SN Computer Science (2024) 5:732
https://doi.org/10.1007/s42979-024-03057-z

ORIGINAL RESEARCH

FLORUS: An Efficient Big Data Framework for Telecom Customer Behavior Analysis

Hong-Phuc Vo[1,2] · **Khoa-Gia-Cat Nguyen**[1,2] · **Thanh-Van Le**[1,2]

Received: 30 March 2024 / Accepted: 14 June 2024

Abstract

The rapid development of telecommunications services is increasingly attracting millions of users due to the convenience of interaction, promotion and communication. The abundance of daily transaction information has led to the creation of large data sources that are collected over time. This data source is a valuable resource for analyzing and understanding user habits and needs, devising a strategy to maintain and attract potential customers. Therefore, it is necessary to have a suitable system capable of collecting, storing and analyzing large datasets with efficient performance. In this article, we introduce Florus, a big data framework based on Lakehouse architecture, which can tackle these challenges. By applying this framework, we are able to propose an approach to analyzing customer behaviors in the telecommunication industry with a large dataset. Our work focuses on specific analysis of a huge volume of data presented in tables of different schemas, reflecting the business operation over time. Clustering based on the Bisecting K-Means algorithm will support the exploration of customer segments varying in density and complexity, and then characterize them into homogeneous groups to gain a better understanding of the market demand. Furthermore, the enterprise can forecast the revenue income at different levels, which can be applied to every customer. The work was tested with the Gradient Boosted Tree at the end of a data enriching and transformation pipeline. Overall, this work highlights the potential of Florus in supporting customer analysis experiments. Implementing the framework would significantly enhance our ability to conduct comprehensive analyses across the entire customer lifecycle.

Keywords Big data framework · Distributed system · Data analysis pipeline · Customer behavior analysis

Introduction

The greater the need for business growth, the more comprehensive the customer relationship management becomes, and the structure expanding on a massive dataset becomes more complex and diverse. The current big data wave makes the cost of collection lower and provides motivation to analyze those data with complex structures [13]. Taking the telecom data as one representative of our various use cases, this industry witnessed outstanding growth in revenue along with the diffusion of telephone service and internet access across geographics and demographics around the world [17][12]. Therefore, the telecom data landscape expands day by day and reveals the nature of the intricate background of customer segments [29]. These prosperous and valuable resources are present in the shape of a large heterogeneous dataset, making the traditional approaches ineffective and infeasible to process.

Traditionally, the business owner uses a data warehouse to process data into a centralized warehouse, which costs the loading power on query and data management efforts for a variety of data formats. They either can use a data lake to overcome the price of storage and perform advanced analytics, but practically implemented in a two-tier architecture. The inconsistency across the two systems raises the cost

This paper is an extension of work originally presented in The First International Conference on Intelligent Systems and Data Science, in CCIS 1950 (Springer, Singapore), pp. 91–105, 2023 [26].

✉ Thanh-Van Le
ltvan@hcmut.edu.vn

[1] Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam

[2] Vietnam National University Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam

[1] Statistic on number of phone subscriptions available at https://www.gso.gov.vn/px-web-2/?pxid=V0924.

[2] Statistic on Vietnam national telecom revenue available at https://www.gso.gov.vn/px-web-2/?pxid=V0923.

Published online: 27 July 2024

and challenges of data management [15]. Mazumdar et al. [4] stated a data lakehouse is a type of data architecture that combines the desirable attributes of a data warehouse and a data lake while overcoming the issues. The advantages include transactional support, open data, no copy, data quality and governance, schema management and scalability.

Therefore many enterprises join the race of this lakehouse service provider for businesses to analyze big data, led by Databricks and Synapsic Azure. Both of these platforms must be hosted on the cloud and purchased the infrastructure from these providers for commercial usage. Companies are paying the fee for the feature but may risking their own policy and customer privacy. In this paper, the Florus framework shares the principle of the lakehouse architecture design to overcome the limitation of huge cost demand for big data, which is generated with great complexity and diversity in time. Florus will transfer the ownership of data rights to the local infrastructure of the business. The cost of purchasing the platform then be reduced and replaced by free access to multiple open-source components. In our specific use case, Florus is implemented with an exposed interaction interface for low-code users to perform analysis for the big dataset. Our first approach applied in the customer behavior analysis is to formulate components and design towards the business requirements.

When seeking a business enhancement strategy, corporations have found that customer behavior analysis is the key to maintaining and developing their relationships with customers, fundamental to driving business performance [32]. Considering this analysis in the aforementioned domain, telecom research is mainly focused on the churn proportion to win a larger market share than competitors. These analyses [2, 7, 18, 27, 30] expose reasons, then alert the provider about the tendency of the user to terminate service. In 2019, Ahmad et al. [1], succeeded in solving this issue under the big data scenarios to gain high accuracy of 93.3% for AUC. Contrary to business objective alignment, solving this issue under big data scenarios has not fulfilled the purpose [9] but this is one step closer to the individual customer care [6].

Given the large scale dataset, the telecom industry demands a detailed, low bias, low error solution for every customer. This is not only about providing insights to enhance the customer experience but also providing a general conclusion for a higher level of customer management on strategy motivation. Wisesa et al. presented the prediction analysis using Gradient Boosted Tree for the business-to-business telecom sales [31]. Wang et al. yielded a low error (0.4%−1.8%) on forecasting revenue of provinces in China by series of 24-month income [28]. This model serves the highest level of strategy for the business on setting total goal income. All in all, these studies about telecom customer behavior have not yet focused on customer spending for millions of users. This study inspires the motivation for our analysis on predicting sales for customers [26], serving in different target sectors. When severely reshaping the data by aggregation, it is possible to lose the granular thread to detain the individual user. However, compared to the perspective of the complexity of the dataset, the vector embedding or principal component analysis can be used to reduce the dimension for 100,000 telecom customers from 220 features to 20 features [3].

As the amount of processed data increases, the traditional implementation for machine learning suffers the bottleneck or exceeds the upper bound of the memory. In our experiment, due to the limitation of a single computer infrastructure, we were only able to handle a table with 2 million rows x 4 columns, which is much smaller than the size of this industry's data. Three solutions to continue working with this approach are sampling the dataset to reduce the number of records or scaling up/out the system. While sampling risks the value of unmanipulated information, scaling up faces the infeasibility of physical computing resources. Therefore, scaling out the system and using the big data framework to construct a stronger cluster seems the best approach.

In this article, we introduce Florus, a big data framework designed based on the Lakehouse architecture. All necessary services have been designed and developed to perform tasks in data collection from multiple sources, storage, visualization and analytical processing, ensuring efficient performance for application in big data analysis problems. Specifically, we have applied Florus in predicting future charged amounts for every user based on their usage. To tackle the imbalanced dataset, the regression analysis will be conducted on groups of segments, where the segment is a cluster of similar users. In the application, to feature the decision support system, we provide a classification model, with its outcome invoking the corresponding revenue prediction model. Gradient-Boosted Trees Regression is our baseline model, and MAE metric is used to evaluate our work. Other measures will also be applied due to the specific characteristics of this heterogeneous dataset such as Inertia, Silhouette score, along with other well-known criteria for the model like accuracy, precision, and recall.

The paper is organized in the following structure. Section Background describes the background of the Lakehouse architecture and the core component structuring this framework. This section also includes the fundamentals of machine learning analysis with the corresponding evaluation metric for the pipeline. The architecture and mechanism inside the framework from Sect. Florus: A Lakehouse Framework for Handling Large Dataset reflects our application of the Lakehouse principles. These components will be separated into blocks and provide the interaction set, as well as a set of data resources to control independently. In Sect. Proposed Data Analysis Pipeline, we propose a

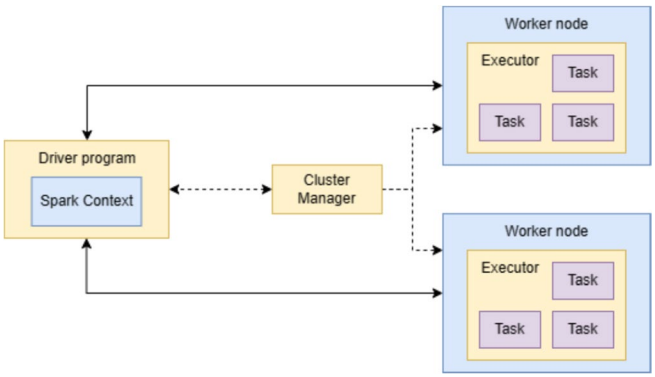


Fig. 1 Architecture of a Spark application [23]

pipeline for analysis of telecom user behavior based on big data techniques. Section Processing Experiment describes the experiment results on how this framework applies to the telecom use case. In particular, Sect. Evaluation of pipeline in telecom's decision support system evaluates the application process and testing on the unknown dataset for the scenario of the decision support system. Finally, Sect. Conclusion summarizes all the conducted results and describes future work.

Background

Lakehouse Architecture

Lakehouse is a new architecture that combines the advantages of data warehouses and data lakes to address the shortcomings of each architecture. Lakehouse enables the possibility to deploy many data structures not only at a low cost in an open format but also with powerful data management features similar to data warehouses [4].

The following are Lakehouse's main characteristics:

- Transaction support: Since concurrent reads and writes are frequently needed in real-world data streams, it is important to provide the ACID property for transactions to guarantee consistency when several parties read or write at the same time. write information.
- Schema management: Data warehouse-like schema structures, including the star or snowflake schema, are supported by Lakehouse.
- Accommodates streaming and unstructured data: Lakehouse supports semi-structured and unstructured data, including text documents, PDF files, system logs, audio and video files, and more, in contrast to data warehouses that can only manage structured data. Furthermore, streaming data from devices like IOT can be supported by Lakehouse.
- Separation of storage and compute resources: Similar to some data warehouses, Lakehouse uses separate clusters of resources for storage and computing. This ensures better scalability: Multiple applications and users can run concurrent queries on separate compute nodes while directly accessing the same storage.
- Support for a wide variety of workflows: Including data science, machine learning, SQL, and data analysis.

Apache Spark Framework

Apache Spark is a multi-language engine for executing data engineering, data science, and machine learning on a single-node machine or a cluster. It is an open-source architecture that implements the MapReduce model along with Hadoop Distributed File System (HDFS). While HDFS is used to store the result of each MapReduce phase in Hadoop, Spark maintains it in memory [20] to make it run significantly faster and consume less storage.

A spark application architecture is also designed to follow the Master/Slave concept where the master is called the driver and the slave is called the executor. When an application is started, the driver first creates a Spark Context, which acts like a gateway to access all functionalities of Spark, to connect to its cluster manager such as Yarn, Meros or Kubernetes depending on how a Spark cluster is deployed. Then, it will request the cluster manager resources and allocate some executors in the worker node [20, 22].

Figure 1 shows the architecture of a Spark application where the dashed line shows the process to request resources, allocate the executor, and the solid lines show the process of passing data through the driver and executor as tasks.

Apache Spark unified our data analytic process, and also a general-purpose framework for applying our findings to operational product data applications [19].

Spark MLlib

Spark MLlib is an open-source distributed machine learning library, which provides efficient functionality for a wide range of learning settings and includes several underlying

statistical, optimization, and linear algebra primitives. Shipped with Spark, MLlib supports several languages and provides a high-level API that leverages Spark's rich ecosystem to simplify the development of end-to-end machine learning pipelines [16]. Assefi et al. in [5] the outperformance of Apache Spark MLlib compared to Weka,[3] while remaining accurate. This library offers fast, flexible, and scalable implementations of a variety of machine learning components for distributed processing.

The MLlib library provides various API functions related to data processing, of which there are 3 main groups that we mention in this study such as Classification, Regression, and Clustering.

Cluster Analysis

Cluster analysis or simply clustering is the process of partitioning a set of data objects (or observations) into subsets [10]. Each of these subsets contains similar objects, whose similarities are different from the other groups. In clustering for big data, Pyspark implements the *KMeans* and *KMeans*|| (parallelized version of *KMeans* + +). A K-means model or any other clustering analysis can be evaluated by the most common metrics, which are the Silhouette score and Inertia score:

– **Inertia**: Inertia measures how internally coherent clusters are.

$$Inertia = \sum_{k=1}^{K} \sum_{x_i \in C_k} distance(x_i, c_k)^2 \quad (1)$$

Where:

- C_k: is the K^{th} cluster
- x_i: is the i^{th} point in the C_k
- c_k: is the centroid of C_k

– **Silhouette**: Silhouette score is used to evaluate the quality of clusters created using clustering algorithms in terms of how well samples are clustered with other samples that are similar to each other.

$$s(x_i) = \frac{b_i - a_i}{max(a_i, b_i)} \quad (2)$$

Where:

- a_i is the mean distance from x_i to others point in cluster x_i.
- b_i is the mean distance from x_i to all points in the nearest cluster of x_i

Classification

Classification is the process of finding a model (or function) that describes and distinguishes data classes or concepts [12]. The prerequisites include training data and test data. They all require a label feature to predict when missing this label. Three experimented algorithms in this paper are Logistic Classification, Gradient-Boosted Trees, and Decision Tree.

Regression

Regression analysis is a statistical methodology that is most often used for numeric prediction, hence the two terms tend to be used synonymously [11]. This method visualizes the distribution trends of data. In [26], the evaluation stage used Mean Absolute Error (MAE) to compare the prediction to the actual value. Even though this metric is dependent on the unit of measuring variable, our dataset contains the actual value of zero, which is not appropriate for the MAPE (Mean Absolute Percentage Error) value [8].

$$MAE = \frac{1}{N} \sum_{i=1}^{N} \left| Predicted_i - Actual_i \right|$$

Where:

- $Predicted_i$ is the predicted value for the i^{th} data point.
- $Actual_i$ is the actual value of the i^{th} data point.
- N is the number of data points.

Florus: A Lakehouse Framework for Handling Large Dataset

Florus is a based-Lakehouse architecture framework designed for handling large datasets. We design this system to support end users to ingest data from multiple sources and then process and visualize the stored data into graphics. For machine learning demand, Florus provides the interface to read data, train models, and save the result in our environment.

The system includes the user interface, a set of microservices, and the infrastructure. This framework can apply to any dataset and does not require redesigning the architecture presented in Fig. 2.

[3] The Weka Workbench https://www.cs.waikato.ac.nz/ml/weka/.

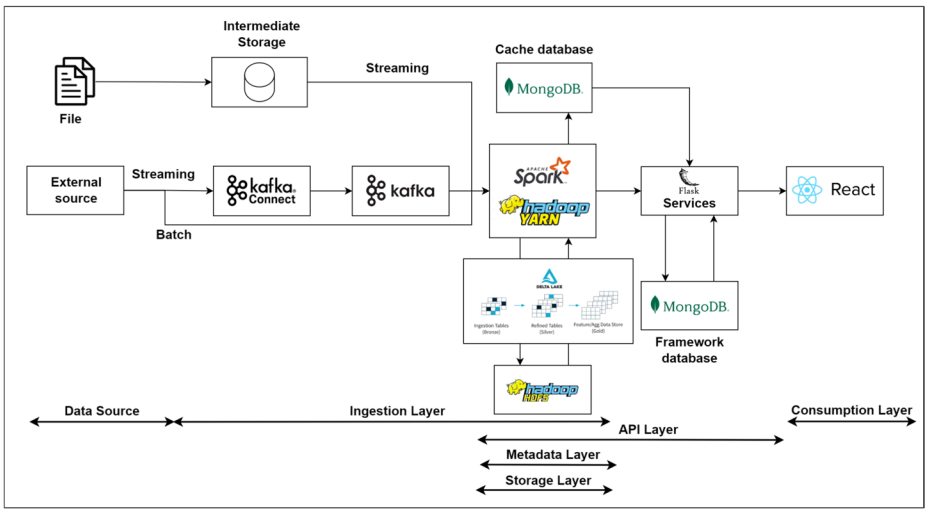


Fig. 2 Architecture of Florus framework

Overview Architecture

We proposed Florus framework with five layers, one holds a specific function in the system.

Ingestion layer - The first layer of Florus is responsible for ingesting data from multiple sources and storing it in the storage engine. We can ingest the data by two types of mechanisms:

- Batch: Scheduling an interval job to read from the database.
- Streaming: Connecting to the source database and streaming newly appended data into the storage. Kafka and Kafka Connector are decided to be used because of their power in handling a huge volume of streaming data with high throughput message delivery and minimum data loss in the pipeline. [21].

In addition, users can also upload files to our intermediate storage before streaming them into our storage.

Storage layer - This layer takes responsibility for storing all the data ingested from the external source. Due to the capacity to store data in multiple formats, scalability and cost-effective, HDFS is chosen to be the storage engine of Florus. HDFS is deployed in distributed mode, data is partitioned and replicated to help avoid loss when network or hardware problems occur.

Metadata layer - The main feature of this layer is to wrap the below storage layer, providing the ACID transaction and the other management features:

– ACID Transaction: The storage layer deployed using HDFS just only provides a set of simple operations on a file system and those operations are not atomic. Jain et al have figured out that Delta Lake makes a greater performance when compared to Apache Hudi and Apache Iceberg [14], so we decided to choose Delta Lake as a part of this layer to manage data
– Management data architecture: The Medallion architecture supports data management in this layer by three levels of cleanliness: Bronze (raw data), Silver (validated, enriched), and Gold (refined, aggregated data) table
– Caching: MongoDB is used for caching to store the content of the analyzed result to speed up the SQL query performance on HDFS.
– Framework operations: Using MongoDB to store data to operate the framework, including user information, project details, metadata of data sources, tables, and other framework-related metadata.

API layer - Flask is used in this layer due to its ease of development, testing, and scalability, maintainability in deployment. The key features of this layer are:

– Providing API endpoints to support other consumption tools for interacting with the below layer easily
– Operate the interaction between infrastructure components in the architecture

Consumption layer - ReactJS is used to provide the user interface for their interaction with the system such as setting data sources, processing data files, training machine learning models, ... In addition, we use Apache ECharts to help users represent visualization for the aggregated results.

Backend Component Design

To achieve the design architecture, we separated the backend into many components (also called service), each taking responsibility for some specific features. The component design is based on two key purposes: (i) Each component should take responsibility for one or a few features and

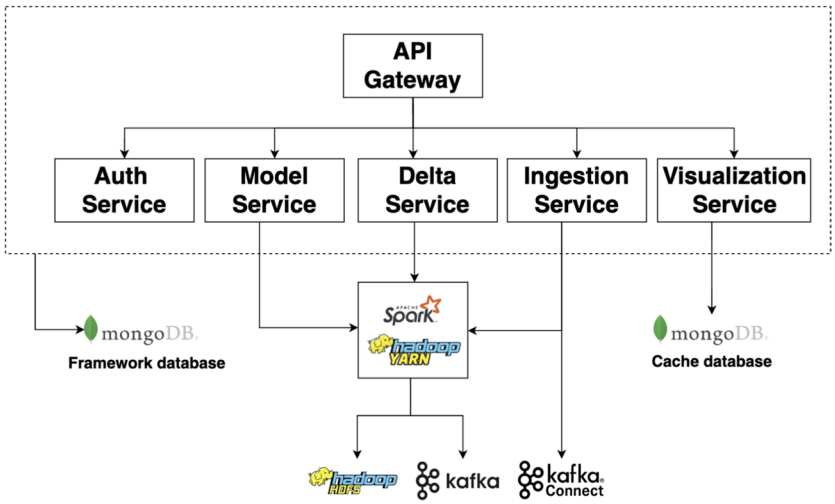


Fig. 3 Back-end component of Florus

ensure the size of the code is at just the right level, easy to read, maintain, and scale. (ii) The whole components should not depend on each other, avoiding when one component is not working as expected, it won't affect the others.

The back end contains six components as described in the Fig. 3, including:

API Gateway: Providing a gateway point for users to interact with the below system. It receives requests from user interface, then authenticates and authorizes the permission before processing

Auth service: This service is responsible for providing a mechanism to authenticate and authorize users using the architecture. Ensure data privacy between users when using the system. It controls the user information, user role, and their permission to manipulate resources in the architecture.

Ingestion service: This component is assigned two missions: (1) interact with Kafka Connect Server and Spark to create the pipeline for data from source to Bronze table storage; (2) provide the API endpoints for users to interact with ingestion sources connection. From various data sources, this framework is capable of working with both batch and streaming sources. The steps of ingesting data include:

- Setting up the schema of Bronze Table: this kind of table represents the raw data from sources. This requires a structure for each data field, such as field name, field type, and nullable flag. Therefore, the records must adhere to this established constraint.
- Ingesting data: Either batch source or streaming source selection will trigger each of their corresponding processes.
    - Batch: Spark component will start collecting data by timer setting or under a certain interval time. The data will be processed through the Delta Tables of Metadata Layer.
    - Streaming: Starting from Kafka Connectors, initiated by the Kafka Connect Server, a connection to the user's database will be settled up by the Kafka topics within the Kafka Cluster. New data at the transaction data repository will be immediately streamed into Kafka.
    - In the meantime, users can upload data files to an intermediate database. Subsequently, this data is ingested into HDFS via Spark Structured Streaming.
- Setting up schema for Silver Table: Raw data in Bronze Table is stored without cleaning, preprocessing, or refining. After any modification or transformation, data will be placed in the Silver Table, being ready for analysis. The Silver Table has a higher level of data quality and also preserves data schema (Figs. 4, 5).

Delta service: This service handles the request for data processing and retrieval. It creates Silver and Gold tables by sending refined logic to Spark, serving the caching data to enhance query performance, and providing the capability to process ad-hoc queries. As the data stream receives new data, these changes will immediately recognized and be applied the existing logic. The updated result always persist in the HDFS storage, while part of its cached for metadata layer functionality (Fig. 6).

Model service: All of the interactions with the machine learning model have to be processed under this service control. It is allowed to work with Spark and SparkML API on training, predicting and testing models. On the request to work with a machine learning model, only some of the components will be invoked.

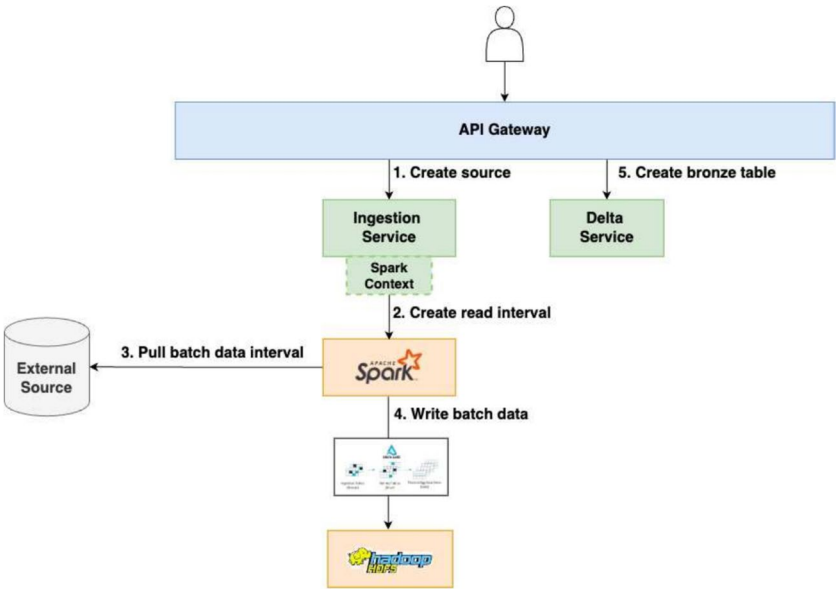


Fig. 4 Data ingestion mechanism for batch sources

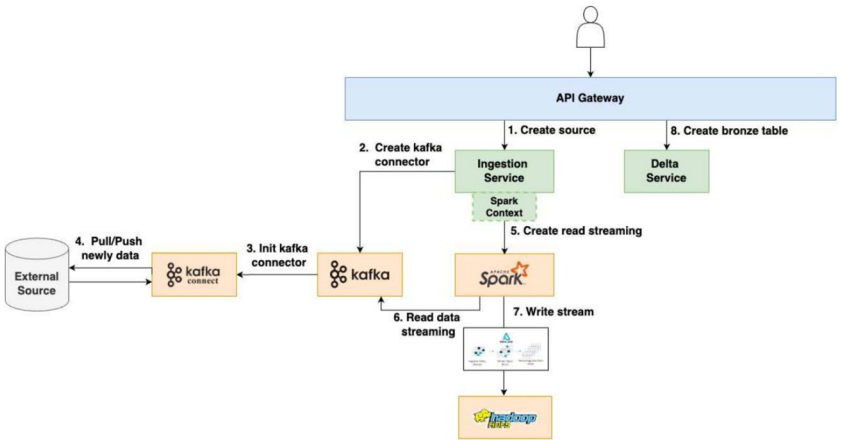


Fig. 5 Data ingestion mechanism for streaming sources

- The system will require a set of metadata about the model, which is made up of preprocessing stages, machine learning algorithm, and re-training mode.
- Interact with Spark combines Spark MLlib to support user training and use the result to predict based on their input
- Since the start of execution, any additional data added to the dataset will not be recognized and affect the outcome.
- Once the process is finished, the status will be updated on the tracking database, meanwhile all other results (model or prediction) stay in HDFS.
- After training successfully, the user can use the model to predict value or retrain with updated data (Fig. 7).

Visualization service: With the ability to contact the Delta Service for Gold Table manipulation, this service supports the demand of visualizing and data consumption for users. To enhance the speed of data query at the Gold Table, the system allows users to periodically or immediately cache the data into the intermediate database (the MongoDB component). Therefore, the time of data file accessing, and graphics processing is reduced and also promotes the role of metadata Layer of Lakehouse architecture (Fig. 8).

Infrastructure Recommendation

The Florus framework consists of multiple components, which are HDFS, YARN, Kafka, MongoDB and Back-End services. In addition, it also contains Spark History Server to track the job running state in Spark Application and Kerberos KDC to secure HDFS and Yarn.

We advise utilizing four machines for HDFS deployed in cluster mode to guarantee high availability and data integrity. By isolating the Name Node and Secondary Name Node on separate nodes, the architecture minimizes the risk of a single point of failure and reduces resource contention.

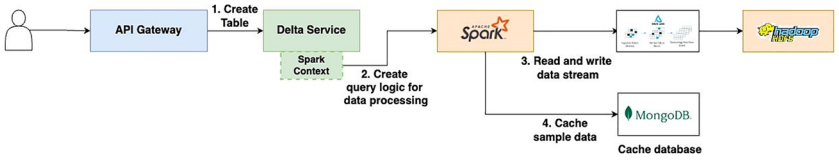


Fig. 6 Data transformation mechanism

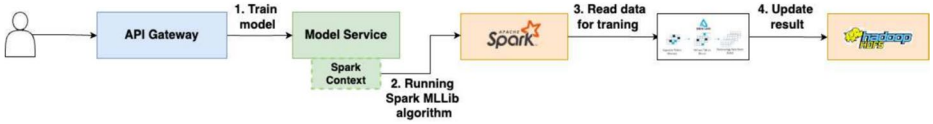


Fig. 7 Machine learning model mechanism

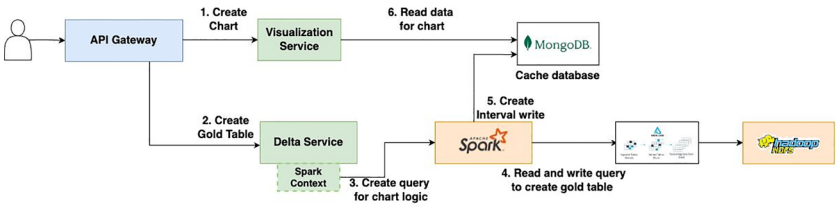


Fig. 8 Data visualization mechanism

Having at least three Data Nodes ensures data redundancy, allowing the system to withstand node failures without data loss. This configuration strikes a balance between fault tolerance and efficient resource utilization, making it suitable for a reliable Hadoop deployment. Together with HDFS, Yarn also could be deployed by pairing the Resource Manager and Node Manager with Name Node and Data Node, respectively. To guarantee that the Name Node and Resource Manager won't be overloaded, the remaining components will be distributed among the other three nodes.

Therefore, we recommend the minimum required capacity shown in Table 1. These machines should be specifically configured with the firewall, IP address, port and component installation.

Regarding resource allocation for each Spark-related service, the configuration as shown in Table 2 is optimized on the set of data over 1GB to 16GB for Ingestion Service, Delta Service, and Model Service. This configuration is suitable enough to carry out data analysis for millions of instances.

The allocation for the system as shown in Table 3 is the deployment details for the Florus system with minimum infrastructure. In corresponding, Fig. 9 demonstrates their alignment on the component functionality compared to the separated machine to be allocated. This graphic was intended to structure the interaction flow and distribute the working capacity across the nodes.

If more nodes and resources are available, each node's memory and storage can be changed for the most optimized setting. So whether the framework is deployed on a different setting or scaling up the cluster size as needed, the new alignment should follow the above-mentioned constraints. This ensures our framework adapts the internal structure to a higher cluster size to increase performance. Therefore, the changes will still maintain our system characteristics and functionality.

Table 1 Minimum infrastructure requirement of Florus framework

Metric	Value
Number of machines	4
Number of cores	16 per machine
RAM	32GB per machine
Storage	1TB per machine

Table 2 Recommended configuration for Spark application

Configuration	Key	Value
Number of executors	spark.executor.instances	4
Number of cores for drivers	spark.driver.cores	3
Number of cores for each executors	spark.executor.cores	3
Size of memory used for driver	spark.driver.memory	3 GB
Size of memory used for each executor	spark.executor.memory	3 GB
Number of partitions in RDDs returned by transformations	spark.default.parallelism	36
Number of partitions to use when shuffling data for join or aggregation	spark.sql.shuffle.partition	36

Proposed Data Analysis Pipeline

User behavior analysis is vital for the management of the telecommunication industry, being considered one of the most important factors contributing to business success. The *revenue* represents the charged amounts on phone calls and internet usage of customers, which are the two main services of this industry. This study aims to predict the spending of each user based on their recorded activities in the latest 2-month period. Even though high benefits would come for customer relationship management, inaccurate analysis can be adverse to the marketing and sales strategy. Especially when businesses witness the number potential of customers with high revenue contributions much lower than the others. Consequently, the recommendations for this group do not satisfy these customers, not only enhance the profits but also cost money and effort to lose customers.

We propose serial steps from customer clustering to revenue prediction so that the enhancement can accommodate both the high revenue and high quantity customer groups. The approach will have 2 flows: the training process for building appropriate models and the predicting process in the Decision Support System (DSS) as shown in Fig. 10.

Table 3 Recommended resource allocation on the minimum infrastructure

Component		Allocation
HDFS	NameNode	BD-Node-1
	Secondary NameNode	BD-Node-4
	DataNode	BD-Node-1, BD-Node-2, BD-Node-3
Yarn	Resource Manager	BD-Node-1
	Node Manager	BD-Node-1, BD-Node-2, BD-Node-3
	Zookeeper	BD-Node-2, BD-Node-3, BD-Node-4
Kafka	Broker	BD-Node-2, BD-Node-3, BD-Node-4
	Kafka Connect Server	BD-Node-2, BD-Node-3, BD-Node-4
MongoDB	Primary Node	BD-Node-4
	Secondary Node	BD-Node-2, BD-Node-3
	Model, Auth	BD-Node-2
Back-end	API Gateway, Delta	BD-Node-3
	Ingestion, Visualization	BD-Node-4
Kerberos		BD-Node-2

Customer Behavior Analysis

To tackle our objectives, we have carried out multiple steps to develop a set of models and their correspondences to give a thorough analysis. The following tasks will be integrated into the pipeline to improve accuracy and analytical processing across large numbers of users:

1. Pre-processing and data splitting:

   The user's revenue is first aggregated from data usage, calling by monthly and weekly logs. In addition, the user's revenue is affected by their behavior, which is usually related to their subscriptions and many other relevant uses. Each plan allows users to purchase a

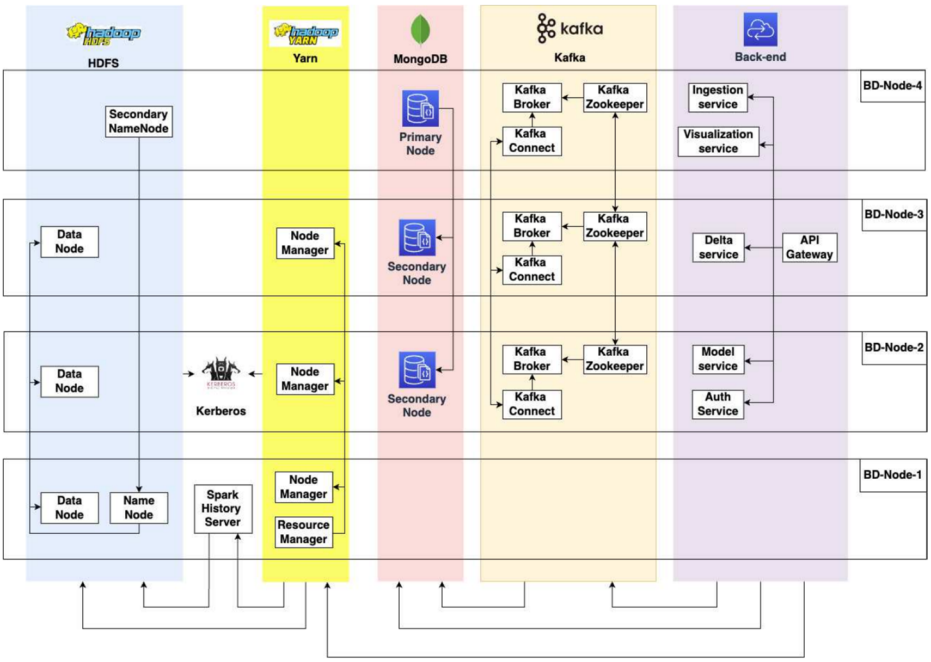


Fig. 9 Recommended resource allocation on the minimum infrastructure

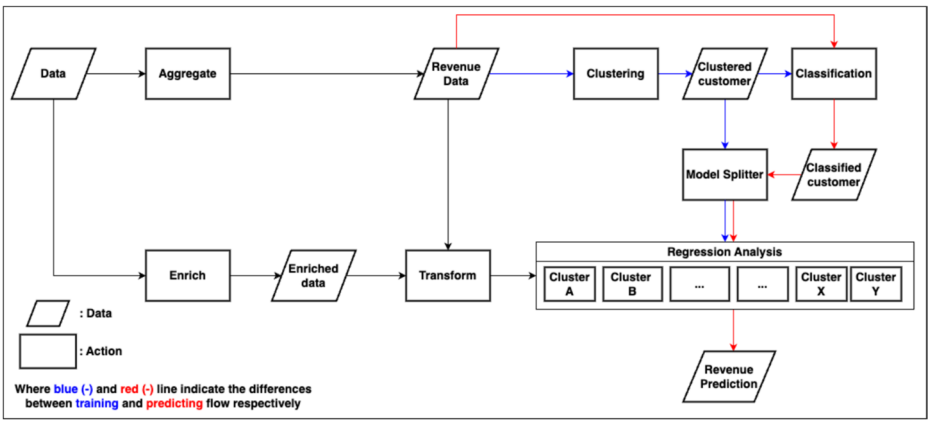


Fig. 10 Data flow representation

fixed amount of non-charge services within a specific period. Customer segments represent a set of users with similar charge fees, counted calls, and internet services,... As a result, feature selection and data aggregation help to explore valuable information to enrich the pool of data fields.
Particularly, the dataset with a high range of values will be transformed to log normalization. Code of service subscription also features in the set of information for decision. On our dataset, the abbreviation of action is:

- Aggregate: aggregate data by the time slice
- Enrich: perform feature engineering on columns
- Transform: Execute log normalization, StandardScaler, PCA [25] to fit the assumption of regression model and data characteristics.

2. Clustering:

   User clustering can bring numerous benefits to businesses. Firstly, grouping customers with similar purchasing behaviors helps businesses gain a better understanding of the needs and preferences of each customer segment. Secondly, this analysis allows businesses to save time and cost in customer management, and focus on high-potential customer segments to increase sales. In this approach, users are then clustered into groups with their different charged usage. However, the cluster size does not correlate to the revenue contribution, which is the cause of the bias in the simple approach. The purpose of this stage is to find user segments and their spending distributions, which will be an important metric to reduce data imbalance.

3. Classification:

   In general, user classification will help businesses understand their customer base better and thereby develop appropriate business strategies, marketing tactics, and customer care approaches tailored to different user groups. Regarding our suggested process, the classification model uses the outcome of the clustering stage to train and aims to classify the users into equivalent segments based on their summary usage.

4. Regression:

   Due to the difference in size and revenue of these above clusters, we extract them into multi-independent models to avoid their interference in the process of other clusters. The classification model acts as the gateway for the regression stage, so the splitting rule should take into account its misclassification behavior. Based on the confusion matrix, clusters are grouped together if they have a mutual misclassification rate of 25%. Finally, the bias is dropped down and thus the model accuracy is improved by the combination of multiple regression models.

In applications, the classification module helps us to determine which revenue prediction model should be used. In case the whole proposed pipeline has not been retrained, their historical usage is recorded but has never been clustered, this pipeline is still able to achieve the revenue prediction output. We proposed an additional scheme to evaluate when applying the model in industrial-scale scope:

1. Pre-processing: similar to the preprocessing stage of the training process.

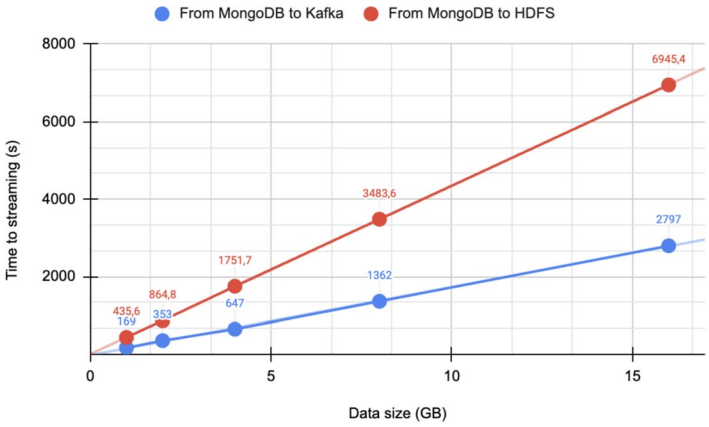


Fig. 11 Completion time for streaming sources having big size

2. Classification: the input contains less than one feature than the clustering training set and will be used to predict the segments of customers. The prediction from this model will be enriched with other features and passed to the next stage.
3. Regression: using the label classified by the previous stage, the model splitter - the component that merged customer segments into clusters based on the defined condition and built the classification model, will invoke the regression model trained for this segment to generate revenue prediction.

Data Description

In this paper, we use the same dataset as in [26]. Briefly, this dataset consists of over 550 million anonymized and daily aggregated records from approximately 7 million telecom customers. With a total of more than 70 attributes in tables, it provides information on subscriptions, phone calls, internet-charged fees, and top-up amounts over 3 months. No demographic data or raw format of user activities is included to ensure confidentiality of user identity and activity.

Processing Experiment

Regarding the Florus framework implementation, we have conducted analytical experiments on a set of individual tasks under the context of big data. Due to the gaps between the functionality of our approach on the cluster and the traditional approach on the independent machine with the same capacity, the customer behavior analysis result will focus on the result based on the framework and technology supported by components in Florus. This section reflects the power of processing data on the demand and the customization of the metric evaluation.

The experiment includes 3 tests that correspond to 3 key services for manipulating big data resources.

- Streaming capacity: The time required to stream data from Sources to the Storage layer (Ingestion service).
- Data processing performance: The execution time on transforming dataset (Delta service).
- Machine learning model training time: The time cost to train a machine learning model with MLlib API (Model service).

The results represented below take into account the setting recommendation suggested in Sect. Infrastructure Recommendation. In total, the memory resource allocated for each service is 15GB on 3 nodes.

Streaming Capacity Test

This test invokes the Ingestion service and measures the arrival time of data from the original source to our system. The streaming functionality was tested on various data sizes with the time metric represented in Fig. 11 above.

We sequentially increased the data size from 512MB to 1GB, 2GB, 4GB, 8GB and 16GB and stored it in the MongoDB database as the external source. After that, we tracked the streaming time from MongoDB to Kafka component and the final component of the streaming flow, which is HDFS. In this experiment, there were no missing rows between the source and the final storage table. The larger the file is, the longer the streaming time would take but it still follows the linear proportional relationship. It has proven that this framework can process data amount overweight the memory resource. In other words, the proposed framework can be scalable for handling the large-scale dataset.

Data Table Processing Test

After having the raw data ingested into our storage place, the analysis function of Delta Service will be applied to the dataset. Florus allows the filtering, cleaning and augmenting

Fig. 12 Execution time of common operators

Table 4 Time to complete training for 3 different models (seconds)

Number of records	KMeans	Logisistic regressions	Linear regression
1,708,670	293	905	124
3,417,340	332	1,482	185
6,834,681	441	2,526	355
13,669,363	869	5,016	417
27,338,728	5,969	*	945
54,677,458	*	*	*

*: Training job frequently met error because of exceeding executor's memory

of the Silver Table, and also delivers business-level aggregation to the Gold Table. In this test, the dataset resulting from the previous test, sizing from 0.5 to 16GB, contains 30 columns of millions of records. We will evaluate the join operator and aggregation on data.

- *Join*: Left join with the table of 17 million rows, based on the one-string column.
- *Group by* and *Aggregate*: Count the number of distinct ID over a category variable.

Figure 12 shows the efficiency of this service in joining and grouping the data, reading from HDFS and writing back to HDFS. Despite the fact that analysis in reality will be much more complex than this test, it represents the feasibility of processing data on big data tables.

Machine Learning Model Training Test

In evaluating the functionality of Model Service, the training time depends on many factors, namely the computational capability, the algorithm complexity, data size, etc... This test aims to identify the relationship of execution time on the training model versus the number of data instances. The following table represents the effect of table length on the evaluated metric. In contrast, the hyperparameters for all of the models are set to default (Table 4).

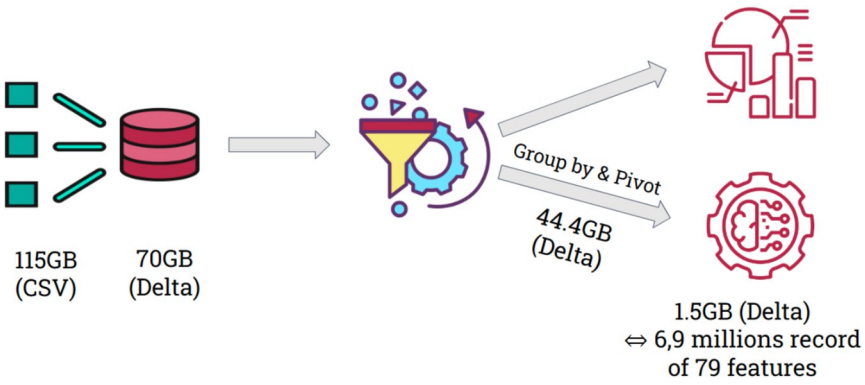
The Model service takes more time to process the larger file. The training time varies between different algorithms and increases as the data set becomes larger. Each of them may suffer the execution error on memory allocation capability at a different bound. This reveals the upper bound of modeling tasks on the Florus framework with a fixed cluster setup. The issue can be solved by adding more nodes or upgrading the physical resources to allocate more memory. Since this framework can be adapted without restructuring the architecture and data mechanism, infrastructure changes difference from the recommendation should follow the constraint in the Sect. Infrastructure Recommendation and their interactions in Fig. 9.

Discussion

Based on the above experiment result, the system can handle synthesis, data processing, and model training with the recommended infrastructure and datasets of different sizes. The Florus system can process and transfer files from outside to the internal storage system without encountering memory limit errors. Similarly, the concatenation operation on a long string data field is executed with increasing results and does not run out of memory. Tested data sets that pass the processing stage using Florus operations are still within the system's operating capabilities.

However, training machine learning models is a complex task. This task depends on many characteristics such as the type of algorithm, the parameters set, and the input data set. With the default parameter set installed on the interface, the algorithms work well when the data reaches 27 million rows. Larger data sets will require resharing resources across services to match the actual needs of the business.

Fig. 13 Data size reduction



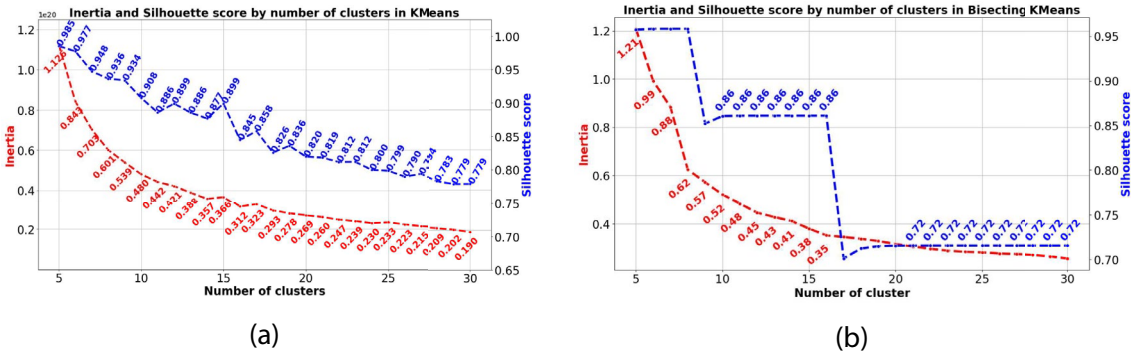


Fig. 14 Inertia and Silhouette scores by cluster numbers. **a** *KMeans*|| **b** Bisecting KMeans

Analysis Pipeline Performance

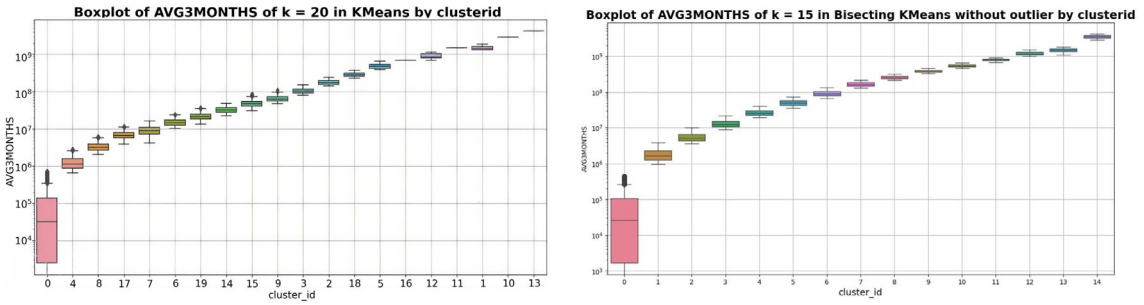
Our dataset was first obtained in the format of *.csv* files, a total of 115GB of data stored outside the system. Florus started the ingestion task to collect all data and save it into the system. Without changing the data or performing any filtering action, the total size significantly drops to 70GB in the Parquet format. The data can be refined, cleaned, and transformed into multiple silver tables, being ready to serve customer behavior analysis with chart visualization or decision support. This framework can transform the data to reshape and aggregate information for millions of instances through grouping and pivoting (Fig. 13).

This result illustrates that the Florus can work and process with our telecom dataset, and efficiently support the analysis on tables. The ingestion component proves the saving for storage size on the infrastructure without modifying or filtering data. Meanwhile, the system also has the ability to manage the tables as well as serve for processing data analysis. Last but not least, the machine learning component can work with this set of approximately 7 million users. The models we have deployed include clustering (KMeans, Bisecting-KMeans), classification (Decision Tree, Logistic Regression, Gradient-Boosted Tree), regression analysis (Linear Regression, Decision Tree Regression, GBT Regression).

While working on the models, the entire dataset will be used for the clustering. However, in the later stages, only 80% of the dataset is included in the training stage (including the train set and validate set), and the other 20% will serve as testing data for both regression and classification models. Finally, we will perform the evaluation stage for the pipeline application on the test dataset.

User Clustering

In this analysis, the quality of the clustering method will be measured by *inertia, silhouette score* and further compared by the distribution of customer average spending. Our set of evaluation metrics previously supported the data transforming and hyperparameter tuning to select the best time window of the month, as well as choosing $k = 20$ as the number of clusters. However, neither KMeans nor *KMeans*|| by MLlib can result in a stable splitting across the random centroid initialization. Then the Bisecting KMeans (a hierarchical clustering algorithm) is applied to explore further the beginning of the analysis approach. Both of the model's



(a) (b)

Fig. 15 Average revenue distribution by clusters. **a** *KMeans*|| **b** Bisecting KMeans

evaluations result in an Elbow method selection on Inertia to narrow down the search space. The silhouette score, then, consolidates the final model with the most stable option. In Fig. 14, the value of selected models will be analyzed by plotting the boxplot of their monthly spending (*AVERAGE-3MONTHS*) distribution of all clusters presented in Fig. 15.

In detail, there is a difference in the silhouette factor. The approach with *KMeans*|| witnessed the fluctuation in silhouette score, while this metric for Bisecting Kmeans dramatically changed on the segments of *k*value. This results from the effect of silhouette changes when the algorithm starts dividing dense clusters, which happened at the 9^{th} and 17^{th} bisecting. Therefore, with the Elbow point around $k = 16$, the silhouette score of Bisecting KMeans led to the k value of 15 to reduce the cost as well as keep the model stable.

Comparing the distribution of average revenue contribution for each cluster, the hierarchical clustering approach proposes a better segmentation. Firstly, the outlier points were closer to the rest of the cluster, and even disappeared at most of the segments. Secondly, the data division is better because of the less overlap in their interquartile range. Consequently, the proposed adjustment proved the improvement in stability and clustering performance, consolidating the foundation for the other stages.

User Classification

In the user classification problem, we will use the revenue of the last two months in 2022 to classify users. Their segment labels are inferred from the result of the previous stage in both versions of the clustering model.

Previously in [26], the evaluation table demonstrates the promising performance of Gradient-Boosted Trees (GBT) in highly imbalanced classification. Meanwhile, the Softmax Regression and Decision Tree algorithm failed to recognize any instance of some minority cluster, GBT does not hold the tendency to favor the majority class. This algorithm, from the Boosting groups of the algorithm, can handle the sample size bias issue [24]. The following table lists in detail the classification result of this model built on different clustering approaches.

Table 5 above shows the performance of the classification models corresponding to the clustering results of Kmeans and Bisecting Kmeans, respectively. They were ordered by the label, and consequently, we do not ensure or imply any similarity on cluster label/characteristic

Table 5 Gradient-Boosted Trees performance based on two clustering results

KMeans Dataset			Bisecting KMeans Dataset		
Accuracy = 0.9609			Accuracy = 0.9845		
Label	Precision	Recall	Label	Precision	Recall
0	0.9862	0.9979	0	0.9940	0.9994
2	0.8788	0.9063	1	0.9466	0.9321
3	0.7344	0.7460	2	0.8373	0.7520
4	0.9177	0.9059	3	0.8279	0.7483
6	0.7687	0.8666	4	0.8055	0.7417
7	0.4942	0.1341	5	0.7949	0.7427
8	0.8119	0.7398	6	0.7979	0.7246
9	0.8188	0.7879	7	0.7969	0.7969
14	0.7825	0.8801	8	0.8077	0.8400
15	0.6383	0.4138			
17	0.7670	0.8680			
19	0.5990	0.3315			

Table 6 Performance of the baseline models

Model	R^2	*MAE*
GBT Regression without clustering	67.81%	500,318
GBT Regression with KMeans clustering		397,781

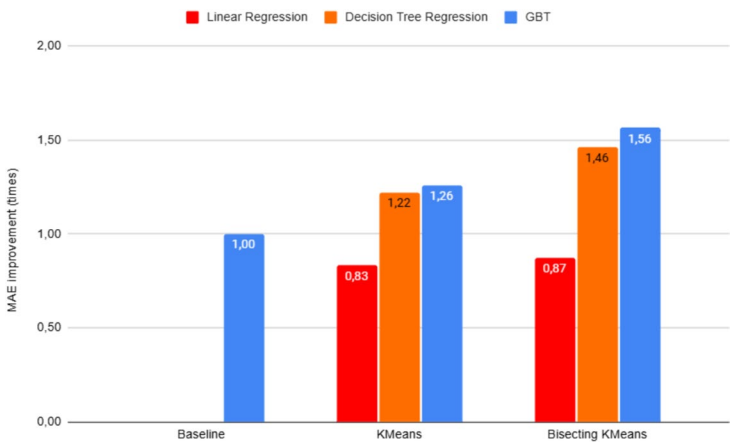


Fig. 16 MAE improvement compared to baseline

through the horizontal view. However, the results demonstrate that the performance of the classification model based on Bisecting Kmeans is better than Kmeans clustering. The updated Bisecting-KMeans based model shows stability of most clusters and viability despite unusually poor performance on some specific segments. Because the same algorithm was used, the difference in the label of instance must be crucial to the improvement, and consolidate the replacement of Bisecting-KMeans for KMeans in our use case.

Regression Analysis

Baseline Result

The most elementary approach for this problem is using a single regression model for all the users. This baseline model is only one single regression without enhancement stages conducted. The result achieved by Gradient-Boosted Trees Regression recorded by Table 6. In addition, we consider the previous result in [26] as the second baseline, to mark the changes in our enhancement proposal.

Since applying the pipeline to analyze customer behavior, the MAE metric significantly dropped by approximately 20%. This proved the effectiveness of the pipeline in segmenting customers with their shared characteristics on service usage. However, we want to broaden our approach across the different algorithms, especially since the clustering revenue distribution has overlapping areas for some neighbor clusters. This inspires us to approach clustering from another hierarchy algorithm and try to convey the gaps that may exist in KMeans initialization.

Model Splitting

Having attained the group of clusters and confusion matrix, groups that are misclassified will be merged together. It is noted that the number of groups may vary by the cardinality of segments and also the error rate of the classification model.

Data Enrichment and Pre-processing

The enriched dataset table contains 79 features about 6.9 million individual users. At the size of rows and columns, the dataset has 1 column of identify key for customers, 3 features of the revenue (with one as the dependent variable), 20 features about phone call history, 20 features of subscription, and 5 internet services columns. All data source is anonymized to ensure the customer's privacy.

Result Analysis

The final pipeline performance will be evaluated using three separate regression algorithms. For each algorithm, we trained individual models for the cluster groups and then combined them to obtain the overall Mean Absolute Error (MAE) on the validation set, which represents 10% of the data.

The enriched dataset plays an important role in this stage since it provides over 70 additional fields for feature selection. However, the optimal hyperparameters for these models will vary depending on the segment characteristics and the chosen algorithm hypothesis, meaning a single set of hyperparameters won't work across all groups or modeling choices.

Figure 16 represents the improvement of each pipeline in forecasting user spending. Corresponding to the highest improvement value, the GBT with Bisecting KMeans achieved the lowest *MAE* at **319,741**, a significant enhancement compared to our baseline result in Table 6. In other words, we achieve more than a **x1.5 times improvement** in predicting revenue from customer spending behavior.

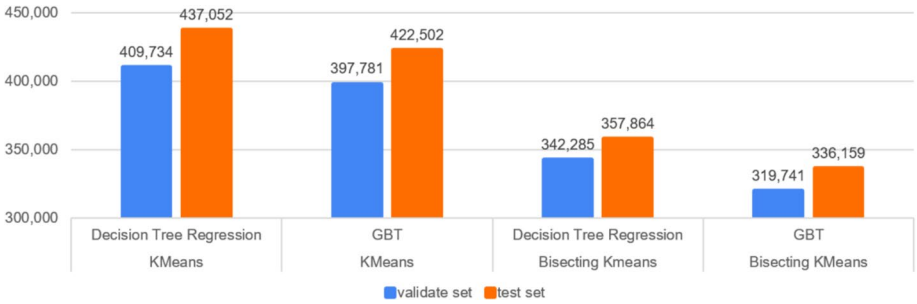


Fig. 17 MAE value on validate set and the pipeline with test set (VND)

The change in clustering stages produced positive results, leading to improved performance for all regression algorithms. This again highlights the impact of clustering segments: better customer characteristic identification leads to more accurate recommendations.

On the other hand, Gradient-Boosted Tree regression outperformed Decision Tree regression and Linear regression in our case study. Linear regression was excluded due to its poor performance on our specific dataset characteristics. Even with the logarithm transformation, Linear regression still could not achieve a lower MAE compared to the first approach described in Sect. Regression Analysis.

Overall, the experiments demonstrate Florus's success in performing basic to advanced analytics, including querying, transforming, and analyzing data with various machine learning models on large datasets.

Evaluation of Pipeline in Telecom's Decision Support System

Using the trained models and 10% of the unused dataset, we test this pipeline through the classification and the invoked regression process. This test removed the Linear Regression algorithm due to the low performance in the previous section. Figure 17 illustrates the variation of the prediction between the classification pipeline and the original training result.

Regarding the lowest MAE value of the pipeline, this chart demonstrates the GBT on Bisecting regression with enriched data as the best selection. The adjustment has significantly reduced the error to **336,159 VND** for approximately 1.5 times performance improvement. In reference, our current work shows better results compared to the previous one presented in [26] because it benefits from classification performance and clustering efficiency.

Conclusion

In this paper, we propose a big data framework for handling large scale datasets, specifically in our telecom domain analysis. Our contribution is to design the system based on lakehouse architecture fundamentals. Therefore, the enterprise can hold its own on-premise platform independently from purchasing the infrastructure and platform solution from an external provider. This can also allow achieving the lakehouse functionality without exposing the data to the physical outbound of geography. The design has shown the mechanism by which these open-source components interact to serve the complex analytic demand.

The big data framework is crucial to support the analysis in this digital era. This framework was designed in general but towards the use case of customer behavior analysis. By applying the lakehouse architecture, the system successfully inherits the advantages of both data warehouse and data lake, therefore allowing us to perform the mining process on the large scale dataset. In particular, we conduct the modeling process step-by-step with the deployed system, which requires a set of minimum infrastructure setups. The system capacity demonstration proved that this design had overcome the memory execution limit on ingesting and transforming data. However, this also reveals the upper bound of our machine learning module on handling datasets with large sizes. That will remain an issue that needs to be explored further. Even though the solution can be easily conducted on the same architecture, the framework constraints on the infrastructure allocation persist. This encourages a comprehensive guideline for industrial usage in the future.

From the perspective of Florus, it will continue to be improved on a friendly low-code platform and increase performance. Additionally, it needs to consider the possibility of automated implementation for applying big data to support demand. This work could later be more influential and comprehensive in terms of practice and expanded use cases.

On the other hand, we proposed a pipeline made up of machine learning models for analyzing telecom users' behavior. The research contribution is to enable those companies in this industry to understand their customers' activities. In addition, it not only gives a perspective on what benefits they will get but also may support a future recommendation system of appropriate service to each user. As the

dataset has millions of records, the solution is implemented in a Spark cluster and takes advantage of this framework's machine learning library.

The pipeline of clustering and enrichment divides the original data into segments and analyzes their characteristics. This improved the regression result to a 1.5x lower MAE for the monthly usage prediction. While the clustering can perform better on Bisecting KMeans, the regression shows the same pattern for a different approach and proves the suitability of our dataset over GBT algorithm. Even though the misclassified rate exists for new customers, the pipeline can still ensure the MAE is lower than our baseline value. Based on the evaluation result, this research can be featured in the Telecom Decision Support System to promote business operations.

Acknowledgements The authors acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

Data Availability The anonymized data used in this study are confidential. Due to privacy concerns, the data cannot be publicly shared. The corresponding author may be able to process additional anonymized or aggregated data summaries upon reasonable request.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

Research Involving Human and /or Animals Not Applicable.

Informed Consent All authors agreed with the content and that all gave explicit consent to submit and publish the manuscript.

References

1. Ahmad AK, Jafar A, Aljoumaa K. Customer churn prediction in telecom using machine learning in big data platform. J Big Data. 2019;6(1):28. https://doi.org/10.1186/s40537-019-0191-6.
2. Ahmed AA, Maheswari D. Churn prediction on huge telecom data using hybrid firefly based classification. Egypt Inform J. 2017;18(3):215–20. https://doi.org/10.1016/j.eij.2017.02.002 (https://www.sciencedirect.com/science/article/pii/S1110866517300403).
3. Alkhayrat M, Aljnidi M, Aljoumaa K. A comparative dimensionality reduction study in telecom customer segmentation using deep learning and pca. J Big Data. 2020;7(1):9. https://doi.org/10.1186/s40537-020-0286-0.
4. Armbrust M, Ghodsi A, Xin R, Zaharia M. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In: Proceedings of CIDR. 2021;8:28
5. Assefi M, Behravesh E, Liu G, Tafti AP. Big data machine learning using apache spark mllib. In: 2017 IEEE international conference on big data (Big Data). 2017;3492–3498. https://doi.org/10.1109/BigData.2017.8258338
6. Chen CM. Use cases and challenges in telecom big data analytics. APSIPA Trans Signal Inf Process. 2016;5: e19. https://doi.org/10.1017/ATSIP.2016.20.
7. Dalvi PK, Khandge SK, Deomore A, Bankar A, Kanade VA. Analysis of customer churn prediction in telecom industry using decision trees and logistic regression. In: 2016 symposium on colossal data analysis and networking (CDAN). 2016;1–4. https://doi.org/10.1109/CDAN.2016.7570883
8. Davydenko A, Fildes R. Forecast error measures: critical review and practical recommendations 2016. https://doi.org/10.13140/RG.2.1.4539.5281
9. Devriendt F, Berrevoets J, Verbeke W. Why you should stop predicting customer churn and start using uplift models. Inf Sci. 2021;548:497–515. https://doi.org/10.1016/j.ins.2019.12.075 (https://www.sciencedirect.com/science/article/pii/S0020025519312022).
10. Han J, Kamber M, Pei J. 10 - cluster analysis: Basic concepts and methods. In: Han, J., Kamber, M., Pei, J. (eds.) Data Mining (Third Edition), 2012;443–495. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, third edition edn. https://doi.org/10.1016/B978-0-12-381479-1.00010-1, https://www.sciencedirect.com/science/article/pii/B9780123814791000101
11. Han J, Kamber M, Pei J. 6 - mining frequent patterns, associations, and correlations: Basic concepts and methods. In: Han, J., Kamber, M., Pei, J. (eds.) Data Mining (Third Edition), 2012;243–278. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, third edition edn. https://doi.org/10.1016/B978-0-12-381479-1.00006-X, https://www.sciencedirect.com/science/article/pii/B978012381479100006X
12. Han J, Kamber M, Pei J. 8 - classification: Basic concepts. In: Han, J., Kamber, M., Pei, J. (eds.) Data Mining (Third Edition), 2012;327–391. The Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, Boston, third edition edn. https://doi.org/10.1016/B978-0-12-381479-1.00008-3, https://www.sciencedirect.com/science/article/pii/B9780123814791000083
13. Hong Y, Zhang M, Meeker WQ. Big data and reliability applications: the complexity dimension. J Qual Technol. 2018;50(2):135–49. https://doi.org/10.1080/00224065.2018.1438007.
14. Jain P, Kraft P, Power C, Das T, Stoica I, Zaharia M. Analyzing and comparing lakehouse storage systems. In: Proc. Conf. Innov. Data Syst. Res 2023
15. Mazumdar D, Hughes J, Onofre J. The data lakehouse: Data warehousing and more 2023
16. Meng X, Bradley J, Yavuz B, Sparks E, Venkataraman S, Liu D, Freeman J, Tsai D, Amde M, Owen S, Xin D, Xin R, Franklin MJ, Zadeh R, Zaharia M, Talwalkar A. Mllib: Machine learning in apache spark 2015.
17. OECD: Internet access (indicator). https://doi.org/10.1787/69c2b997-en, https://www.oecd-ilibrary.org/content/data/data-00682-en, Accessed 29 Feb 2024.
18. Olle G. A hybrid churn prediction model in mobile telecommunication industry. Int J e-Educ, e-Bus, e-Manag e-Learn (01). https://doi.org/10.7763/IJEEEE.2014.V4.302 2014.
19. Salloum S, Dautov R, Chen X, Peng PX, Huang JZ. Big data analytics on apache spark. Int J Data Sci Anal. 2016;1(3):145–64. https://doi.org/10.1007/s41060-016-0027-9.
20. Shaikh E, Mohiuddin I, Alufaisan Y, Nahvi I. Apache spark: A big data processing engine. In: 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM). 2019;1–6. https://doi.org/10.1109/MENACOMM46666.2019.8988541
21. Shree R, Choudhury T, Gupta SC, Kumar P. Kafka: The modern platform for data management and analysis in big data domain. In: 2017 2nd international conference on telecommunication and networks (TEL-NET). 2017;1–5. https://doi.org/10.1109/TEL-NET.2017.8343593
22. Sleeman WC IV, Krawczyk B. Multi-class imbalanced big data classification on spark. Knowl-Based Syst. 2021;212: 106598.

https://doi.org/10.1016/j.knosys.2020.106598 (https://www.sciencedirect.com/science/article/pii/S0950705120307279).
23. Spark A. Cluster mode overview - SPARK 3.5.1 documentation, https://spark.apache.org/docs/latest/cluster-overview.html
24. Tanha J, Abdi Y, Samadi N, Razzaghi N, Asadpour M. Boosting methods for multi-class imbalanced data classification: an experimental review. J Big Data 2020;7(1). https://doi.org/10.1186/s40537-020-00349-y
25. Uragun B, Rajan R. Developing an appropriate data normalization method. In: 2011 10th international conference on machine learning and applications and workshops. 2011;2:195–199. https://doi.org/10.1109/ICMLA.2011.53
26. Vo HP, Nguyen KGC, Nguyen KL, Le TV. A big data approach for customer behavior analysis in telecommunication industry. In: Thai-Nghe N, Do TN, Haddawy P (eds.) intelligent systems and data science. 2024;91–105. Springer Nature Singapore, Singapore. https://doi.org/10.1007/978-981-99-7666-9_8
27. Wagh SK, Andhale AA, Wagh KS, Pansare JR, Ambadekar SP, Gawande S. Customer churn prediction in telecom sector using machine learning techniques. Results Control Optim. 2024;14:100342. https://doi.org/10.1016/j.rico.2023.100342 (https://www.sciencedirect.com/science/article/pii/S2666720723001443).
28. Wang M, Wang Y, Wang X, Wei Z. Forecast and analyze the telecom income based on arima model. Open Cybern Syst J. 2015;9:2559–64.
29. Wassouf WN, Alkhatib R, Salloum K, Balloul S. Predictive analytics using big data for increased customer loyalty: Syriatel telecom company case study. J Big Data. 2020;7(1):29. https://doi.org/10.1186/s40537-020-00290-0.
30. Win NAS, Thwin MMS. Comparative study of big data predictive analytics frameworks. Ph.D. thesis, MERAL Portal 2017
31. Wisesa O, Adriansyah A, Khalaf OI. Prediction analysis sales for corporate services telecommunications company using gradient boost algorithm. In: 2020 2nd international conference on broadband communications, wireless sensors and powering (BCWSP). 2020:101–106. https://doi.org/10.1109/BCWSP50066.2020.9249397
32. Woodcock N, Green A, Starkey M, Framework™ TC. Social crm as a business strategy. J Database Mark Customer Strategy Manag 2011;18(1):50–64. https://doi.org/10.1057/dbm.2011.7