
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2900	30	##############	20120130
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	2930	30	##############	20120130
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	3100	30	##############	20120130
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	3190	30	##############	20120130
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	3910	30	##############	20120201
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	4515	30	##############	20120202
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	4600	30	##############	20120202
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	4730	30	##############	20120202
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	4800	30	##############	20120202
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	4950	30	##############	20120203
2	ASEF3	01UY35B001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	5100	30	##############	20120203
2	ASEF3	01UY35D001	01UY35	TN8B786.00-1	A60806PM/ATC-ES-H	5150	30	##############	20120203
3	ASEF3	01UYAT	01UYAT	TNFP751.00-1A	MT6589WK/ATC-H	2190	30681	##############	20130105
3	ASEF3	01UYAT	01UYAT	TNFP751.00-1A	MT6589WK/ATC-H	2195	30681	##############	20130105

Figure 25: Records of production machine loading

(3) Semiconductor product yield

In Fig. 26, the product test data provided by a significant packaging and testing company in Taiwan includes various semiconductor test items and PASS or FAIL results. The file format is a standard .csv format (separated by commas). The purpose is to calculate the yield rate of the product to see if it meets the company's yield rate standard (99.7%).

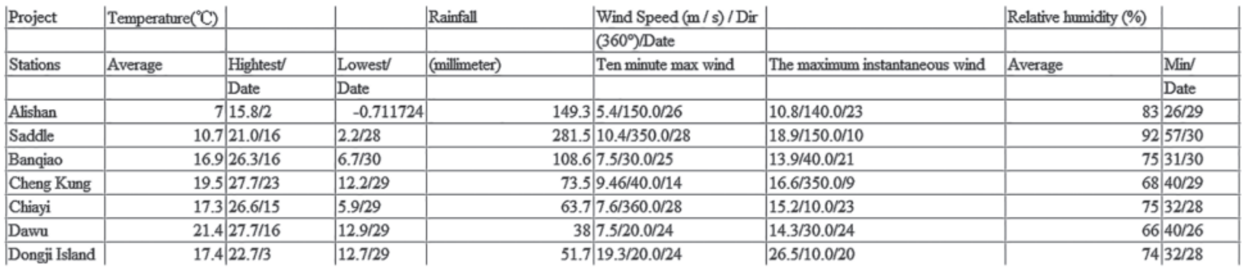


device	X	Y	SBIN	HBIN	P/F		V	V	V	V	uA	uA	uA	uA	uA	uA	uA	uA	uA	uA	uA	V	V	V
1	-11	3	1		PASS	-1.20517	-0.59995	-0.59997	-0.60031	-0.60109	199.1422	188.8498	189.0057	164.5223	164.8342	20.27289	20.11695	16.37426	14.34697	137.3878	43.57726	0.078374	0.197655	0.202058
2	-10	3	1		PASS	-1.20609	-0.60105	-0.6016	-0.60126	-0.60297	203.3527	193.3722	193.5282	167.9531	168.265	19.64911	19.49317	15.75048	13.72319	136.92	43.73401	0.080269	0.198268	0.202702
3	-9	3	1		PASS	-1.20761	-0.60253	-0.60291	-0.60273	-0.60352	197.8946	187.7582	188.382	163.7426	164.2104	20.27289	20.11695	16.06237	14.19103	134.4249	43.89076	0.079964	0.201426	0.205251
4	-8	3	1		PASS	-1.21474	-0.60437	-0.60481	-0.6049	-0.60567	199.766	189.9414	190.2533	166.0818	166.2377	20.27289	20.27289	16.21831	14.34697	136.2962	43.89076	0.081492	0.199678	0.204269
5	-7	3	1		PASS	-1.21718	-0.60602	-0.60536	-0.60557	-0.60653	197.8946	188.226	188.5379	164.2104	164.5223	20.89667	20.74073	16.68615	14.65886	134.4249	44.04751	0.081981	0.199678	0.204484
6	-6	3	1		PASS	-1.21852	-0.60636	-0.60667	-0.60652	-0.60705	179.1812	169.6685	170.1364	147.5243	147.6802	19.64911	19.80506	15.75048	13.72319	123.0409	44.20427	0.080881	0.203909	0.208722
7	-5	3	1		PASS	-1.21962	-0.60704	-0.60778	-0.60772	-0.60844	205.536	195.3995	195.5554	170.7601	171.072	19.961	19.961	15.75048	14.03508	140.5067	44.36102	0.079994	0.201211	0.205958
8	-4	3	1		PASS	-1.22157	-0.60802	-0.60808	-0.60814	-0.60875	194.6198	184.7952	185.1071	161.4034	161.7153	20.11695	20.27289	16.06237	13.87913	131.7738	44.51777	0.081248	0.202468	0.207309
9	-3	3	1		PASS	-1.22126	-0.60888	-0.60856	-0.6086	-0.60878	190.7211	181.2085	181.6763	158.1286	158.2845	20.42884	20.27289	16.21831	14.50292	129.2787	44.51777	0.080483	0.200015	0.204975
10	-2	3	1		PASS	-1.22108	-0.60894	-0.60826	-0.6086	-0.60881	189.9414	180.1169	180.4288	156.1013	156.5691	19.961	19.80506	16.06237	14.19103	126.7836	44.67452	0.081492	0.200169	0.204914
11	-1	3	1		PASS	-1.22053	-0.60967	-0.60923	-0.60805	-0.60881	195.5554	185.5749	186.0428	161.7153	161.8713	20.11695	20.11695	16.21831	14.34697	132.2416	44.20427	0.078557	0.199126	0.203839
12	0	3	1		PASS	-1.22157	-0.60894	-0.60935	-0.60879	-0.60915	185.7309	175.9063	176.5301	152.9824	153.2943	20.27289	20.27289	16.37426	14.50292	124.7563	44.36102	0.078649	0.196306	0.201013
13	1	3	1		PASS	-1.22023	-0.60881	-0.60866	-0.60799	-0.60891	196.1792	186.3547	186.8225	162.8069	163.1188	20.58478	20.58478	16.53021	14.50292	132.2416	44.36102	0.079108	0.202284	0.207186
14	2	3	1		PASS	-1.2198	-0.60777	-0.60808	-0.60787	-0.60816	206.9395	196.959	197.2708	172.9434	173.2553	21.20857	21.20857	17.15399	14.81481	137.6997	44.51777	0.080361	0.201272	0.205866
15	3	3	1		PASS	-1.21773	-0.60731	-0.6068	-0.60649	-0.60622	204.6003	194.1519	194.4638	169.0447	169.3566	20.42884	20.42884	16.37426	14.50292	136.92	44.36102	0.07764	0.199923	0.20476
16	4	3	1		PASS	-1.21651	-0.60737	-0.60704	-0.60646	-0.60659	196.803	186.9785	187.1344	162.9629	163.2748	20.74073	20.89667	16.68615	14.81481	132.8654	44.51777	0.081737	0.20164	0.209766
17	5	3	1		PASS	-1.21554	-0.60701	-0.60686	-0.60701	-0.60696	192.5925	182.612	182.9239	158.7524	158.7524	20.27289	20.42884	16.21831	14.19103	129.9025	44.67452	0.080545	0.203051	0.207678
18	6	3	1		PASS	-1.21541	-0.60771	-0.60674	-0.60659	-0.60622	193.3722	183.5477	183.8595	160.1559	160.6237	20.74073	20.74073	16.37426	14.65886	131.15	44.67452	0.081248	0.205534	0.209582
19	7	3	1		PASS	-1.2159	-0.60624	-0.60595	-0.60565	-0.60598	208.6519	198.3625	198.6744	173.7231	173.879	20.42884	20.42884	16.06237	14.19103	141.9103	44.83128	0.080636	0.20302	0.20737
20	8	3	1		PASS	-1.21456	-0.60612	-0.60583	-0.60637	-0.60562	200.8576	190.7211	191.189	166.3937	166.5496	20.89667	20.89667	16.68615	14.65886	135.3606	44.20427	0.080269	0.200935	0.204177
21	9	3	1		PASS	-1.21249	-0.60578	-0.60516	-0.605	-0.60446	182.1442	172.9434	173.0993	150.9551	151.267	21.05262	21.05262	16.8421	14.81481	122.1052	44.20427	0.079261	0.200537	0.204914
22	10	3	1		PASS	-1.21224	-0.60495	-0.60497	-0.60451	-0.60464	203.9765	194.1519	194.6198	169.6685	169.9804	20.74073	20.58478	16.68615	14.50292	136.7641	44.83128	0.080636	0.200996	0.205681
23	11	3	1		PASS	-1.21133	-0.60293	-0.60459	-0.60378	-0.60435	192.1247	182.612	183.0798	159.5321	159.844	20.89667	21.05262	16.8421	14.81481	129.4346	44.83128	0.079627	0.199586	0.20301
24	12	3	7		FAIL	-1.20901	-0.60382	-0.60337	-0.6038	-0.60348	206.7835	196.6471	196.959	172.4755	172.6315	20.58478	20.58478	16.53021	14.65886	139.883	109.8837			

Figure 26: Records of semiconductor product

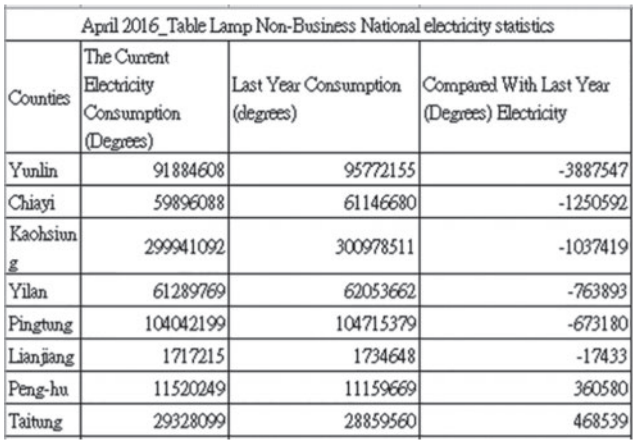
(4) Temperature, rainfall, and electricity consumption related to livelihood

The website of Taiwan meteorological bureau has collected the rainfall and temperature data, as shown in Fig. 27. The website of TAIPOWER has provided livelihood electricity data, as shown in Fig. 28. For both of them data collection period is from January 01, 2007 to April 30, 2020. The purpose is to find the correlation between rainfall, temperature, and electricity consumption in Taiwan. Based on statistical "Correlation," the correlation coefficient between the data is calculated and displayed as positive, negative, or no correlation. A certain linear correlation exists when $0 < |r| < 1$ between the two variables. The closer $|r|$ approaches to 1, the closer the linear relationship between the two variables is. Contrarily, the closer $|r|$ approaches to 0, the weaker the linear relationship between the two variables is. Generally, it can be divided into three levels: $|r| < 0.4$ is a low-degree linear correlation; $0.4 \leq |r| < 0.7$ is a significant correlation; $0.7 \leq |r| < 1$ is a high-degree linear correlation.



Project	Temperature(℃)			Rainfall	Wind Speed (m / s) / Dir		Relative humidity (%)	
					(360°)/Date			
Stations	Average	Hightest/	Lowest/	(millimeter)	Ten minute max wind	The maximum instantaneous wind	Average	Min/
		Date	Date					Date
Alishan	7	15.8/2	-0.711724	149.3	5.4/150.0/26	10.8/140.0/23	83	26/29
Saddle	10.7	21.0/16	2.2/28	281.5	10.4/350.0/28	18.9/150.0/10	92	57/30
Banqiao	16.9	26.3/16	6.7/30	108.6	7.5/30.0/25	13.9/40.0/21	75	31/30
Cheng Kung	19.5	27.7/23	12.2/29	73.5	9.46/40.0/14	16.6/350.0/9	68	40/29
Chiayi	17.3	26.6/15	5.9/29	63.7	7.6/360.0/28	15.2/10.0/23	75	32/28
Dawu	21.4	27.7/16	12.9/29	38	7.5/20.0/24	14.3/30.0/24	66	40/26
Dongji Island	17.4	22.7/3	12.7/29	51.7	19.3/20.0/24	26.5/10.0/20	74	32/28

Figure 27: Records of temperature and rainfall

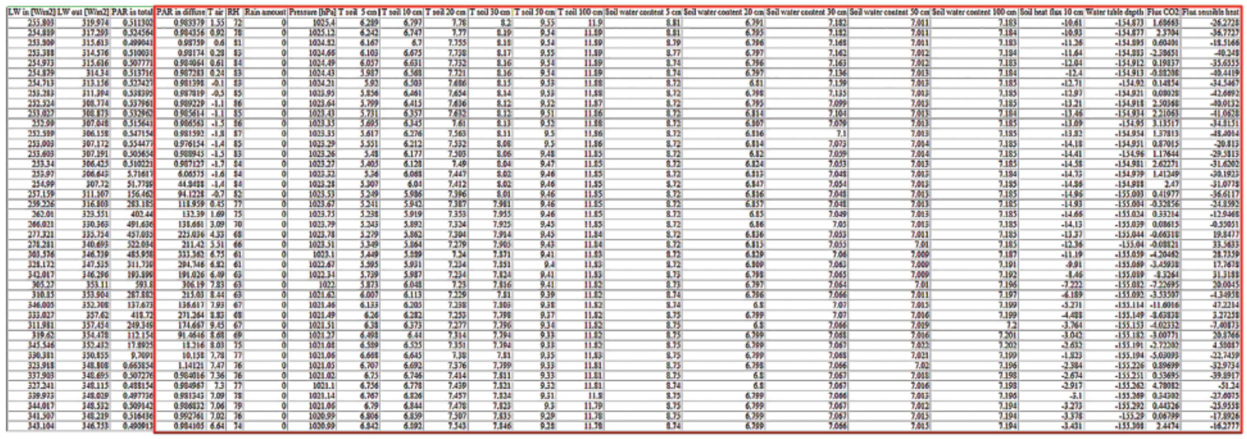


April 2016_Table Lamp Non-Business National electricity statistics			
Counties	The Current Electricity Consumption (Degrees)	Last Year Consumption (degrees)	Compared With Last Year (Degrees) Electricity
Yunlin	91884608	95772155	-3887547
Chiayi	59896088	61146680	-1250592
Kaohsiung	299941092	300978511	-1037419
Yilan	61289769	62053662	-763893
Pingtung	104042199	104715379	-673180
Lianjiang	1717215	1734648	-17433
Peng-hu	11520249	11159669	360580
Taitung	29328099	28859560	468539

Figure 28: Records of livelihood electricity consumption

(5) Forest flux station data

In Fig. 29, EU Open Data Portal has provided the forest flux station data that contains various flux information, including time and location, illuminance, soil information, and atmospheric flux. The file format is a standard .csv format (separated by commas). The purpose is to calculate the correlation coefficient between the CO_2 flux coefficient and the light intensity, temperature, and humidity, respectively, to examine the degree of correlation.

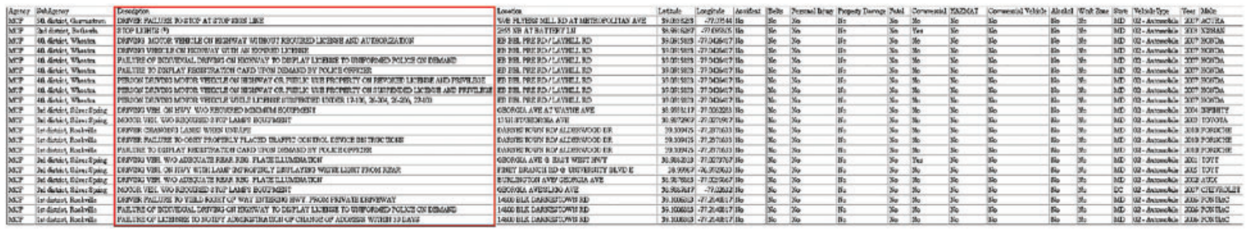


LW in [W/m2]	LW out [W/m2]	PAR in total	PAR in diffuse	T air	RH	Rain amount	Pressure [hPa]	T soil 5 cm	T soil 10 cm	T soil 20 cm	T soil 30 cm	T soil 50 cm	T soil 100 cm	Soil water content 5 cm	Soil water content 20 cm	Soil water content 30 cm	Soil water content 50 cm	Soil water content 100 cm	Soil heat flux 10 cm	Water table depth	Flux CO2	Flux sensible heat
255.803	319.974	0.511302	0.983379	1.55	72	0	1025.4	6.289	6.797	7.78	8.2	9.55	11.9	8.81	6.791	7.182	7.011	7.183	-10.61	-154.873	1.69693	-26.2728
254.839	317.293	0.524564	0.984356	0.92	78	0	1025.12	6.242	6.747	7.77	8.19	9.54	11.89	8.81	6.795	7.182	7.011	7.184	-10.93	-154.877	2.3704	-36.7727
253.909	315.613	0.499041	0.98759	0.6	81	0	1024.82	6.167	6.7	7.755	8.18	9.54	11.89	8.79	6.786	7.168	7.011	7.183	-11.26	-154.895	0.60401	-18.5166
253.388	314.576	0.510031	0.98174	0.28	83	0	1024.66	6.103	6.675	7.738	8.17	9.55	11.89	8.77	6.797	7.162	7.012	7.184	-11.64	-154.883	-2.38651	-40.248
254.973	315.616	0.507771	0.984064	0.61	84	0	1024.49	6.057	6.631	7.732	8.16	9.54	11.89	8.74	6.796	7.163	7.012	7.183	-12.04	-154.912	0.19837	-35.6555
254.879	314.34	0.513716	0.987283	0.24	83	0	1024.43	5.987	6.568	7.721	8.16	9.54	11.89	8.74	6.797	7.136	7.013	7.184	-12.4	-154.913	-0.88208	-40.4419
254.713	313.156	0.527427	0.981398	-0.1	83	0	1024.21	5.92	6.503	7.686	8.15	9.53	11.88	8.72	6.81	7.139	7.013	7.185	-12.71	-154.92	0.14854	-34.5467
253.283	311.394	0.538399	0.987819	-0.5	85	0	1023.95	5.856	6.461	7.654	8.14	9.53	11.88	8.72	6.798	7.135	7.013	7.185	-12.97	-154.921	0.08028	-42.6892
252.524	308.774	0.537961	0.989229	-1.1	86	0	1023.64	5.799	6.415	7.636	8.12	9.52	11.87	8.72	6.795	7.099	7.013	7.185	-13.21	-154.918	2.50388	-40.0132
253.027	308.873	0.532962	0.985614	-1.1	85	0	1023.43	5.731	6.357	7.632	8.12	9.51	11.86	8.72	6.814	7.104	7.013	7.184	-13.46	-154.934	2.21063	-41.0628
252.99	307.048	0.515641	0.986563	-1.5	86	0	1023.35	5.695	6.345	7.61	8.13	9.52	11.88	8.72	6.807	7.079	7.013	7.185	-13.09	-154.95	3.13517	-34.8151
252.589	306.158	0.547154	0.981592	-1.8	87	0	1023.35	5.617	6.276	7.563	8.11	9.5	11.86	8.72	6.816	7.1	7.013	7.185	-13.82	-154.954	1.37813	-48.4004
253.003	307.172	0.554477	0.976154	-1.4	85	0	1023.29	5.551	6.212	7.532	8.08	9.5	11.86	8.72	6.814	7.073	7.014	7.185	-14.18	-154.951	0.87015	-20.813
253.603	307.191	0.505654	0.988945	-1.5	83	0	1023.26	5.48	6.177	7.503	8.06	9.48	11.85	8.72	6.82	7.059	7.014	7.185	-14.41	-154.96	1.17644	-29.5813
253.34	306.425	0.510221	0.987127	-1.7	84	0	1023.27	5.405	6.128	7.49	8.04	9.47	11.85	8.72	6.824	7.053	7.013	7.185	-14.58	-154.981	2.62271	-31.6202
253.97	306.643	5.71617	6.06575	-1.6	84	0	1023.32	5.36	6.068	7.447	8.02	9.46	11.85	8.72	6.817	7.048	7.013	7.184	-14.73	-154.979	1.41249	-30.1923
254.99	307.72	51.7759	44.9488	-1.4	84	0	1023.28	5.307	6.04	7.412	8.02	9.46	11.85	8.72	6.847	7.054	7.013	7.185	-14.86	-154.988	2.47	-31.0778
257.159	311.307	156.462	94.1228	-0.7	82	0	1023.53	5.249	5.986	7.396	8.01	9.46	11.85	8.72	6.816	7.048	7.013	7.185	-14.96	-155.003	0.41977	-36.6117
259.226	316.803	283.185	118.959	0.45	77	0	1023.67	5.241	5.942	7.387	7.981	9.46	11.85	8.72	6.837	7.048	7.013	7.185	-14.93	-155.004	-0.32856	-24.8592
262.01	323.551	402.44	132.39	1.69	75	0	1023.75	5.238	5.919	7.353	7.955	9.46	11.85	8.72	6.85	7.049	7.013	7.185	-14.66	-155.024	0.33214	-12.9468
266.021	330.363	491.636	138.661	3.09	70	0	1023.79	5.243	5.892	7.324	7.925	9.45	11.85	8.72	6.86	7.05	7.013	7.185	-14.15	-155.039	0.08615	-0.55051
277.321	335.754	457.035	225.036	4.33	68	0	1023.78	5.279	5.862	7.304	7.914	9.45	11.84	8.72	6.836	7.053	7.011	7.185	-13.37	-155.044	-0.66318	19.8477
278.281	340.693	522.034	211.42	5.51	66	0	1023.51	5.349	5.864	7.279	7.903	9.43	11.84	8.72	6.815	7.055	7.01	7.185	-12.36	-155.04	-0.08821	33.5633
303.576	346.739	485.958	333.362	6.75	61	0	1023.1	5.449	5.889	7.24	7.871	9.41	11.83	8.72	6.829	7.06	7.009	7.187	-11.19	-155.059	-4.20482	28.7339
328.172	347.535	311.739	294.746	6.82	61	0	1022.67	5.595	5.931	7.234	7.851	9.4	11.83	8.72	6.809	7.063	7.009	7.191	-9.91	-155.069	-3.45938	17.7678
342.017	346.296	193.899	191.026	6.49	63	0	1022.34	5.739	5.987	7.234	7.824	9.41	11.83	8.73	6.798	7.065	7.009	7.192	-8.46	-155.089	-8.3264	31.3188
305.27	353.11	593.8	306.19	7.83	63	0	1022	5.873	6.048	7.23	7.816	9.41	11.82	8.73	6.797	7.064	7.01	7.196	-7.222	-155.082	-7.22695	20.0045
310.85	353.904	287.882	215.03	8.44	63	0	1021.62	6.007	6.113	7.229	7.81	9.39	11.82	8.74	6.796	7.066	7.011	7.197	-6.189	-155.092	-3.53507	-4.34958
346.005	352.708	137.673	136.617	7.93	67	0	1021.46	6.133	6.205	7.238	7.803	9.38	11.82	8.74	6.8	7.07	7.015	7.199	-5.271	-155.114	-11.6016	47.2214
333.027	357.62	418.72	271.264	8.83	68	0	1021.49	6.26	6.282	7.253	7.798	9.37	11.82	8.75	6.799	7.07	7.016	7.199	-4.488	-155.149	-8.63838	3.27258
311.981	357.454	249.349	174.667	9.45	67	0	1021.51	6.38	6.373	7.277	7.796	9.34	11.82	8.75	6.8	7.066	7.019	7.2	-3.764	-155.153	-4.02332	-7.40873
319.62	354.478	112.154	91.4646	8.68	69	0	1021.27	6.498	6.44	7.314	7.794	9.33	11.82	8.75	6.799	7.068	7.016	7.201	-3.042	-155.182	-3.00771	20.8766
345.546	352.432	17.8925	18.216	8.03	75	0	1021.08	6.589	6.525	7.351	7.794	9.33	11.82	8.75	6.799	7.067	7.022	7.202	-2.632	-155.191	-2.72202	4.50087
330.381	350.855	9.7091	10.158	7.78	77	0	1021.06	6.668	6.645	7.38	7.81	9.35	11.83	8.75	6.799	7.068	7.021	7.199	-1.823	-155.194	-5.03093	-22.7459
323.918	348.808	0.665854	1.14121	7.47	76	0	1021.05	6.707	6.692	7.376	7.799	9.33	11.81	8.75	6.798	7.066	7.02	7.196	-2.384	-155.226	0.89699	-32.9734
337.903	348.695	0.507276	0.984016	7.36	76	0	1021.02	6.75	6.746	7.414	7.811	9.33	11.81	8.75	6.8	7.067	7.018	7.198	-2.674	-155.251	0.53695	-39.8917
327.241	348.115	0.488154	0.984967	7.3	77	0	1021.1	6.756	6.778	7.439	7.821	9.32	11.81	8.74	6.8	7.067	7.016	7.198	-2.917	-155.262	4.78082	-51.24
339.973	348.029	0.497736	0.981343	7.09	78	0	1021.14	6.767	6.826	7.457	7.824	9.31	11.8	8.75	6.799	7.066	7.013	7.196	-3.1	-155.269	0.34302	-27.6075
344.017	348.532	0.509142	0.988832	7.06	79	0	1021.06	6.79	6.844	7.478	7.823	9.3	11.79	8.75	6.799	7.067	7.012	7.194	-3.273	-155.292	0.44326	-25.9558
341.507	348.219	0.516436	0.992761	7.02	76	0	1020.99	6.806	6.859	7.507	7.835	9.29	11.78	8.75	6.799	7.067	7.015	7.194	-3.378	-155.29	0.06799	-17.8926
343.104	346.333	0.490913	0.984105	6.64	74	0	1020.99	6.842	6.892	7.543	7.846	9.28	11.78	8.74	6.799	7.066	7.015	7.194	-3.431	-155.308	2.4474	-16.2777

Figure 29: Records of forest flux station

(6) Traffic violations/accidents

Fig. 30 has given the information about traffic violation/accident recorded from Maryland state in USA. Data are in the standard .csv format, i.e., each item separated by commas. Calculate the frequency statistics of monthly traffic violations and accident locations.



Agency	SubAgency	Description	Location	Latitude	Longitude	Accident	Belts	Personal Injury	Property Damage	Fatal	Commercial	HAZMAT	Commercial Vehicle	Alcohol	Work Zone	State	VehicleType	Year	Make
MCP	5th district, Germantown	DRIVER FAILURE TO STOP AT STOP SIGN LINE	WB FLYERS MILL RD AT METROPOLITAN AVE	[illegible]	-77.03544	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	ACURA
MCP	2nd district, Bethesda	STOP LIGHTS (*)	2655 NB AT BATTERY LN	[illegible]	[illegible]	No	No	No	No	No	Yes	No	No	No	No	MD	02 - Automobile	2008	NISSAN
MCP	4th district, Wheaton	DRIVING MOTOR VEHICLE ON HIGHWAY WITHOUT REQUIRED LICENSE AND AUTHORIZATION	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	DRIVING VEHICLE ON HIGHWAY WITH AN EXPIRED LICENSE	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	FAILURE OF INDIVIDUAL DRIVING ON HIGHWAY TO DISPLAY LICENSE TO UNIFORMED POLICE ON DEMAND	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	FAILURE TO DISPLAY REGISTRATION CARD UPON DEMAND BY POLICE OFFICER	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	PERSON DRIVING MOTOR VEHICLE ON HIGHWAY OR PUBLIC USE PROPERTY ON REVOKED LICENSE AND PRIVILEGE	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	PERSON DRIVING MOTOR VEHICLE ON HIGHWAY OR PUBLIC USE PROPERTY ON SUSPENDED LICENSE AND PRIVILEGE	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	4th district, Wheaton	PERSON DRIVING MOTOR VEHICLE WHILE LICENSE SUSPENDED UNDER 17-106, 26-204, 26-206, 27-103	EB BEL PRE RD / LAYHILL RD	39.0815833	-77.0426417	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2007	HONDA
MCP	3rd district, Silver Spring	DRIVING VEH. ON HWY. W/O REQUIRED MINIMUM EQUIPMENT	GEORGIA AVE AT WAYNE AVE	38.9918217	-77.0262283	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2004	INFINITY
MCP	3rd district, Silver Spring	MOTOR VEH. W/O REQUIRED STOP LAMPS EQUIPMENT	13TH ST/GEORGIA AVE	38.9872967	-77.0275967	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2003	TOYOTA
MCP	1st district, Rockville	DRIVER CHANGING LANES WHEN UNSAFE	DARNESTOWN RD/ ALDERWOOD DR	39.109475	-77.2871633	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2010	PORSCHE
MCP	1st district, Rockville	DRIVER FAILURE TO OBEY PROPERLY PLACED TRAFFIC CONTROL DEVICE INSTRUCTIONS	DARNESTOWN RD/ ALDERWOOD DR	39.109475	-77.2871633	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2010	PORSCHE
MCP	1st district, Rockville	FAILURE TO DISPLAY REGISTRATION CARD UPON DEMAND BY POLICE OFFICER	DARNESTOWN RD/ ALDERWOOD DR	39.109475	-77.2871633	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2010	PORSCHE
MCP	3rd district, Silver Spring	DRIVING VEH. W/O ADEQUATE REAR REG. PLATE ILLUMINATION	GEORGIA AVE @ EAST WEST HWY	38.9862019	-77.0273767	No	No	No	No	No	Yes	No	No	No	No	MD	02 - Automobile	2001	TOYT
MCP	3rd district, Silver Spring	DRIVING VEH. ON HWY WITH LAMP IMPROPERLY DISPLAYING WHITE LIGHT FROM REAR	PINEY BRANCH RD @ UNIVERSITY BLVD E	38.99967	-76.9929633	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2005	TOYT
MCP	3rd district, Silver Spring	DRIVING VEH. W/O ADEQUATE REAR REG. PLATE ILLUMINATION	BURLINGTON AVE/ GEORGIA AVE	38.9878283	-77.0295667	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2002	AUDI
MCP	3rd district, Silver Spring	MOTOR VEH. W/O REQUIRED STOP LAMPS EQUIPMENT	GEORGIA AVE/SLIGO AVE	38.9883617	-77.02612	No	No	No	No	No	No	No	No	No	No	DC	02 - Automobile	2007	CHEVROLET
MCP	1st district, Rockville	DRIVER FAILURE TO YIELD RIGHT OF WAY ENTERING HWY. FROM PRIVATE DRIVEWAY	14800 BLK DARNESTOWN RD	39.1006083	-77.2548817	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2006	PONTIAC
MCP	1st district, Rockville	FAILURE OF INDIVIDUAL DRIVING ON HIGHWAY TO DISPLAY LICENSE TO UNIFORMED POLICE ON DEMAND	14800 BLK DARNESTOWN RD	39.1006083	-77.2548817	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2006	PONTIAC
MCP	1st district, Rockville	FAILURE OF LICENSEE TO NOTIFY ADMINISTRATION OF CHANGE OF ADDRESS WITHIN 30 DAYS	14800 BLK DARNESTOWN RD	39.1006083	-77.2548817	No	No	No	No	No	No	No	No	No	No	MD	02 - Automobile	2006	PONTIAC

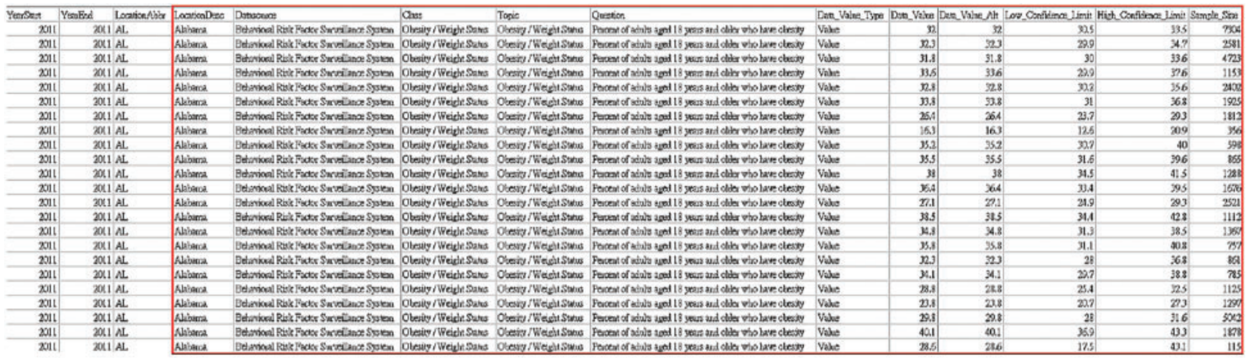
Figure 30: Records of traffic violation/accident

(7) Analysis of obesity factors

U.S. government has published the information about the obesity factors from 2011 to 2019, as shown in Fig. 31. Data are in the standard .csv format, i.e., each item separated by commas. The objective of this test is to analyze the relationship between age, weekly exercise status, vegetable and fruit intake, and BMI through data statistics. Therefore, people can understand whether these factors affect human's body overweight.

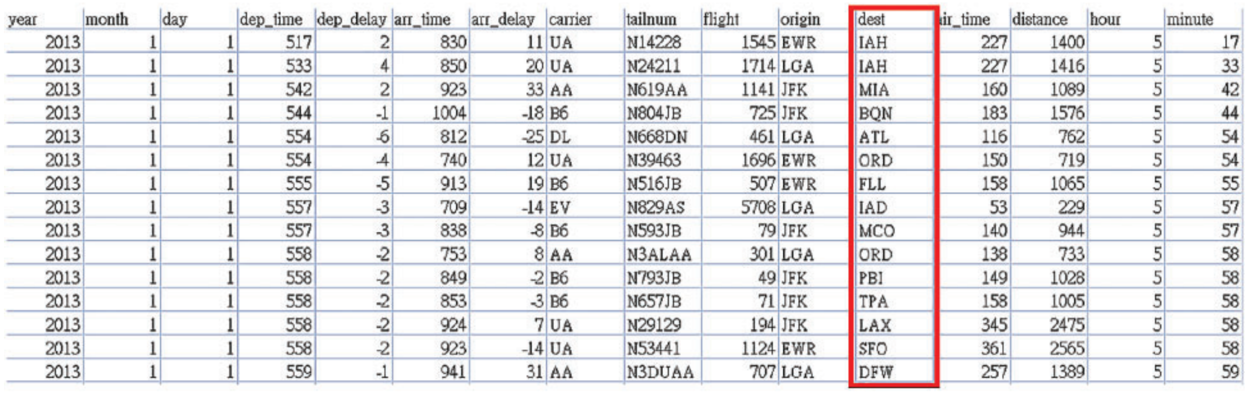
(8) Airport flight data

Fig. 32 has shown the airport flight information recorded from New York airports in USA. Data are in the standard .csv format, i.e., each item separated by commas. This test is to calculate the proportion of the airport's flights to Taiwan in the total flights in the year.



YearStart	YearEnd	LocationAbbr	LocationDesc	Datasource	Class	Topic	Question	Data_Value_Type	Data_Value	Data_Value_Alt	Low_Confidence_Limit	High_Confidence_Limit	Sample_Size
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	32	32	30.5	33.5	7304
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	32.3	32.3	29.9	34.7	2581
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	31.8	31.8	30	33.6	4723
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	33.6	33.6	29.9	37.6	1153
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	32.8	32.8	30.2	35.6	2402
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	33.8	33.8	31	36.8	1925
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	26.4	26.4	23.7	29.3	1812
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	16.3	16.3	12.6	20.9	356
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	35.2	35.2	30.7	40	598
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	35.5	35.5	31.6	39.6	865
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	38	38	34.5	41.5	1288
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	36.4	36.4	33.4	39.5	1676
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	27.1	27.1	24.9	29.3	2521
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	38.5	38.5	34.4	42.8	1112
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	34.8	34.8	31.3	38.5	1367
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	35.8	35.8	31.1	40.8	757
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	32.3	32.3	28	36.8	861
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	34.1	34.1	29.7	38.8	785
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	28.8	28.8	25.4	32.5	1125
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	23.8	23.8	20.7	27.3	1297
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	29.8	29.8	28	31.6	5042
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	40.1	40.1	36.9	43.3	1878
2011	2011	AL	Alabama	Behavioral Risk Factor Surveillance System	Obesity / Weight Status	Obesity / Weight Status	Percent of adults aged 18 years and older who have obesity	Value	28.5	28.6	17.5	43.1	115

Figure 31: Records of obesity factor



year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	ir_time	distance	hour	minute
2013	1	1	517	2	830	11	UA	N14228	1545	EWR	IAH	227	1400	5	17
2013	1	1	533	4	850	20	UA	N24211	1714	LGA	IAH	227	1416	5	33
2013	1	1	542	2	923	33	AA	N619AA	1141	JFK	MIA	160	1089	5	42
2013	1	1	544	-1	1004	-18	B6	N804JB	725	JFK	BQN	183	1576	5	44
2013	1	1	554	-6	812	-25	DL	N668DN	461	LGA	ATL	116	762	5	54
2013	1	1	554	-4	740	12	UA	N39463	1696	EWR	ORD	150	719	5	54
2013	1	1	555	-5	913	19	B6	N516JB	507	EWR	FLL	158	1065	5	55
2013	1	1	557	-3	709	-14	EV	N829AS	5708	LGA	IAD	53	229	5	57
2013	1	1	557	-3	838	-8	B6	N593JB	79	JFK	MCO	140	944	5	57
2013	1	1	558	-2	753	8	AA	N3ALAA	301	LGA	ORD	138	733	5	58
2013	1	1	558	-2	849	-2	B6	N793JB	49	JFK	PBI	149	1028	5	58
2013	1	1	558	-2	853	-3	B6	N657JB	71	JFK	TPA	158	1005	5	58
2013	1	1	558	-2	924	7	UA	N29129	194	JFK	LAX	345	2475	5	58
2013	1	1	558	-2	923	-14	UA	N53441	1124	EWR	SFO	361	2565	5	58
2013	1	1	559	-1	941	31	AA	N3DUAA	707	LGA	DFW	257	1389	5	59

Figure 32: Records of airport flight information

4.3 Data Retrieval Experiment

This experiment tested each data set in different experimental environments, executed them according to the issued SQL commands, and compared their performance using different approaches. Table 3 has listed the different approaches applied in the experiments.

Table 3: Test method

Method	Description
FIFO-Hive	Use the 'enforced hive' command to switch to the hive interface and enter the SQL command [28]
FIFO-Impala	Use the 'enforced impala' command to switch to the impala interface and enter the SQL command [28]
FIFO-SparkSQL	Use the 'enforced sparksql' command to switch to the sparksql interface and enter the SQL command [28]
MSHEFT	According to the job data size for scheduling and memory space for interface selection [22]

(Continued)

Table 3 (continued)

Method	Description
DAE-SORL + MSHEFT	Pre-process the data with DAE-SOLR in advance and according to the job data size for scheduling and memory space for interface selection [21]
SSAE-ES + DNNSJF	Pre-process the data with SSAE-Elasticsearch in advance, and use DNN to predict job execution time for SJF scheduling and memory space for interface selection (proposed in this study)

In Table 3, MSHEFT, DAE-SOLR + MSHEFT, and SSAE-ES + DNNSJF are different scheduling algorithms, and thus the order of job execution is scheduled in different sequences. In particular, the Hive interface can only complete the mission when the job has a large amount of data for analysis. However, the Impala or SparkSQL interface could not work successfully because the remaining memory size is insufficient. We noted that the performance of the Impala interface is more prominent when the remaining memory size is moderate. When the remaining memory size is significantly large, the SparkSQL interface can perform best due to its in-memory computing capability.

With automatic interface selection, the system can select the appropriate SQL interface and accept the SQL command from a user to perform the corresponding job in different scheduling algorithms, MSHEFT, DAE-SOLR + MSHEFT, and SSAE-ES + DNNSJF. If the data set sored in HDFS has completed data retrieval optimization in advance, the optimized data clustering and indexing can significantly reduce the average execution time of data searching. Experimental results have shown each job execution time, as shown in Figs. 33–35, and the average job execution time and system throughput, as listed in Tables 4–6.

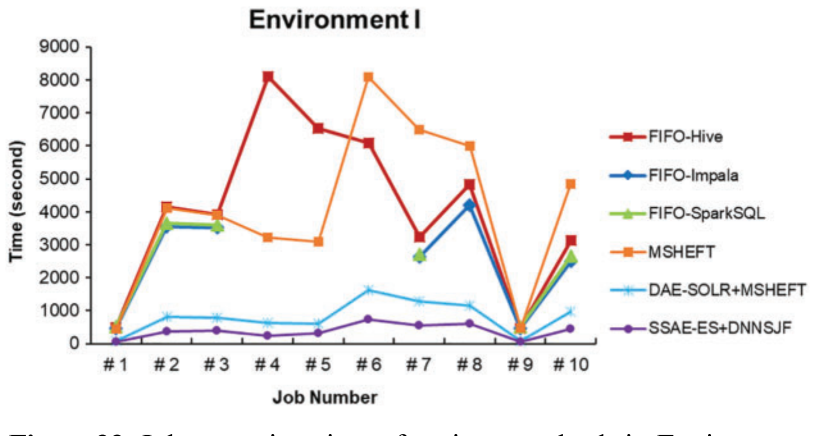


Figure 33: Job execution time of various methods in Environment I

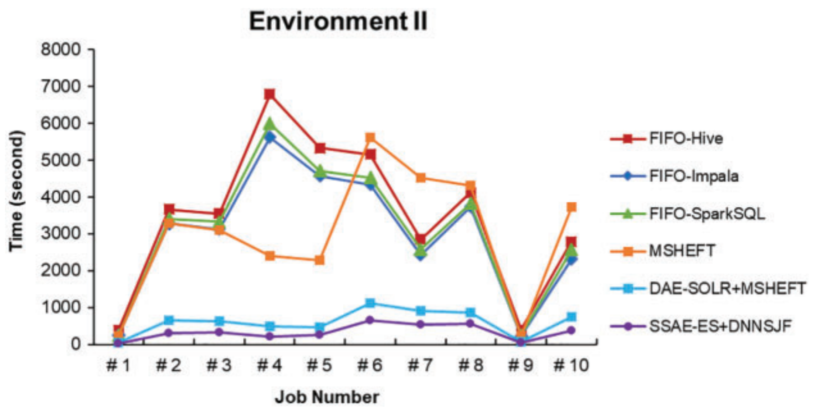


Figure 34: Job execution time of various methods in Environment II

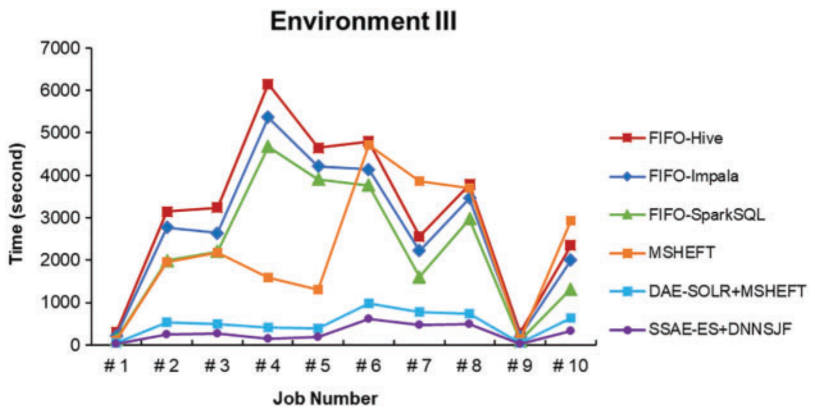


Figure 35: Job execution time of various methods in Environment III

Table 4: Average job execution time of various methods in Environment I

Method	Average Execution Time (second)	System Throughput (# of jobs/hour)
FIFO-Hive	4097	0.88
FIFO-Impala	–	–
FIFO-SparkSQL	–	–
MSHEFT	4070	0.88
DAE-SOLR + MSHEFT	806	4.47
SSAE-ES + DNNSJF	375	9.60

Note: symbol "–" stands for "not available".

Table 5: Average job execution time of various methods in Environment II

Method	Average Execution Time (second)	System Throughput (# of jobs/hour)
FIFO-Hive	3500	1.03
FIFO-Impala	2987	1.21
FIFO-SparkSQL	3143	1.15
MSHEFT	2971	1.21
DAE-SOLR + MSHEFT	597	6.03
SSAE-ES + DNNSJF	332	10.85

Table 6: Average job execution time of various methods in Environment III

Method	Average Execution Time (second)	System Throughput (# of jobs/hour)
FIFO-Hive	3126	1.15
FIFO-Impala	2725	1.32
FIFO-SparkSQL	2269	1.59
MSHEFT	2250	1.60
DAE-SOLR + MSHEFT	507	7.11
SSAE-ES + DNNSJF	285	12.63

4.4 Job Scheduling Experiment

Since this study uses an automatic SQL interface selection, the appropriate SQL interface can be automatically selected to perform data analysis according to available memory size. Therefore, the experiments use this way to test an appropriate SQL interface selected to perform data analysis in different environments and compares the average job waiting time for execution among different scheduling algorithms.

To comply with the requirement of the experimental tests hereafter, they will use SSAE and Elasticsearch for all data sets to execute data retrieval together with the various job scheduling, as listed in Table 7. We here tested the experimental environment I, II, and III, as listed in Table 2. Since FIFO, MSHEFT, and DNNSJF are different job scheduling algorithms, the job scheduling algorithms will execute in a different order.

According to the experimental results, different data analysis jobs take different times. Generally speaking, the job scheduling performance of MSHEFT is good. However, the MSHEFT algorithm will do the same job scheduling as the FIFO algorithm when different jobs have the same data set size. The experimental results show that the proposed approach DNNSJF algorithm obtains the shortest average waiting time and average weighted turnaround time for each job execution, followed by the MSHEFT scheduling algorithm that schedules the jobs based on the amount of data size, and the FIFO algorithm reaches the poor one. The experimental results have shown the waiting time for each job execution, as shown in Figs. 36–38. They have also shown the average job waiting time for execution and average weighted turnaround time, as listed in Tables 8–10.

Table 7: Job scheduling algorithm

Method	Description
SSAE-ES + FIFO	Use SSAE-Elasticsearch to preprocess the data in advance. The program will first calculate the remaining memory size of the system, automatically select the execution interface, and use the first-in-first-out algorithm for scheduling. (proposed in this study)
SSAE-ES + MSHEFT	Use SSAE-Elasticsearch to preprocess the data in advance. The program will first calculate the remaining memory size of the system, automatically select the execution interface, and schedule according to the size of the data from small to large. (proposed in this study)
SSAE-ES + DNNSJF	Use SSAE-Elasticsearch to preprocess the data in advance. The program will first calculate the remaining memory size of the system, automatically select the execution interface, and use DNN to predict the job execution time for SJF scheduling. (proposed in this study)

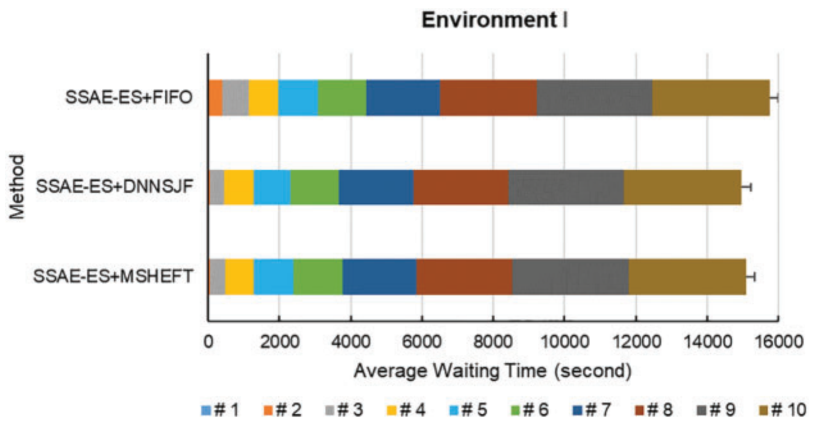


Figure 36: Job waiting time of various methods in Environment I

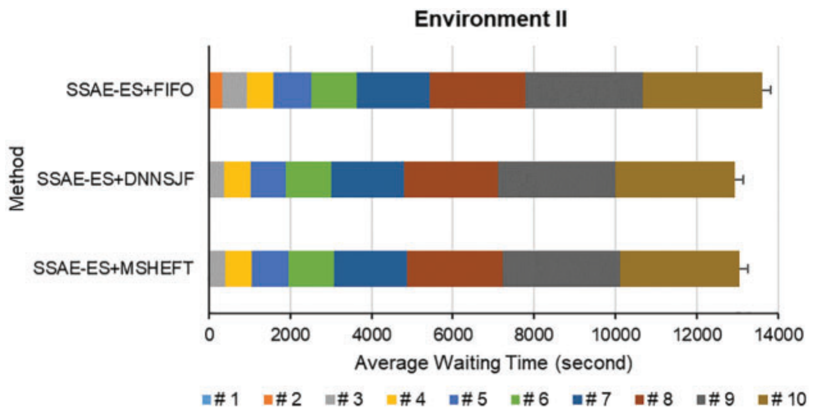


Figure 37: Job waiting time of various methods in Environment II

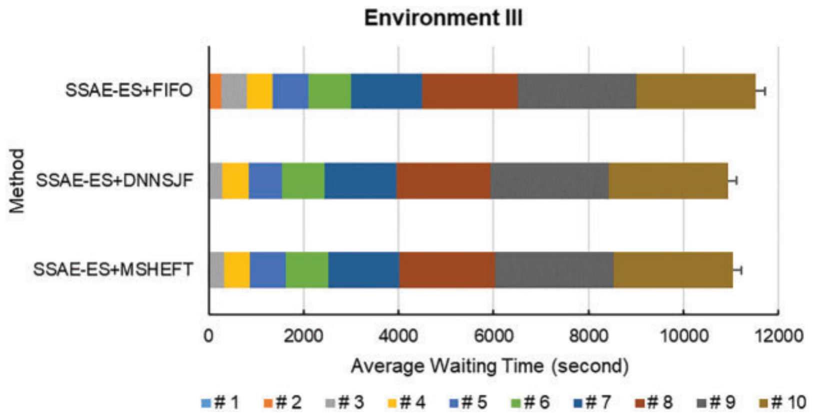


Figure 38: Job waiting time of various methods in Environment III

Table 8: Average job waiting time of various methods in Environment I

Method	Average Waiting Time (second)	Average Weighted Turnaround Time
SSAE-ES + FIFO	1576	9.51
SSAE-ES + DNNSJF	1497	9.43
SSAE-ES + MSHEFT	1510	11.17

Table 9: Average job waiting time of various methods in Environment II

Method	Average Waiting Time (second)	Average Weighted Turnaround Time
SSAE-ES + FIFO	1361	9.78
SSAE-ES + DNNSJF	1294	9.70
SSAE-ES + MSHEFT	1306	11.58

Table 10: Average job waiting time of various methods in Environment III

Method	Average Waiting Time (second)	Average Weighted Turnaround Time
SSAE-ES + FIFO	1152	10.27
SSAE-ES + DNNSJF	1094	10.18
SSAE-ES + MSHEFT	1104	12.54

4.5 Hypothesis Testing

The data tested in this study cannot be directly assumed to obey a specific distribution, and this study should test it using a nonparametric statistical method. This paper will introduce the Wilcoxon signed-rank test [42] as a hypothesis testing in this study, which is a nonparametric statistical method for testing a single sample. In Table 11, the first hypothesis testing will proceed with the Wilcoxon signed-rank test between the previous method DAE-SOLR + MSHEFT and the proposed approach

SSAE-ES + DNNSJF where test has sampled 30 valid data from Experiments I, II, and III of data retrieval. Next, the second one will proceed with the Wilcoxon signed-rank test between the previous method SSAE-ES + MSHEFT and the proposed approach SSAE-ES + DNNSJF where the test has sampled nine valid data from Experiments I, II, and III of job scheduling.

Table 11: Wilcoxon signed-rank test

Variable/Statistic	Cases in Data Retrieval	Cases in Job Scheduling
n	30	9
Sum of positive sequence	9177	348
Sum of negative sequence	0	0
T	0	0
E(T)	232.5	22.5
VAR(T)	2363.75	71.25
Z_α ($\alpha = 0.05$)	1.65	1.65
Z	4.78	2.61∗

Notes: symbol "∗" represents the value of Z^*.

Assuming a one-tailed test of $\alpha = 0.05$, the null hypothesis and the alternative hypothesis are as follows:

$$H_0 : M_P = M_C \quad H_a : M_P > M_C \tag{4}$$

In Eq. (4), M_P and M_C are the median time-consuming of using the previous method and the proposed approach. Assuming that T is the smallest of the positive and negative sorted sums, Eq. (5) expresses the following formula:

$$T = min\left\{\left|sum\ of\ positive\ sequence\right|, \left|sum\ of\ negative\ sequence\right|\right\} \tag{5}$$

$E(T)$ in Eq. (6) is the expected value of the random variable T and $VAR(T)$ in Eq. (7) the variance

$$E(T) = (1 + 2 + \ldots + n) = n(n+1)/4 \tag{6}$$

$$VAR(T) = \frac{1}{2}\left(1^2 + 2^2 + \ldots + n^2\right) = n(n+1)(2n+1)/24 \tag{7}$$

When the number of samples is proper (n $\geq$ 20), Z stands for the test statistic in Eq. (8). If the number of samples is too small (n < 20), Z^* represents the continuity correction of the test statistic Z in Eq. (9).

$$Z = {\left|T - E(T)\right|}\Big/{\sqrt{VAR(T)}} \tag{8}$$

$$Z^* = {\left|T - E(T)\right| - \tfrac{1}{2}}\Big/{\sqrt{VAR(T)}} \tag{9}$$

As a result, looking-up Z table, the Z value of $\alpha = 0.05$ is 1.65 that is less than both 4.78 (Z value) and 2.61 (Z^* value) as shown in Table 11. The decision can reject the null hypothesis H_0. That indicates that the sampling data have obtained significant results in the Wilcoxon signed-rank test for the proposed approaches in this paper.

4.6 Discussion

According to the data retrieval experiments, the SSAE-ES approach partitions the vast data to eight data zones successfully, adds sparse constraints to enhance searching capability, and implements an indexing function for the large-scale database to realize fast indexing of target data. Compared with the previous method DAE-SOLR, the SSAE-ES approach effectively improves the speed of data retrieval up to 44∼53% and increases system throughput by 44∼53%. As a result, the proposed approach SSAE-ES outperforms the other alternatives in data retrieval in the experiments. The results of the experiments showed the strengths of the proposed SSAE-ES algorithm. Besides, a statistical test using Wilcoxon signed-rank test can support the claim on improved results obtained with the proposed approach. SparkSQL performs better data retrieval than Impala or Hive in the experiments given considerable memory size.

Next, according to the job scheduling experiments, the MSHEFT algorithm performs job scheduling based on the data size for each job. It can decrease the waiting time of job execution in a queue. However, when it comes up with different jobs with the same data size, the MSHEFT algorithm proceeds with the job scheduling just like a FIFO scheduling. That is a bad situation. Therefore, the proposed DNNSJF approach can effectively overcome this problem because it can infer the different execution times for the same data size of many different jobs. Compared with FIFO and MSHEFT job scheduling algorithms, the DNNSJF approach can shorten the job's average waiting time to 3∼5% and 1∼3%, and the average weighted turnaround time by 0.8%∼09.9% and 16%∼19%. The results of the experiments showed the strengths of the proposed DNNSJF approach. Besides, a statistical test using Wilcoxon signed-rank test can present the claim confirmed on the significance test results with the proposed approach.

The experiments found that in-memory cache will lose part of retrieval data when a block of data size is more extensive than 100,000 bytes. Memcached may cause data loss when it has written a large amount of retrieval data back to the in-memory cache in a single stroke. The experiments showed the weaknesses of the Memcached used in the in-memory cache.

5 Conclusion

In this paper, the theoretical implication is to improve the efficiency of big data analytics on two aspects: speed up data retrieval in big data analytics, increase system throughput concurrently, and optimize job scheduling to shorten the waiting time for job execution in a queue. There are two significant findings in this study. First, this study explores the advanced searching and indexing approaches to improve the speed of data retrieval up to 53% and increase system throughput by 53% compared with the previous method. Next, this study exploits the deep learning model to predict the job execution time used to arrange prioritized job scheduling, shortening the average waiting time up to 5% and the average weighted turnaround time by 19%. As a result, big data analytics and its application of business intelligence can achieve high performance and high efficiency based on our proposed approaches in this study. However, the system has shortcomings about the limits of in-memory cache operations. When a large amount of data in a single in-memory cache block (more than 100,000 bytes) occurs, we will lose part of the block and not write the retrieval results entirely into the in-memory cache. Moreover, the system can only write a single stroke to the in-memory cache sequentially, i.e., time-consuming. We have to find a way to improve both problems as mentioned earlier in the future.

According to the themes discussed in this paper, there are some aspects worth exploring more profoundly in the future. The first is to find a model with better precision in the job execution time prediction. Looking for the other deep learning model, e.g., a long short-term memory is technically feasible instead of the DNN model. Secondly, you can write a JDBC interface to connect with the main program to extend the other SQL interfaces. Finally, we have to find a better solution to the limits of in-memory cache operations about the size of a block and a single sequential stroke writing to avoid losing part of blocks while writing.

Author Contributions: B.R.C. and Y.C.L. conceived and designed the experiments; H.F.T. collected the experimental dataset, and H.F.T. proofread the paper; B.R.C. wrote the paper.

Funding Statement: This paper is supported and granted by the Ministry of Science and Technology, Taiwan (MOST 110-2622-E-390-001 and MOST 109-2622-E-390-002-CC3).

Conflicts of Interest: The authors declare that there is no conflict of interests regarding the publication of the paper.

References

1. Guedea-Noriega, H. H., García-Sánchez, F. (2019). Semantic (Big) Data analysis: An extensive literature review. *IEEE Latin America Transactions, 17,* 796–806. DOI 10.1109/TLA.2019.8891948.
2. Gheorghe, A. G., Crecana, C. C., Negru, C., Pop, F., Dobre, C. (2019). Decentralized storage system for edge computing decentralized storage system for edge computing. *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, Amsterdam, Netherlands. DOI 10.1109/ISPDC.2019.00009.
3. Lee, J., Kim, B., Chung, J. M. (2019). Time estimation and resource minimization scheme for apache spark and hadoop Big Data systems with failures. *IEEE Access, 7,* 9658–9666. DOI 10.1109/ACCESS.2019.2891001.
4. Deshpande, P. M., Margoor, A., Venkatesh, R. (2018). Automatic tuning of SQL-on-Hadoop engines on cloud platforms. *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, San Francisco, CA, USA. DOI 10.1109/CLOUD.2018.00071.
5. Hadjar, K., Jedidi, A. (2019). A new approach for scheduling tasks and/or jobs in Big Data Cluster. *2019 4th MEC International Conference on Big Data and Smart City (ICBDSC)*, Muscat, Oman.
6. Sun, H., Liu, Z., Wang, G., Lian, W., Ma, J. (2019). Intelligent analysis of medical Big Data based on deep learning. *IEEE Access, 7,* 142022–142037. DOI 10.1109/ICBDSC.2019.8645613.
7. Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y. et al. (2016). Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA Journal of Automatica Sinica, 3,* 113–120. DOI 10.1109/JAS.2016.7471613.
8. Lu, M., Li, F. (2020). Survey on lie group machine learning. *Big Data Mining and Analytics, 3,* 235–258. DOI 10.26599/BDMA.2020.9020011.
9. Klinefelter, E., Nanzer, J. A. (2020). Interferometric microwave radar with a feedforward neural network for vehicle speed-over-ground estimation. *IEEE Microwave and Wireless Components Letters, 30,* 304–307. DOI 10.1109/LMWC.2020.2966191.
10. Ma, L., Bao, W., Yuan, W., Huang, T., Zhao, X. (2017). A Mongolian information retrieval system based on solr. *2017 9th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, Changsha, China. DOI 10.1109/ICMTMA.2017.0087.
11. Yan, B., Han, G. (2018). Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access, 6,* 41238–41248. DOI 10.1109/ACCESS.2018.2858277.
12. Chen, D., Chen, Y., Brownlow, B. N., Kanjamala, P. P., Garcia Arredondo, C. A. et al. (2016). Real-time or near real-time persisting daily healthcare data into HDFS and ElasticSearch index inside a Big Data platform. *IEEE Transactions on Industrial Informatics, 13,* 595–606. DOI 10.1109/TII.2016.2645606.

13. Chen, B., Liu, Y., Zhang, C., Wang, Z. (2018). Time series data for equipment reliability analysis with deep learning. *IEEE Access, 8,* 105484–105493. DOI 10.1109/ACCESS.2020.3000006.

14. Teraiya, J., Shah, A. (2018). Comparative study of LST and SJF scheduling algorithm in soft real-time system with its implementation and analysis. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Bangalore, India. DOI 10.1109/ICACCI.2018.8554483.

15. Guo, W., Tian, W., Ye, Y., Xu, L., Wu, K. (2020). Cloud resource scheduling with deep reinforcement learning and imitation learning. *IEEE Internet of Things Journal, 8(5),* 3576–3586. DOI 10.1109/JIOT.2020.3025015.

16. Yeh, T., Huang, H. (2018). Realizing prioritized scheduling service in the hadoop system. *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*, Barcelona, Spain. DOI 10.1109/FiCloud.2018.00015.

17. Thangaselvi, R., Ananthbabu, S., Jagadeesh, S., Aruna, R. (2015). Improving the efficiency of MapReduce scheduling algorithm in hadoop. *2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Davangere, India. DOI 10.1109/ICATCCT.2015.7456856.

18. Sinaga, K. P., Yang, M. S. (2020). Unsupervised K-means clustering algorithm. *IEEE Access, 8,* 80716–80727. DOI 10.1109/ACCESS.2020.2988796.

19. Marquez, E. S., Hare, J. S., Niranjan, M. (2018). Deep cascade learning. *IEEE Transactions on Neural Networks and Learning Systems, 29,* 5475–5485. DOI 10.1109/TNNLS.2018.2805098.

20. Gupta, A., Thakur, H. K., Shrivastava, R., Kumar, P., Nag, S. (2017). A big data analysis framework using apache spark and deep learning. *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, New Orleans, LA, USA. DOI 10.1109/ICDMW.2017.9.

21. Lee, Y. D., Chang, B. R. (2018). *Deep learning-based integration and optimization of rapid data retrieval in Big Data platforms* (*Master Thesis*), Department of Computer Science and Information Engineering, National University of Kaohsiung, Taiwan. DOI 10.1155/2021/9022558.

22. Chang, B. R., Lee, Y. D., Liao, P. H. (2017). Development of multiple Big Data analytics platforms with rapid response. *Scientific Programming, 2017,* 6972461. DOI 10.1155/2017/6972461.

23. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-Qaness, M. A. et al. (2021). Aquila optimizer: A novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering, 157,* 107250. DOI 10.1016/j.cie.2021.107250.

24. Abualigah, L., Abd Elaziz, M., Sumari, P., Geem, Z. W., Gandomi, A. H. (2022). Reptile search algorithm (RSA): A nature-inspired meta-heuristic optimizer. *Expert Systems with Applications, 191,* 116158. DOI 10.1016/j.eswa.2021.116158.

25. Abualigah, L., Diabat, A., Sumari, P., Gandomi, A. H. (2021). Applications, deployments, and integration of Internet of Drones (IoD): A review. *IEEE Sensors Journal, 21(22),* 25532–25546. DOI 10.1109/JSEN.2021.3114266.

26. Liu, J., Li, C., Yang, W. (2018). Supervised learning via unsupervised sparse autoencoder. *IEEE Access, 6,* 73802–73814. DOI 10.1109/ACCESS.2018.2884697.

27. Karacan, L., Erdem, A., Erdem, E. (2017). Alpha matting with KL-divergence-based sparse sampling. *IEEE Transactions on Image Processing, 26,* 4523–4536. DOI 10.1109/TIP.2017.2718664.

28. Chang, B. R., Tsai, H. F., Lee, Y. D. (2018). Integrated high-performance platform for fast query response in Big Data with hive, impala, and SparkSQL: A performance evaluation. *Applied Sciences, 8(9),* 1514, 26. DOI 10.3390/app8091514.

29. Topcuoglu, H., Hariri, S., Wu, M. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems, 13(3),* 260–274. DOI 10.1109/71.993206.

30. Carra, D., Michiardi, P. (2016). Memory partitioning and management in memcached. *IEEE Transactions on Services Computing, 12,* 564–576. DOI 10.1109/TSC.2016.2613048.

31. Ali, A. M., Farhan, A. K. (2020). A novel improvement with an effective expansion to enhance the MD5 hash function for verification of a secure e-document. *IEEE Access, 8,* 80290–80304. DOI 10.1109/ACCESS.2020.2989050.
32. Verma, C., Stoffová, V., Illés, Z., Tanwar, S., Kumar, N. (2020). Machine learning-based student's native place identification for real-time. *IEEE Access, 8,* 130840–130854. DOI 10.1109/ACCESS.2020.3008830.
33. Lin, Y. C. (2021). World-famous books. https://github.com/did56789/World-famous-books.git.
34. Lin, Y. C. (2021). Production machine load data. https://github.com/did56789/Production-machine-load.git.
35. Lin, Y. C. (2021). Semiconductor product yield data. https://github.com/did56789/Semiconductor-product-yield.git.
36. MOTC, Central Weather Bureau (2021). Rainfall and temperature data. https://www.cwb.gov.tw/V8/C/C/Statistics/monthlydata.html.
37. Taiwan Power Company (2021). Livelihood electricity data. https://www.taipower.com.tw/tc/page.aspx?mid=5554.
38. EU Open Data Portal (2021). The forest flux station data. https://data.europa.eu/data/datasets/jrc-abcis-it-sr2-2017?locale=en.
39. Lin, Y. C. (2021). Traffic violations accidents data. https://github.com/did56789/Traffic-violations-accidents.git.
40. Centers for Disease Control and Prevention (2021). Nutrition, physical activity, and obesity-behavioral risk factor surveillance system. https://chronicdata.cdc.gov/Nutrition-Physical-Activity-and-Obesity/Nutrition-Physical-Activity-and-Obesity-Behavioral/hn4x-zwk7.
41. Lin, Y. C. (2021). Airport flight data. https://github.com/did56789/Airport-flight-data.git.
42. Zhang, Z., Hong, W. C. (2021). Application of variational mode decomposition and chaotic grey wolf optimizer with support vector regression for forecasting electric loads. *Knowledge-Based Systems, 228,* 107297. DOI 10.1016/j.knosys.2021.107297.