CloudChurn: Optimizing Enterprise Customer Churn Prediction in Cloud Services for Huawei Cloud

Hengyu Ye[1], Yulong Song[1], Zhipeng Bian[2], Xiaofeng Gao✉[1], Guihai Chen[1], Xin Jin[2], and Zhenli Sheng[2]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2] Algorithm Innovation Lab, Huawei Cloud Computing Technologies Co.
{cs_22_yhy, sylacd}@sjtu.edu.cn, {gao-xf, gchen }@cs.sjtu.edu.cn
{bianzhipeng1, jinxin109, shengzhenli}@huawei.com

Abstract. Churn prediction plays a critical role in the success of cloud service providers, enabling them to identify potential churners in advance and prevent customer attrition. In cloud computing scenarios, enterprise customers constitute a significant portion of the cloud services market. They contribute substantially more revenue and exhibit much more stable subscription patterns compared to individual consumers. Therefore, it is essential and feasible to perform churn prediction and user retention, focusing on enterprise customers. This paper is grounded in actual enterprise customer churn prediction scenario in Huawei Cloud Computing Technologies Co., one of the largest cloud service providers in China. The data collected for churning behavior analysis consists of three parts: order data, webpage browsing data, and monitoring metric data. Building upon the data, we propose an effective churn prediction system that provides a comprehensive solution for enterprise customer churn prediction in cloud services. The system is based on a data-driven churn predictor, *CloudChurn*, which utilizes specially designed structures for handling distinct feature subset. It integrates these sub-models in a boosting-like manner, effectively addressing the data imbalance problem and culminating in enhanced predictive performance. The effectiveness of *CloudChurn* is demonstrated through both offline and online evaluation. Since its deployment on the ModelArts platform in June 2023, it has consistently achieved a 78.3% accuracy in recognizing potential churners, contributing to revenue retention at the million-level. This highlights *CloudChurn*'s potential as a valuable tool for cloud service providers to address customer churn and optimize their service offerings. Codes are publicly available at https://github.com/YHYHYHYHYHY/CloudChurn/tree/main.

Keywords: Churn prediction · Cloud service · Transformer · Boosting.

1 Introduction

The global cloud services market is growing rapidly with fierce competition. In China, over 45 million enterprises rely on cloud platforms for news, games, e-commerce, and online services [3]. For cloud service providers, the quantity and retention of users are vital for ensuring sustainable revenue. Retaining existing users proves more cost-effective than acquiring new ones, as statistics indicate that the cost is only 20% of

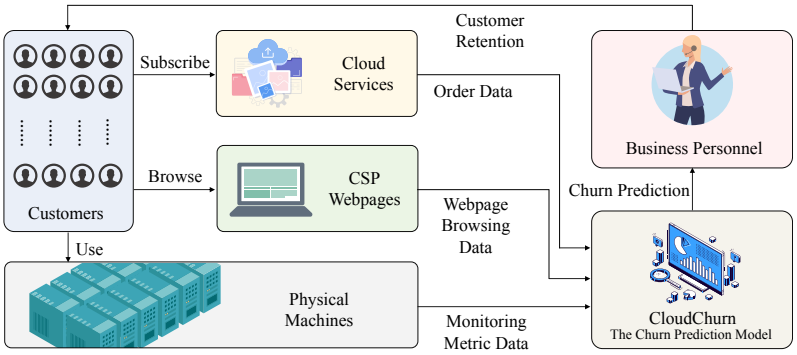

Fig. 1. The entire framework of customer churn prediction and customer retention in cloud services. Here CSP represents Cloud Sevice Provider.

acquiring new users [2]. Previous research has explored churn prediction in various domains, including telecommunication services, online games, news, social media, and finance.[9,4,18]. However, limited studies have addressed churn prediction specifically within the cloud services context. In this scenario, enterprise customers represent a substantial portion of the market, contributing significantly more revenue and demonstrating more stable subscription patterns compared to individual consumers. Consequently, there is both a necessity and an opportunity to focus on churn prediction and user retention strategies for enterprise customers.

This paper proposes an effective system for predicting churn among enterprise users of cloud services, utilizing actual customer data from Huawei Cloud. The pipeline of the entire system is depicted in Figure 1. The data includes order information, webpage browsing patterns, and monitoring metrics. The monitoring metric data covers over 60 Huawei Cloud products and more than 90 monitoring metrics, while webpage browsing data captures daily browsing duration and frequency, and order data includes subscription, renewal, change, and unsubscription information. To leverage these diverse data subsets effectively, we introduce a novel data-driven churn prediction model, *CloudChurn*, which can predict potential churn behavior for users by gathering data from these components, providing timely feedback for user retention strategies.

CloudChurn employs customized sub-models tailored for each feature subset, integrated using a boosting approach. This includes two neural networks with residual connections for order and webpage browsing data and a customized Transformer for sequential features from monitoring metrics. We implement a three-phase training process to enhance sub-model integration, effectively addressing data imbalance and achieving superior churn prediction performance compared to existing methods.

CloudChurn has been deployed on the ModelArts platform of Huawei Cloud since June 2023. It significantly improved the accuracy of identifying churning enterprise customers, outperforming the previous-deployed churn prediction algorithm by over 30% in F1-score. Over a four-month period (June to September 2023), the model achieved an average accuracy rate of 78.3%, contributing to revenue retention at million-level.

2 Data Pre-processing & Feature Extraction

This section provides an overview of the data used for churning behavior analysis. It includes three distinct categories of data, offering insights into user behavior: 1) Order data, 2) Webpage browsing data, and 3) Monitoring metric data.

Users in the dataset are all enterprises subscribing to Huawei Cloud services, which are divided into two categories: churners and non-churners. Churners are defined as users who have not engaged with any cloud services for over 30 days following the expiration of their last subscription. In subsequent sections, we will provide detailed information about each of the three data categories.

2.1 Order Data

The order data effectively captures users' subscription behavior across a diverse range of cloud services. This data offers valuable insights into the frequency and duration of their subscriptions, and the underlying patterns guiding their subscription choices. Order data includes the subscription, renewal, change, and unsubscription of cloud services, as well as information about activities of discount. In the raw order data, each entry represents a specific order.

Yearly and monthly orders, which make up over 90% of total revenue, are the primary focus in this study. These orders are subscribed on a monthly or yearly basis and are continuously deducted, mainly used by enterprise users with stable requirements. We transform this order data into numerical vectors for each user, denoted as $X_O \in \mathbb{R}^{m \times p}$, where m represents remaining time segments, and p denotes user portraits, including average amount, discount, activities, coupons, and coupon amounts associated with yearly/monthly resources at different remaining time periods.

2.2 Webpage Browsing Data

The incorporation of user webpage browsing data from Huawei Cloud's homepage, significantly enriches our understanding of user behavior. The webpage browsing data comprises four main page types: console page, product catalog page, price calculator page, and support document page. For each page type, the web page browsing statistics include the specific URLs accessed by users, browsing duration, and browsing times on the current day. It is recorded with a daily sampling frequency. However, due to the non-continuous browsing behavior of users, the web page browsing data presents substantial gaps, leading to considerable sparseness.

For each user, we obtain browsing duration and browsing times on each type of the webpages on a certain day. For a user, we have corresponding webpage browsing data:

$$X_W = \{X_{W1}, X_{W2}, X_{W3}, X_{W4}\},$$

$$X_{Wi} = \{x_{Wi}^{(1)}, x_{Wi}^{(2)}, \ldots, x_{Wi}^{(n)}\},$$

where $x_{Wi}^{(j)} = (t_i^{(j)}, c_i^{(j)})$, here $t_i^{(j)}$ and $c_i^{(j)}$ denotes the browsing duration and browsing times on pages of type i (i=1,2,3,4 indicates 4 different types of webpage) on the j days before current date.

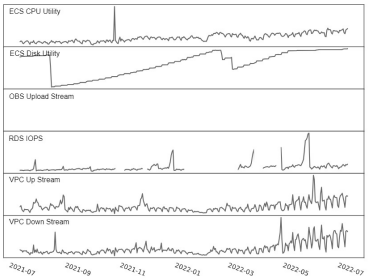

Fig. 2. Monitoring metric data.

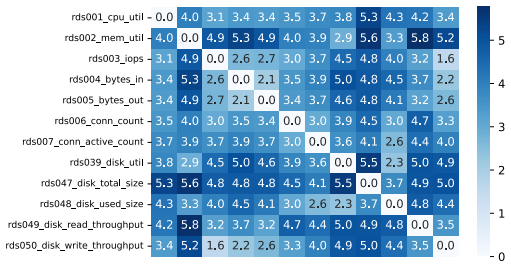

Fig. 3. DTW similarity of RDS-related metrics.

2.3 Monitoring Metric Data

The monitoring metric data includes over 90 metrics across 14 Huawei Cloud product categories, reflecting user activity. It provides each user with multi-dimensional time series data. However, not all metrics are equally suitable due to variations between products, differing units, and sampling periods. Some metrics may have significant empty data for non-subscribers, and similar metrics can introduce redundancy. To address these issues, data pre-processing focuses on eliminating inferior features and reducing time series dimensionality, enhancing information utilization for subsequent model feature processing.

Analysis of Missing Data For each monitoring metric, there are two types of missing: overall missing and partial missing. Overall missing indicates that all monitoring metrics are missing for a user within a year. This situation depends more on the difference between users' cloud service subscriptions. As shown in Figure 2, the user may have not subscribed to or used OBS-related cloud services , thus the OBS Upload Stream metric is missing all the time. Partial missing data occurs when a metric has incomplete sampling data for a user, often due to differing sampling frequencies and cloud service usage patterns. For example, Figure 2 shows missing periods for RDS IOPS.

Table 1. Selected Monitoring Metrics.

Cloud Service Type	Metric
SYS.VPC	up_stream
SYS.ECS	core_hour
SYS.ECS	cpu_util
SYS.ECS	disk_read_requests_rate
SYS.ECS	diskutil_inband
SYS.ECS	mem_util
SYS.RDS	cpu_util
SYS.RDS	mem_util
SYS.RDS	bytes_in
SYS.RDS	conn_active_count
SYS.RDS	disk_total_size
SYS.RDS	disk_used_size
SYS.RDS	disk_read_throughput
SYS.ELB	act_conn
SYS.ELB	inact_conn
SYS.ELB	in_pps
SYS.EVS	disk_read_requests_rate
SYS.EVS	disk_write_requests_rate

Analysis of similarity Over 90 product monitoring metrics from 14 cloud service categories exist, yet metrics from the same service class often reflect similar user behavior. Similar sampling methods lead to akin time series, introducing unnecessary redundancy. To address this, we use Dynamic Time Warping (DTW) [11] to measure time series similarity, accounting for variations in time slice lengths due to different sampling modes and frequencies. We calculate DTW similarity for each monitoring metric across all users and filter out metrics with small DTW distances. To consider possible similarity among metrics of the same service type, we apply 2 rounds of filtering, first for each service type and then for the selected metrics. Figure 3 displays DTW similarity for RDS cloud service-related monitoring metrics, with darker colors indicating greater differences between metrics. Using the DTW similarity matrix, we select representative indicators for inclusion in time series features. For instance, "rds002_mem_util" describes memory usage, distinct from other metrics, while "rds003_iops," "rds004_bytes_in," and "rds005_bytes_out" all relate to network throughput and are similar.

In summary, we initially choose metrics based on the ratio of missing data and then employ DTW to pick representative metrics from these, resulting in 18 representative metrics for subsequent user churn classification, listed in Table 1.

We then get the chronological features for each user from these metrics:

$$X_M = \{X_{M1}, X_{M2}, \ldots, X_{M18}\},$$

$$X_{Mi} = \{x_{Mi}^{(1)}, x_{Mi}^{(2)}, \ldots, x_{Mi}^{(n)}\},$$

where $x_{Mi}^{(j)}$ denotes the value of metric i ($i = 1, 2, \ldots, 18$ indicates the metrics) on j days before current date.

3 Proposed Methods

3.1 Input Features

The input features are extracted from the pre-processed data from the three aforementioned parts in previous section, and are devided into 3 corresponding feature subsets.

Order Features. For each user, we denote order features by $X_O \in \mathbb{R}^{m \times p}$, where m is the number of remaining time segments and p represents the number of portraits. The portraits represent the average amount, discount, number of coupons, and amount of coupons of the yearly/monthly resources with different remaining time.

Webpage Browsing Features. For each user, we denote webpage browsing features by $X_W \in \mathbb{R}^{n \times d \times 2}$, where n represents n different types of webpages, in this study we take $n = 4$, d represents the length of time span, and 2 corresponds to the browsing duration and browsing times.

Monitoring Metric Features. For each user, we denote monitoring metric features by $X_M \in \mathbb{R}^{d \times k}$, where d represents the length of time span, and k represents the number of metrics, i.e. dimension of the time series.

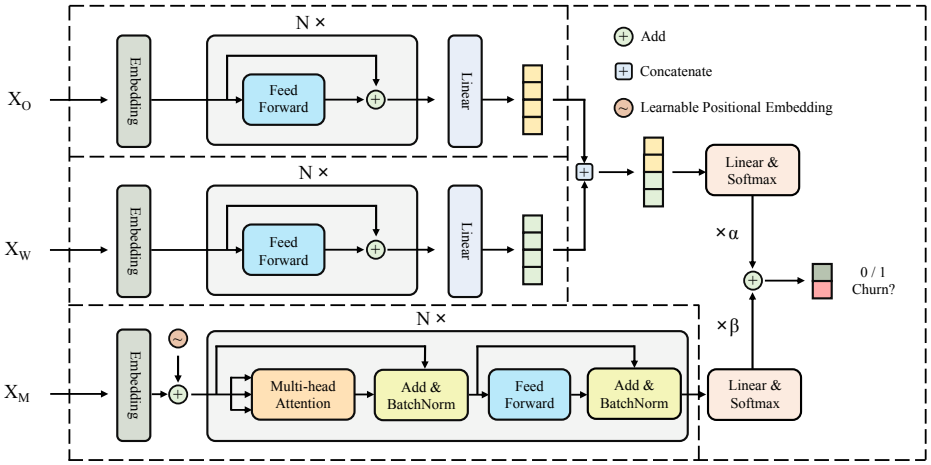

Fig. 4. Model architecture of CloudChurn.

3.2 Model Structure

As shown in Figure 4, the model comprises three main components that take the corresponding feature subsets as input, each of which can be regarded as an independent classifier. They are integrated to give the final output, a vector of length 2, denoting the probability of being churner or non-churner. In the subsequent parts, we will elaborate on the detailed model structures of these sub-models.

N_O & N_W: Base classifier with residual connections for processing order features and webpage browsing features. The model structure of N_O and N_W are kept the same, both consist of multiple linear layers with residual connections. Typically, we set the number of residual layers of both modules as 3 and utilize ReLU as the activate function. They together generate a base classification result to provide with initial churn identification, denoted as $\mathbf{y}_c$.

N_M: Customized Encoder-only Transformer for handling time series features from monitoring metric data. To adapt the Transformer for time series data, we introduce 2 modifications that have been proven effective by [19]: 1) We replace the original deterministic, sinusoidal positional encoding with fully learnable positional encoding. 2) Instead of using LayerNorm, we employ BatchNorm for all "Add & Norm" parts.

The embedded features are fed into several layers of attention layer, we set the number of attention layers as 3 and the number of multi heads as 8 by default. Finally, the results are generated with a linear projection, denoted as $\mathbf{y}_M$.

Model integration. In the preceding section, we explained how we obtain the combined result $\mathbf{y}_C$ from N_O and N_W, and the result $\mathbf{y}_M$ from N_M. To obtain the final output of the entire model $\mathbf{y}$, we combine these two results with appropriate weights:

$$\mathbf{y} = \alpha \mathbf{y}_C + \beta \mathbf{y}_M \tag{1}$$

Here $\mathbf{y}_C, \mathbf{y}_M$ and $\mathbf{y}$ are all vectors of length 2, representing the probability of churning and non-churning. The weights α and β are neither hyperparameters nor trainable pa-

rameters, they are calculated according to the performance of corresponding sub-model during the process of boosting-based integration.

3.3 Model Training

During the training period, we employ cross-entropy as the loss function to train both the entire model and its sub-models, which can be computed as follows:

$$L = \frac{1}{N} \sum_{i} - \left[y_i \cdot \log \left(p_i \right) + \left(1 - y_i \right) \cdot \log \left(1 - p_i \right) \right]. \tag{2}$$

Here, p_i represents the probability of sample i being a churner, and y_i denotes its corresponding label. The proposed model deviates from the end-to-end structure, where raw input data is directly fed into the model to produce the desired output, which is then used to compute the loss with the label. Instead, we draw inspiration from boosting and apply it to model integration. We divide the training process into 3 phases:

Phase 1: In this phase, N_O and N_W are jointly trained. Once the two sub-modules have converged, we evaluate the performance of the combined network on the entire training set. We obtain the weight of the combined network based on the error rate e, which is defined as the proportion of wrongly classified samples in the entire training set. The weight α is calculated as follows:

$$\alpha = \frac{1}{2} \ln \left(\frac{1 - e}{e + \epsilon} \right). \tag{3}$$

Here, ϵ is a small number used to handle the case when $e = 0$. We then adjust the weights of the samples from the training set according to formula 4, assigning higher weights to the wrongly classified samples. Consequently, these samples receive more attention from the Transformer Encoder during Phase 2 training.

$$w_i = \frac{e^{k_i \alpha}}{\sum_i e^{k_i \alpha}}. \tag{4}$$

In the above equation, $k_i = 1$ if sample i is misclassified, while $k_i = -1$ if sample i is correctly classified.

Phase 2: In this phase, the Encoder-only Transformer is trained using the re-weighted samples. The samples with different weights contribute differently to the total loss, which is defined as follows:

$$Loss = \sum_{i \in \mathbb{B}} w_i \times L(y_i, \hat{y_i}). \tag{5}$$

Here, $\mathbb{B}$ represents the set of all samples from the batch, w_i is the weight of sample i, y_i is the output of the Encoder network, $\hat{y_i}$ is the label of sample i, and L denotes the loss function. When the Encoder-only Transformer is trained til convergence, we calculate the weight of it β similarly

$$\beta = \frac{1}{2} \ln \left(\frac{1 - e}{e + \epsilon} \right), \tag{6}$$

where $e = \sum_{i, y_i \neq \hat{y_i}} w_i$.

Phase 3: In this phase, since we have already obtained the weights of different parts of the entire model, i.e. α and β, we apply supervised end-to-end fine-tuning on the entire network for several epochs.

In contrast to end-to-end networks that amalgamate all features into a single framework, our approach involves the deployment of distinct sub-models to address specific feature subsets and employs a three-phase training regimen leveraging boosting techniques. It not only make full use of all three distinct categories of feature, but also effectively address the negative influence of data imbalance, significantly enhancing the performance of churner recognition.

4 Experimental Evaluation

In this section, we conduct extensive experiments to evaluate the performance of *CloudChurn*. We have uploaded the code to ensure the reproducibility of our experiments [3].

4.1 Dataset and Experimental Setup

The experiments in this study utilized two proprietary datasets[4], namely Churn-1 and Churn-2, both are portions of actual enterprise customer data collected by Huawei Cloud. A summary of the overall dataset information is presented in Table 2.

Table 2. Dataset Information.

Dataset	Start Date	End Date	Number of Users	Churners	Non-churners
Churn-1	2021.07.14	2022.07.13	1342	286	1056
Churn-2	2021.11.08	2022.11.05	1944	182	1962

To select the observation window for non-churners, we employ a random process, ensuring that the time gap between the window's end date and the dataset's end date is less than 30 days. As for churners, we utilize data from the two months preceding their churning date. In this section, all experiments adopt a 10-fold cross-validation approach to assess the performance of *CloudChurn* and other benchmark models. Specifically, the data is randomly divided into 10 equal parts, ensuring an equal proportion of churners in each part. Then, for 10 iterations, one part is selected as the testing set, while the remaining nine parts serve as the training sets. This process is repeated for all models using the same splits, and the average performance is used for evaluation.

[3] https://github.com/YHYHYHYHYHY/CloudChurn/tree/main

[4] Presently, no publicly available datasets specifically target the domain of enterprise customer churn prediction in cloud services, encompassing all three essential data components—order data, webpage browsing data, and monitoring metric data. This absence can be attributed to the elevated privacy with such data types.

Table 3. Comparation of Performance.

Method		ICCP	LSTM	MLP	LR	XGBoost	RF	SVM	LDA	CloudChurn
Churn-1	Precision	0.861	0.829	0.774	0.838	0.805	**0.898**	0.843	0.292	0.830
	F1-score	0.772	0.760	0.747	0.771	0.828	0.778	0.225	0.386	**0.845**
	Accuracy	0.913	0.905	0.896	0.911	0.924	0.917	0.808	0.617	**0.933**
	AUC	0.838	0.831	0.835	0.841	0.899	0.837	0.562	0.599	**0.909**
Churn-2	Precision	0.628	0.000	0.458	0.551	0.629	**0.875**	0.000	0.233	0.631
	F1-score	0.402	0.000	0.436	0.509	0.532	0.327	0.000	0.331	**0.613**
	Accuracy	0.917	0.906	0.896	0.915	0.923	0.925	0.906	0.781	**0.930**
	AUC	0.644	0.500	0.687	0.719	0.723	0.601	0.500	0.689	**0.781**

4.2 Benchmarks & Evaluation Metrics

The benchmarks encompass Internet Card Churn Prediction (ICCP) [14], Long Short-Term Memory (LSTM) [1], Multi-layer Perceptron (MLP) [13], Logistic Regression (LR) [8,5,10], eXtreme Gradient Boosting (XGBoost) [13,7,12], Random Forest (RF) [17], Support Vector Machine (SVM) [6,15], and Linear Discriminant Analysis (LDA) [16]. Notably, ICCP (Internet Card Churn Prediction) [14] is a comparable churn prediction model recognized for its use of distinct sub-models to handle various data segments. The inclusion of ICCP serves as a valuable means to evaluate the efficacy of our model integration. To ensure a fair and unbiased comparison, all benchmarks utilize identical feature inputs as *CloudChurn*.

For the evaluation of classification performance, we employ four widely recognized metrics: Precision, F1-Score, Accuracy, and AUC (Area Under the Curve). In the practical churn prediction scenario within cloud services, achieving good performance in both precision and recall is crucial. A low precision score can erode the confidence of staff responsible for user retention in the model's prediction results. Likewise, a low recall score can result in a failure to identify a significant number of churners in time, leading to revenue loss. In this context, the F1-score emerges as the most vital evaluation metric since it comprehensively considers both precision and recall, thus provides a holistic assessment of the model's predictive performance.

4.3 Performance Comparison

We first exam the overall performance of our proposed *CloudChurn* and other benchmarks. Specifically, *CloudChurn*, ICCP and LSTM are implemented in PyTorch, and the rest benchmarks are implemented in Scikit-learn. The results are listed in Table 3.

Key observation from the results are listed as follows:

1. *CloudChurn* outperforms other benchmarks in terms of F1-score, Accuracy, and AUC in both datasets.
2. In terms of Precision, several benchmark models, including ICCP, Logistic Regression, Random Forest, and SVM, outperform *CloudChurn* in the Churn-1 dataset. These models demonstrate relatively accurate predictions of churners but struggle to identify true churners, often misclassifying them as non-churners in numerous cases. This is intolerable in the practical churn prediction scenario since most

churners are wrongly classified as non-chunrers, which may result in a loss of revenue because of not recognizing potential churners in time.
3. In the Churn-2 results, we observe that both SVM and LSTM obtain a precision and F1-score of 0, indicating that they predict all samples as non-churners. Despite achieving good accuracy, this can be attributed to the high proportion of non-churners in the dataset.

4.4 Ablation Study

We conducted an ablation study to gain valuable insights into the impact and effectiveness of individual components within our proposed method on the Churn-1 dataset. To facilitate this analysis, we designed 5 distinct models, each with a different configuration: *CloudChurn*-1 retains N_M only. *CloudChurn*-2 retains N_O only. *CloudChurn*-3 retains N_W only. *CloudChurn*-4 retains only the combined network of N_O and N_W, focusing on their integration. *CloudChurn*-5 utilizes end-to-end training instead of our proposed 3-phases training. In this case, we simply set the weights α=1 and β=1.

By doing so, we aim to understand the contributions and importance of each component in the overall performance of *CloudChurn*. Based on the results presented in

Table 4. Comparation of Performance in Ablation Study.

Method	CC	CC-1	CC-2	CC-3	CC-4	CC-5
Precision	**0.830**	0.650	0.804	0.259	0.806	0.810
F1-score	**0.845**	0.668	0.783	0.340	0.810	0.611
Accuracy	**0.933**	0.850	0.910	0.535	0.919	0.871
AUC	**0.909**	0.798	0.859	0.858	0.882	0.739

Table 4 (CC is short for CloudChurn), it is evident that *CloudChurn* outperforms other models in all evaluation metrics. Key observations are listed as follows:

1. Among the models that utilize only one data segment, namely *CloudChurn*-1, 2, and 3, *CloudChurn*-2 (order data only) achieves the best results. This suggests that user order data provides the most informative features for churner classification.
2. Although the performance of *CloudChurn*-3 (webpage browsing data only) is relatively poor, when combined with order data in *CloudChurn*-4, the webpage browsing data actively contributes to improved performance.
3. *CloudChurn*-5's performance is even worse than *CloudChurn*-1, considering that *CloudChurn*-5 takes all 3 parts of data as input while *CloudChurn*-1 only uses monitoring metric data. This indicates that our proposed 3-phases training outperforms traditional end-to-end training. By allocating suitable weights to sub-models, they make appropriate contributions to the integrated model.

4.5 Model Efficiency

To evaluate the efficiency of *CloudChurn*, we present the results based on the Churn-1 dataset, as shown in Table 5. The training time represents the duration required for a complete training process on the training set consisting of 1208 samples. Conversely,

Table 5. Comparation of Model Efficiency.

	CloudChurn	ICCP	LSTM	LR	XGBoost	RF	SVM	LDA
Training (s)	10.11	10.93	5.17	0.04	1.53	1.07	1.08	0.42
Inference (ms)	19.21	208.11	2.41	0.61	2.75	1.55	189.04	0.49

the inference time represents the time taken to perform predictions on the test set containing 134 samples. The experiments are conducted on a single NVIDIA Tesla-V100-SXM2-32GB GPU. While the integration and multi-phase training approach of *CloudChurn* incurs higher computational costs compared to some baseline models, it is important to emphasize that the overall computational requirements remain within an acceptable range, with inference times at the millisecond level.

5 Application in Practice

Table 6. Results of Online Deployment.

Period	June	July	August	September
# reported users	21	30	16	14
# churn users	16	25	12	11
Precision	0.762	0.833	0.750	0.786

Our proposed *CloudChurn* is deployed on the ModelArts platform in Huawei, where the model identifies churn users from the whole enterprise users in Huawei Cloud. Following the offline training of the model based on labeled historical data, it performs daily inferences using live data from enterprise users on the platform. Since it is hard for business personnel to identify and conduct to all users reported by the model, we add post-process module so that only the most urgent and essential churn users will be reported. The model has been deployed online for several months (the results identified by the business personnel between 2023.6 and 2023.9 are shown in Table 6), and the average precision is over 78% in total 81 churn warning records. In practice, the model does propose early churn warning for business personnel in Huawei Cloud with a relatively high precision, mitigating revenue losses that could reach a millionaire level.

6 Conclusion

In this paper, we present a comprehensive and effective enterprise customer churn prediction system in Huawei Cloud. It is based on a novel data-driven churn prediction model, *CloudChurn*, which employs specially designed structures tailored to handle specific feature subsets. Since its deployment on the ModelArts platform in June 2023, *CloudChurn* has significantly improved the accuracy of identifying churning enterprise customers, outperforming the previously deployed churn prediction algorithm by over 30% in F1-score. Over a four-month period (June to September 2023), the model achieved an average accuracy rate of 78.3%, contributing to revenue retention at the million-level. The effectiveness of *CloudChurn* is demonstrated through both offline and online evaluations, highlighting its potential as a valuable tool for cloud service providers in customer churn prediction. This capability empowers providers to enhance user retention and optimize their service offerings.

References

1. Ahmet Tuğrul Bayrak, Asmin Alev Aktaş, Orkun Susuz, and Okan Tunalı. Churn prediction with sequential data using long short term memory. In *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020.
2. Anol Bhattacherjee. Understanding information systems continuance: An expectation-confirmation model. *MIS quarterly*, pages 351–370, 2001.
3. Pei-Yu Chen and Lorin M Hitt. Measuring switching costs and the determinants of customer retention in internet-enabled businesses: A study of the online brokerage industry. *Information systems research*, 13(3):255–274, 2002.
4. Manirupa Das, Micha Elsner, Arnab Nandi, and Rajiv Ramnath. Topchurn: maximum entropy churn prediction using topic models over heterogeneous signals. In *Proceedings of the 24th International Conference on World Wide Web*, 2015.
5. Arno De Caigny, Kristof Coussement, and Koen W De Bock. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*, 269(2), 2018.
6. Mohammed Abdul Haque Farquad, Vadlamani Ravi, and S Bapi Raju. Churn prediction using comprehensible support vector machine: An analytical crm application. 2014.
7. Iqbal Hanif. Implementing extreme gradient boosting (xgboost) classifier to improve customer churn prediction, 2020.
8. Hemlata Jain, Ajay Khunteta, and Sumit Srivastava. Churn prediction in telecommunication using logistic regression and logit boost. *Procedia Computer Science*, 167:101–112, 2020.
9. Xi Liu, Muhe Xie, Xidao Wen, Rui Chen, Yong Ge, Nick Duffield, and Na Wang. A semi-supervised and inductive embedding model for churn prediction of large-scale mobile games. In *2018 ieee international conference on data mining (icdm)*, pages 277–286. IEEE, 2018.
10. Guangli Nie, Wei Rowe, Lingling Zhang, Yingjie Tian, and Yong Shi. Credit card churn forecasting by logistic regression and decision tree. *Expert Systems with Applications*, 2011.
11. Hiroaki Sakoe. Dynamic-programming approach to continuous speech recognition. In *1971 Proc. the International Congress of Acoustics, Budapest*, 1971.
12. Pan Tang. Telecom customer churn prediction model combining k-means and xgboost algorithm. In *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pages 1128–1131. IEEE, 2020.
13. Qi Tang, Guoen Xia, Xianquan Zhang, and Feng Long. A customer churn prediction model based on xgboost and mlp. In *2020 International Conference on Computer Engineering and Application (ICCEA)*, pages 608–612. IEEE, 2020.
14. Fan Wu, Ju Ren, Feng Lyu, Peng Yang, Yongmin Zhang, Deyu Zhang, and Yaoxue Zhang. Boosting internet card cellular business via user portraits: A case of churn prediction. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*.
15. Xiancheng Xiahou and Yoshio Harada. B2c e-commerce customer churn prediction based on k-means and svm. *Journal of Theoretical and Applied Electronic Commerce Research*.
16. Yaya Xie and Xiu Li. Churn prediction with linear discriminant boosting algorithm. In *2008 International conference on machine learning and cybernetics*, volume 1, 2008.
17. Yaya Xie, Xiu Li, EWT Ngai, and Weiyun Ying. Customer churn prediction using improved balanced random forests. *Expert Systems with Applications*, 36(3):5445–5449, 2009.
18. Carl Yang, Xiaolin Shi, Luo Jie, and Jiawei Han. I know you'll be back: Interpretable new user clustering and churn prediction on a mobile social application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.
19. George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference*.