A System for Time Series Feature Extraction in Federated Learning

Siqi Wang
wangsiqi@4paradigm.com
4Paradigm Inc.

Jiashu Li
lijiashu@4paradigm.com
4Paradigm Inc.

Mian Lu
lumian@4paradigm.com
4Paradigm Inc.

Zhao Zheng
zhengzhao@4paradigm.com
4Paradigm Inc.

Yuqiang Chen
chenyuqiang@4paradigm.com
4Paradigm Inc.

Bingsheng He
hebs@comp.nus.edu.sg
National University of Singapore

ABSTRACT

Federated learning (FL), which enables collaborative learning without revealing raw data, is an emerging topic in privacy-preserving machine learning. Based on our experiences in thousands of real-world applications, time-series feature extraction plays a significant role in improving model quality. In this work, we propose a system automatically integrating time series feature extraction for training FL models. Our experiments show that by adopting time series feature extraction, the model accuracy (AUC) is improved by 3% on average, and recall is increased by 10% in recommender systems. We have open-sourced the project[1] and provided a step by step demonstration on how audiences can use our system to create their own FL pipeline that extracts time series features. [2]

CCS CONCEPTS

• **Information systems** → **Temporal data**; • **Computing methodologies** → *Distributed computing methodologies*; • **Security and privacy** → *Privacy-preserving protocols.*

ACM Reference Format:
Siqi Wang, Jiashu Li, Mian Lu, Zhao Zheng, Yuqiang Chen, and Bingsheng He. 2022. A System for Time Series Feature Extraction in Federated Learning. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557176

1 INTRODUCTION

Privacy-preserving machine learning emerges to be an important technique for real-world applications [16], especially with the increasingly stringent data protection requirements, such as GDPR [5] and PDPA [15]. Federated learning (FL) is a promising solution to tackle such privacy concerns [9–12]. In this demo, we focus on vertical FL (VFL), which uses different feature space among parties to jointly train a global model [7].

[1]https://github.com/4paradigm/tsfe
[2]Demonstration video at: https://youtu.be/UW27dWT-ays

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557176

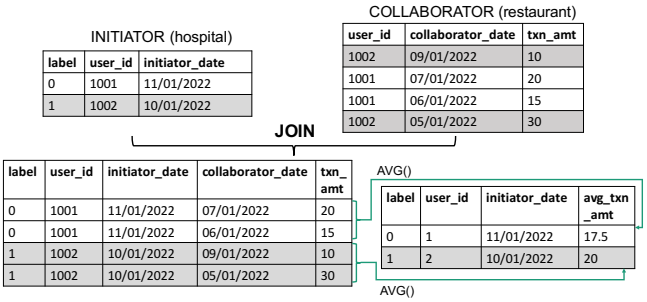

Figure 1: An example of time series feature extraction with a window size of 7-day.

This demonstration is motivated by the experiences and scenarios of 4Paradigm in providing machine learning solutions for 8,000+ customers in 12,000+ scenarios in the past 7 years. Based on our experiences in applying machine learning to thousands of real-world applications from banks, retailers, restaurants, manufacturers, healthcare systems and so on, we identify that feature engineering, especially time series feature extraction, is a must in achieving reasonable model quality in many machine learning applications [2]. With time series feature extraction, temporal information in transactional records can reveal meaningful behaviour patterns. For example, a user's purchase history in the past week is a good indication of his/her habit and can be used for better recommendations.

We illustrate the process of time series feature extraction for 2 data tables with an example as shown in Figure 1. Terms *initiator* and *collaborator* are used to represent the data tables held by two parties in VFL, while the initiator holds the label, and the collaborator holds additional information. In a normal feature engineering process, the two tables are firstly joined based on the key *user_id*. The joined table is then partitioned by the key *user_id*, which in this case generates two partitions (white and gray background). Finally, within each partition, records that has *collaborator_date* falls within 7-day of the corresponding *initiator_date* are chosen and aggregation function (average) is applied to produce the final feature result. However, raw data cannot be exchanged in federated learning due to privacy concerns. Therefore in VFL, the challenge is how to efficiently and effectively perform the action of join and aggregation on the data within collaborator, with the time window constructed within initiator.

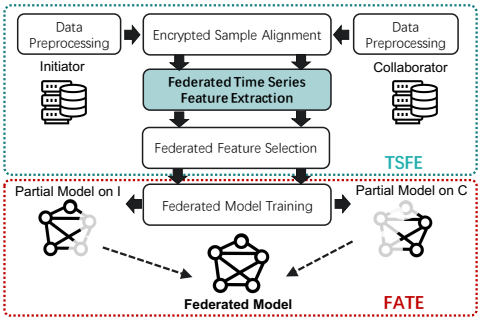

Figure 2: Federated learning workflow with TSFE.

In this work, we demonstrate a system TSFE which is designed for time series feature extraction for FL. TSFE is built based on two industry solutions: OpenMLDB [13] and FATE [4]. OpenMLDB is an open-source full-stack solution by 4Paradigm to facilitate feature engineering in machine learning. It provides a set of practices to develop, deploy, and maintain feature engineering in production efficiently and reliably. Efficient time series feature extraction is one of the most important functionalities in OpenMLDB. On the other hand, FATE is a popular federated learning framework that provides common FL functionalities based on a federated learning protocol. However, OpenMLDB is not designed for federated learning, and FATE does not contain time series feature extraction. Based on this motivation, TSFE extends OpenMLDB's time series feature extraction functions based on the FATE's federated learning protocol, and is further integrated into FATE's FL pipeline. Also, it could be quite challenging for users to perform time series feature engineering in the context of VFL. This demo offers an step by step demonstration for users to create or refit their federated learning applications to extract time series features.

The major contributions of this work are three-fold: 1) To the best of our knowledge, this is the first work to propose and adopt time series feature extraction in federated learning. 2) We propose an algorithm that performs federated table join and aggregation abiding by privacy requirements to extract time series feature features. 3) We integrate the process into the FATE framework and demonstrate that when employing time series feature extraction to a well-known FL model SecureBoost [3], the model quality (AUC) is improved by 3% on average, and recall is increased for 10% in a recommender systems.

2 DESIGN OF TSFE

Figure 2 shows the block diagram of the proposed TSFE. We first describe the overall system design (section 2.1) and tools (section 2.2), followed by the algorithmic details for the implementation (section 2.3 and 2.4).

2.1 System Design

Figure 2 shows the block diagram of the proposed TSFE. Machine learning training usually comprises of 2 stages: **data processing** and **model training**. Similarly, in a federated learning workflow, we engage TSFE for data processing and leverage FATE federated algorithms for model training.

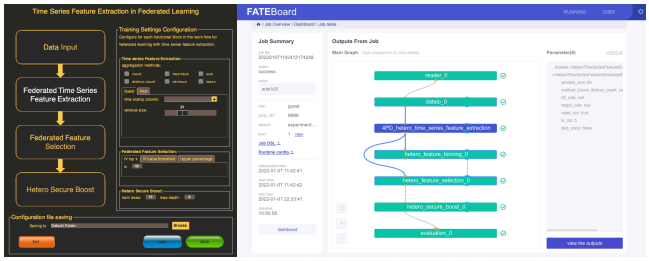

Figure 3: Screenshot of parameter UI and workflow in FATE-Board.

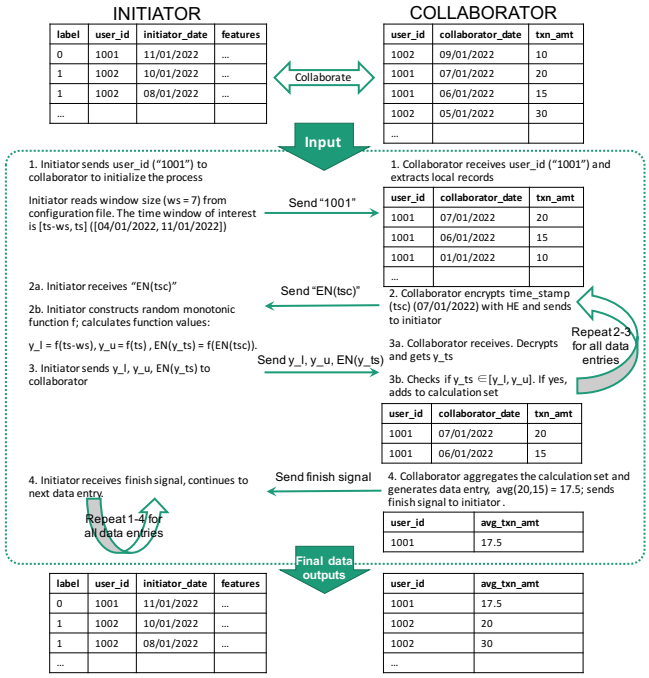

Figure 4: Illustration with an example the federated time series feature extraction process with window size = 7.

The raw data is firstly pre-processed and fed into sample alignment to obtain the overlapping samples of involved parties. The federated model is built based on those overlapping samples. After which, a stage named *federated time series feature extraction* (FTSFE) is used to perform time series feature extraction abiding by privacy requirements. With several aggregation methods agreed beforehand by all the parties, FTSFE generates multiple new features on the collaborator side. Then in federated feature selection stage, the new features together with the original features are evaluated through information value (IV) as the criterion to select good features. The processed data is then fed into FATE federated model training to obtain the final federated model. The process is fully integrated into the FATE framework. FTSFE is implemented based on OpenMLDB while the other functional blocks are modified from FATE's original components. Specifically, in FATE workflow, a domain-specific language (DSL) is adopted to describe modeling tasks to

facilitate flexible model training. Each functional block, such as data-io, feature-engineering and classification module etc, can be combined as a DAG, connected through data and model. To fit seamlessly into the existing flow, we implement FTSFE as a self-contained functional block in FATE. The output of the block is a data table that can be directly plugged into any other functional blocks in FATE.

To be more specific, comparing to a normal federated learning workflow, a **FTSFE** block (as shown in blue in Figure 2) is added to enable the generation of federated time series features. The later training processes remain the same, with the only difference being a wider data table at the collaborator side with newly generated time series features.

2.2 Tools and Usage

We also implement a simple user interface (UI) to facilitate the tuning of parameter settings. Figure 3 shows the screenshot of the parameter UI and a complete workflow in FATEBoard. The configurable variables include aggregation methods (count, sum, max, etc), time window size and hyper-parameters for model training. Users can firstly generate configuration files with the UI and engage the TSFE workflow. After the federated time series features are generated, the FATE workflow will be engaged automatically to continue the federated model training. The status of the job and results can be reviewed in real-time with FATEBoard visualizations, as shown in Figure 3.

2.3 Federated Time Series Feature Extraction

The detailed process in FTSFE is illustrated with an example in Figure 4. The input is a data table with labels in initiator, and a data table with additional features to be extracted in collaborator. The output of the block is a new data table generated within collaborator, with each data sample matched with initiator and the features aggregated according to initiator time stamp. The process describes as follows:

Step 1: The initiator, with the intention to extract time series features for a data sample with an *user_id* and timestamp, initiates the process by sending *user_id* ("1001") as the join key to the collaborator. The collaborator collects all the matched records (id == "1001") as a set.

Steps 2 & 3: For each record, the collaborator identifies through a privacy-preserving algorithm whether it falls within the required time window. The algorithm is implemented based on homomorphic encryption (HE) [1, 14] for data privacy (detailed discussion in section 2.4).

Step 4: The collaborator applies aggregate functions $AGG(f)$ (average value) to produce time series features for the data sample. Note that the generated features are stored in the collaborator side and never shared with the initiator. The protocol repeats step 1 to 4, until all data entries in initiator are processed.

In a privacy-preserving setting where two parties are sharing minimum raw information, several functions for $AGG(f)$ can be chosen from a set of aggregation functions. In our demo, users can choose several aggregation functions (see Figure 3), including count of occurrence, distinct count of occurrence, sum, min, max, mean.

2.4 Privacy-Preserving Algorithm

Steps 2 & 3 of FTSFE require the collaborator to identify whether a timestamp falls within the time window of interest. The idea is to use HE and random function to encode the raw time stamp information to ensure data privacy. We use $EN(m)$ to denote the homomorphic encrypted value of m.

First, the collaborator encrypts its time stamp with HE ($EN(tsc)$) and sends it to the initiator (step 2). Upon receiving the encrypted time stamp (step 2a), the initiator generates a random monotonic function ($y = ax + b$), and calculates the function values of its time window boundaries (y_l, y_u), and the function value of the encrypted time stamp (y_ts) (step 2b). Because of the nature of HE, the initiator can perform computations on the encrypted value without first decrypting it ($EN(tsc)^a * b = EN(a * tsc + b)$). Finally, the collaborate decrypts and gets the function value of its time stamp ($a * tsc + b$)(step 3a). By comparing with y_l and y_u, the collaborator can deduce if tsc falls within the time window of interest (step 3b). Note that for the above algorithm to work, the time stamp is converted into integer as number of seconds since origin of time.

In the proposed algorithm, all timestamp information is shared as cipher text or concealed within random functions to maximize privacy protection for both parties. No raw data is exchanged to ensure privacy for both the initiator and the collaborator. In the later stage, the initiator can use other techniques such as federated feature selection, to choose which of the features are more useful for model training without knowing the actual meaning of the features from the collaborator.

3 DEMO SCENARIO AND EVALUATION

We demonstrate TSFE with real-life scenarios through a typical VFL workflow: partitioning dataset for VFL scenarios, task configuration, hyper parameter tuning, and model accuracy evaluation. The implementation is open-sourced. Sample data and configuration files are provided to build a federated recommendation model with federated time series features. The audiences can also prepare their own data accordingly as described in this section to create their own application.

As the entire process of building a VFL pipeline is rather complicated, this demo provides a step by step guide on how to use our system to build VFL with time series features. This demo also visualizes the results and demonstrates the effectiveness and efficiency of our approach.

3.1 The Scenario

We use three real-world datasets, including two internal datasets (dataA, dataB) and an open-source dataset (dataC) [8]. DataC contains customer activity data log for the purpose of predicting customer's intention to purchase certain products. The dataset contains 4 sub tables, namely "User Data", "Product Data", "Comments Data" and "Action Data". "User Data" and "Product Data" contain attribution information including users' age, gender, products' properties, brands, etc. Both "Comments Data" and "Action Data" contain activity logs, for example user A purchases product B at time X, product C receives bad comments at time Y, etc. Both tables contain **time**

stamps which mark the time of the action. Data **label** is in "Action Data" table which marks whether the customer has made a purchase. Further details of the data features are summarized in [8].

The other two internal datasets contain data logs supplied by 4Paradigm's clients in a similar structure of multiple tables of attribution and activity logs. DataA is from a bank with user transaction logs for credit risk assessment; dataB is from a content application who intends to provide content recommendation with users' reading habits. The size of the dataset is at the scale of millions.

3.2 Partitioning Dataset for VFL Scenarios

Both parties need to prepare their dataset for VFL before passing it into the training workflow. The initiator dataset is required to at least have a **label**, an **identifier** to be used as the join key, and a **time stamp**. The collaborator dataset is required to have the same **identifier**, a **time stamp** and several additional features. After this manual preparation, the datasets can be uploaded to be engaged in later training.

Since these datasets are not naturally collected for federated scenarios, we partition them to simulate a reasonable federated training scenario. Since all three datasets are multi-table, we assign one table to collaborator, and the rest tables to initiator. The tables at the initiator are aggregated into one table through local feature engineering. We take advantage of the OpenMLDB feature engineering tool [13] to perform local feature extractions within each party. The result is one data table each in all participating parties.

For example for dataC, we assign the "Comments Data" table to the collaborator. The other three data tables are assigned to the initiator, which are further aggregated into one data table by OpenMLDB. The details of the features generated by OpenMLDB can be found in [8]. Similarly for dataA, the result initiator table has a label, *id* as the identifier and time stamp in column *f_original_eventTime_0*; collaborator table has two additional features (*action*, *subaction*), and time stamp in column *original_ts*.

3.3 Task Configuration

Initiator and collaborator are required to agree on a common workflow and parameter settings. Both parties will execute on the same work flow with same settings. For example, as shown in Figure 3, the workflow includes data input, FTSFE, federated feature selection, SecureBoost [3] for federated training, and finally model evaluation. For the FTSFE functional block, initiator needs to decide the window size that it wants to extract the time series features for, such as 1-day, 7-day, 31-day. Both parties can agree on a set of aggregation functions, choosing from {*count*, *d_count*, *sum*, *min*, *max*, *mean*}. There are also other configurations that the user can play with, including feature selection strategies and model training parameters. The UI allows users to tweak these settings. Then, the user can launch the training through FATE flow and observe the training progress and result in FATEBoard.

3.4 Model Accuracy Evaluation

The model accuracy is evaluated based on the metric of testing AUC [6]. For each dataset, the baseline setting is model training with initiator side data table only, and the proposed setting is model training with FTSFE. The settings for SecureBoost are kept the same. Figure 5 shows the results for AUC performance comparison between baseline and our proposed system. For different datasets, we are able to achieve an average AUC improvement of 3%.

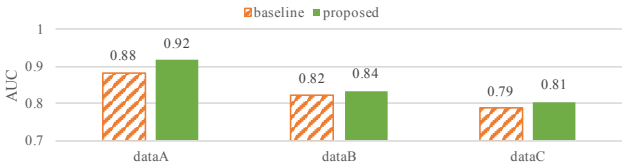

Figure 5: AUC performance for baseline and with proposed FTSFE.

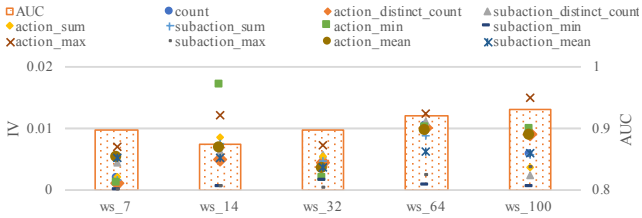

Figure 6: Feature IV and AUC with different window sizes.

While AUC acts as a good indicator for the performance of a recommender system, metrics like recall can show in better business sense how the model is doing in recommending relevant products to customers. For example in dataC scenario, with similar settings of k, the recall of the model trained with federated time series features increases as high as 10%, which effectively translates into a increase of 10% in advertisement effectiveness.

3.5 Tuning the Window Size

Tuning window sizes for time series feature extraction has great impact on model performance. Here we choose several window size settings, ranging from 7 to 100, to illustrate such impact. DataA is used in these experiments with 2 features (*action*, *subaction*) on the collaborator side. The aggregation functions are set to {*count*, *d_count*, *sum*, *min*, *max*, *mean*}, generating 11 new features in collaborator. The federated feature selection is configured to select the overall top 40 features. As shown in Figure 6, we plot the information value for each newly generated features and the final AUC result. For window size 64 and 100, the overall average IV for new features are higher, resulting in a higher model AUC performance. Thus, one can observe feature IV to further fine-tune model performance.

4 CONCLUSION

In this work, we propose a system that extends the automatic time series feature extraction function to federated learning. We implement the system and integrate it fully into the FATE system. We demonstrate the system design, an algorithm for performing federated time series feature extraction, and show that the AUC of SecureBoost can be improved on average by 3%, and recall can be increased for 10% in a recommender systems. The project is currently open-sourced. We provide a step by step demonstration as the entire process of building a VFL pipeline can be complicated. To the best of our knowledge, this is the first work to propose federated time series feature extraction and evaluate its impact.

REFERENCES

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. 2018. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. *ACM Comput. Surv.* 51, 4, Article 79 (July 2018), 35 pages. https://doi.org/10.1145/3214303

[2] Cheng Chen, Jun Yang, Mian Lu, Taize Wang, Zhao Zheng, Yuqiang Chen, Wenyuan Dai, Bingsheng He, Weng-Fai Wong, Guoan Wu, Yuping Zhao, and Andy Rudoff. 2021. Optimizing In-Memory Database Engine for AI-Powered on-Line Decision Augmentation Using Persistent Memory. *Proc. VLDB Endow.* 14, 5 (Jan. 2021), 799–812. https://doi.org/10.14778/3446095.3446102

[3] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang. 2021. SecureBoost: A Lossless Federated Learning Framework. *IEEE Intelligent Systems* 01 (May 2021), 1–1. https://doi.org/10.1109/MIS.2021.3082561

[4] FATE. 2021. Federated AI Technology Enabler. https://github.com/FederatedAI/FATE. Accessed: 2022-06-10.

[5] GDPR. 2021. General Data Protection Regulation. https://gdpr-info.eu/. Accessed: 2022-06-10.

[6] Google. 2021. Classification: ROC Curve and AUC. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc. Accessed: 2022-06-10.

[7] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).

[8] JD. 2021. JD Prediction of Purchasing Intention of High Potential Users. https://github.com/4paradigm/tsfe/blob/main/DATASET.md. Accessed: 2022-06-10.

[9] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210.

[10] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. arXiv:1907.09693 [cs.LG]

[11] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[13] OpenMLDB. 2021. An Open Source Database for Machine Learning Systems. https://github.com/4paradigm/OpenMLDB. Accessed: 2022-06-10.

[14] Pascal Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In *Advances in Cryptology — EUROCRYPT '99*, Jacques Stern (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 223–238.

[15] PDPA. 2021. Personal Data Protection Act. https://www.pdpc.gov.sg/Overview-of-PDPA/The-Legislation/Personal-Data-Protection-Act. Accessed: 2022-06-10.

[16] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2, Article 12 (Jan. 2019), 19 pages. https://doi.org/10.1145/3298981