Heliyon 7 (2021) e07761

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Identifying SME customers from click feedback on mobile banking apps: Supervised and semi-supervised approaches

Suchat Tungjitnob [a], Kitsuchart Pasupa [a,*], Boontawee Suntisrivaraporn [b,1]

[a] *Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand*
[b] *Data Analytics, Chief Data Office, Siam Commercial Bank, Bangkok 10900, Thailand*

ARTICLE INFO

Keywords:
Click feedback
Customer analysis
Supervised learning
Semi-supervised learning

ABSTRACT

Nowadays, the banking industry has moved from traditional branch services into mobile banking applications or apps. Using customer segmentation, banks can obtain more insights and better understand their customers' lifestyle and their behavior. In this work, we described a method to classify mobile app user click behavior into two groups, *i.e.* SME and Non-SME users. This task enabled the bank to identify anonymous users and offer them the right services and products. We extracted hand-crafted features from click log data and evaluated them with the Extreme Gradient Boosting algorithm (XGBoost). We also converted these logs into images, which captured temporal information. These image representations reduced the need for feature engineering, were easier to visualize and trained with a Convolutional Neural Network (CNN). We used ResNet-18 with the image dataset and achieved 71.69% accuracy on average, which outperformed XGBoost, which only achieved 61.70% accuracy. We also evaluated a semi-supervised learning model with our converted image data. Our semi-supervised method achieved 73.12% accuracy, using just half of the labeled images, combined with unlabeled images. Our method showed that these converted images were able to train with a semi-supervised algorithm that performed better than CNN with fewer labeled images. Our work also led to a better understanding of mobile banking user behavior and a novel way of developing a customer segmentation classifier.

1. Introduction

Modern mobile technology has developed rapidly, making smartphones an essential item in people's daily lives. Using smartphones, the banking industry took this opportunity to make its services easier to access with mobile banking applications or apps. Bank customers can immediately make banking transactions, anywhere and anytime with this app. These advantages deliver the customer a different method to make financial and banking transactions. Additionally, the mobile banking app reduces the cost of increasing extra branches. The bank also earns the benefit of obtaining customers' data from the app. Making the banking industry rapidly shifted from traditional banking services to digital platforms.

Customer segmentation splits customers into separate groups, based on their personality and behavior. This helps banks to market and sell their product to the right customer. Data science also enabled us to conduct a method to reveal hidden insights from mobile banking customer data. Allowing a more efficient system to segment a customer. A mobile banking app allows the bank to receive useful information, for example, user feedback, which leads to a better perception of customer behavior. An easy way to collect data from mobile banking apps is user click feedback. Thailand ranked first for mobile banking according to the Digital 2021 report [1]. Referring to statistics from the Bank of Thailand, the number of agreements that customers have been applied for the mobile banking service is 68.4 million agreements in December 2020 with an average of 770 million transactions per month in 2020 [2]. Mobile banking apps are now the most commonly used banking platforms in Thailand. It provides most of the core banking functionality, such as fund transfer, personal loan, QR payment, card-less-ATM (withdraw money from ATM without the need of ATM card). The app provides a large volume of customer data which leads the bank to gain more insight and possibilities to recognize the customer preferences, lifestyle, and financial behavior.

Banks aim to provide the right products for each specific group of customers. One of those groups is a small and medium-sized enterprise (SME), which are registered businesses that maintain revenues, assets

* Corresponding author.
E-mail address: kitsuchart@it.kmitl.ac.th (K. Pasupa).
[1] The author was affiliated with Siam Commercial Bank until April 2020.

https://doi.org/10.1016/j.heliyon.2021.e07761
Received 8 April 2021; Received in revised form 28 July 2021; Accepted 9 August 2021

or have several employees below a certain threshold. The bank desires to reach out to its SME customers to promote and encourage SME customers to use its SME product. However, many SME customers do not register as companies. They are only natural persons or stores, who regularly generate incomes—selling their products or services online and through physical stores. These customers rarely reveal their status as SMEs, so the bank cannot easily recognize their identities unless the customer notifies them. Therefore, the banks want to identify this group of customers and offer them the right products.

In this work, we showed how to classify mobile banking app users into two groups, *i.e.* SME and Non-SME, based on their behavior on the app. We examined click feedback data collected from a mobile banking app to distinguish between SME and Non-SME users, using their click behavior. The data contained a user click log of a group of anonymous users. Unfortunately, we only had a limited number of users, labeled as SME or Non-SME, but a large number of unlabeled users. There was a possibility that many SME users were among the unlabeled users. We can consider this problem as a supervised learning problem, that learns from labeled users and predicts unlabeled users. The model allows the bank to offer the right products or services to the right customers.

However, labeled users are generally more expensive and more difficult to obtain than unlabeled users, and we would prefer to have more labeled users to create a good prediction model. Since we have a large number of unlabeled users, we used this data to its full potential. We expected that we could build a better model, than using supervised learning alone, by adding cheap and abundant unlabeled data. Therefore, we tackled this problem as a semi-supervised learning problem.

The contributions of our work are:

1. We crafted our click log data into a set of hand-crafted features. These features trained an Extreme Gradient Boosting (XGBoost) model to distinguish between SME and Non-SME users.
2. Click log data was converted into visual representations that captured temporal information. We used these images to train a Convolutional Neural Network (CNN) model to classify SME or Non-SME users and that performed better than handcrafted features.
3. We used consistency training to train a semi-supervised learning model with unlabeled data. This model combined labeled with unlabeled data and allowed us to train a model, that performed better than CNN, while using less labeled data.

2. Related work

In the past, a customer segmentation task was handled by traditional statistical methods [3, 4]. However, due to the advancement of machine learning and big data technology, numerous machine learning techniques, including classification, *i.e.* decision tree [5, 6], neural networks [7, 8] and clustering, *i.e.* k-means clustering [9], DBSCAN [10], was used to segment customers in various industries. Many reports used machine learning techniques to segment customers in the banking industry as well. Mihova et al. used customer credit history with k-means clustering to determine customer groups [9]. Ogwueleka et al. classified new bank customers based on previously observed customer behavior with a neural network [8]. Zakrzewska et al. described the shortcomings and advantages of k-mean clustering, DBSCAN and two-phase clustering algorithm, applied to noisy high dimensional bank customer data [10]. Many works also showed that temporal data, such as website click feedback, assisted customer segmentation. Bock et al. trained a random forest classifier, with website visitor click feedback patterns [11]. The classifier was able to form a demographic profile of a website visitor, *i.e.* gender, age and occupation, to support an advertising team. Su et al. used an e-commerce website click data to form an insight into user click behavior. They clustered customers into a group of interest patterns, by using user browsing behavior data, with a lead clustering algorithm [12]. These works used machine learning algorithms to tackle customer segmentation tasks. However, there were some problems, when considering customer segmentation as classification or clustering tasks. A model aims to learn customer type from labeled data in a classification task. However, labeling data is a crucial and expensive task. The amount of data created is growing faster ever than before—accelerated by emerging technologies. Thus, it is difficult to obtain labels for all samples. On the other hand, clustering tasks do not require labels and do not directly specify the desired type of customer, but it groups the sample to discover the inherent structure of unlabeled data.

Among many machine learning techniques, deep learning is widely used, due to its ability to automatically extract features from raw data and its promising performance. It has been applied to predict user click behavior as well. Du et al. used a sequential model, Long short-term memory (LSTM) with click log data to tackle an online recommendation problem [13]. Gharibshah et al. also used LSTM to create a deep learning framework, that could predict users' interest and display recommended advertisements, when they are browsing the website [14]. In addition, deep learning techniques have been used with various types of data other than clicking log data—e.g., face [15, 16], eye gaze [17], gesture [18], text [19, 20, 21, 22]—to assist in understanding customers and improving customer satisfaction. However, a deep learning technique also requires a large labeled dataset to train a generalized model, but, again, collecting a large labeled dataset is expensive and laborious. A conventional deep learning model cannot reach its full potential training on a small labeled dataset, such model easily overfit [23, 24]. It requires some other techniques, *e.g.* augmentation, dropout to prevent the problem.

In addition to these techniques, semi-supervised learning can solve the problem as well. This approach uses both labeled and unlabeled data to train a model. There are many approaches to semi-supervised learning together with deep learning, whether a graph-based method [25, 26] or generative methods [27, 28], which achieved state-of-the-art performance. Recently, the consistency training method has been often used and outperformed many existing state-of-the-art semi-supervised methods. Consistency training is a semi-supervised learning technique, that aims to tackle the lack of labeled data problem in the CNN model. It regularizes model predictions, assuming that the model should give the same prediction no matter whether the original input or its variants are fed in. Examples of the variants are noise addition, perturbation, augmentation. Many researchers used this framework and achieved better performance than the state-of-the-art for semi-supervised learning in image classification. Most used noise, for example, Gaussian noise or adversarial noise injection to the image [29, 30, 31]. Recently, all of these methods were surpassed by Xie et al. [32], who introduced a novel approach to perturb the image with data augmentation—their "Unsupervised Data Augmentation for Consistency Training"—a deep semi-supervised learning algorithm based on the consistency training assumption, that used both labeled and unlabeled data in the training phase. The model could learn differences between original unlabeled data and the augmented unlabeled data. Later, Sohn et al. [33] combined the consistency training technique and pseudo-labeling technique [34, 35] to create a new semi-supervised learning framework—"FixMatch", which learned from both labeled data with supervised learning and unlabeled data with semi-supervised learning, assuming consistency training, along with the pseudo label created in the supervised learning part.

To the best of our knowledge, existing banking industry papers have not considered temporal information in their models. Although some previous works considered temporal information, they were only based on click-through data from e-commerce websites. Some works successfully used deep learning techniques, *e.g.* LSTM, with click log data from a website, as sequential input [13, 14]. In this work, we considered temporal information of user clicks on a mobile banking app to classify users into SME or Non-SME groups based on the click log. To be useful, results must be interpretable. Our click log mentioned many banking features. Most banks aim to visualize user behavior in time. However, simply using click logs as sequences makes it hard for the business to visualize and understand behavior. Therefore, we tackled the problem from another point of view by representing the click logs as a set of images, allowing us to interpret the results to the business that used this information. Lastly, this information can assist the management team to plan marketing or advertising strategies from the pattern that lies in it.

3. Methods

3.1. Data collection

The raw click log data, that consisted of a row of click event types and a timestamp, were collected from a mobile app database and from 25,868 anonymous users, including 12,621 SME users and 13,247 Non-SME users [36]. The users were initially classified as SME or Non-SME, when the customer opened the bank account. Initially, there were 535 unique events, which can be divided into interface events and banking function events. We wanted to select only events, that were essential to our task. We created a matrix from our data by counting the number of total usage times for each event of each user, leading to a $25,868 \times 535$ matrix. The matrix represented the click event frequency in one month for each user. A random forest algorithm used this matrix to calculate an importance score for each event. Then we ranked event importance scores in descending order and used an elbow method to determine the number of events that should be kept, based on its importance score. From the elbow method, 56 events remained—21 banking function events and 35 interface events. Since we were interested in banking functions, we kept only the 21 banking events and eliminated interface events. Then we grouped the remaining banking events into seven groups of events—the ‘primary events’. Any event, which contained a keyword from one of the seven primary events, was labeled a sub-event, *e.g.* transfer was a primary-event and transfer_slip was its sub-event. We also included additional sub-events, eliminated by the elbow method, to be part of a primary event. Lastly, our click log data were separated into seven primary events and 57 sub-events, see Fig. 1.

3.2. Hand-crafted feature extraction

We extracted a feature from a click log data to train a machine learning model to classify SME or Non-SME users. We show three groups of hand-crafted feature methods in Table 1.

1. Number of clicks for each primary event $\{F_1, F_2, \dots, F_7\} \in \mathcal{F}_f$: We counted a frequency for each primary event. We assumed that $\mathcal{F}_f$ should have a unique attribute, which can distinguish between SME and Non-SME users.
2. Number of clicks for each primary event within a period $\{F_8, F_9, \dots, F_{28}\} \in \mathcal{F}_t$: The feature was crafted to include a period-of-time taken from the timestamp in our click log data. Each primary event frequency was counted, for three periods of the working day (8-hour shift) in Thailand: 08:00 to 17:00, 17:00 to 24:00 and 24:00 to 08:00. We assumed that mobile banking app use would be related to the user working hour periods, *i.e.* the SME users would mainly use the app between 08:00 to 17:00 on weekdays.
3. Combination of $\mathcal{F}_f$ and $\mathcal{F}_t$ ($\mathcal{F}_c$): We concatenated $\mathcal{F}_f$ and $\mathcal{F}_t$, assuming that a model would perform better, when trained with more features.

3.3. Visualization of the log

When we extracted the click log data with the hand-crafted method, we found that it was difficult to extract the features that included temporal information for the click log data. The features were based on our assumption on the use of the app related to time intervals. We decided to convert the click logs into images, which could capture the timestamp attribute. The image also enabled us to visualize and identify the trends and pattern behind these click log data.

Table 1. Features extracted from click log data. $\mathcal{F}_f$ denotes feature crafted using a number of clicks for each primary event. $\mathcal{F}_t$ denotes feature crafted using a number of clicks for each primary event within a period of time.

Type	Index	Name
$\mathcal{F}_f$	1	billpay
	2	cardless_atm
	3	export_stmt
	4	loan
	5	moneymovement
	6	topup
	7	transfer
$\mathcal{F}_t$	8	billpay 08:00 to 17:00
	9	billpay 17:00 to 24:00
	10	billpay 24:00 to 08:00
	11	cardless_atm 08:00 to 17:00
	12	cardless_atm 17:00 to 24:00
	13	cardless_atm 24:00 to 08:00
	14	export_stmt 08:00 to 17:00
	15	export_stmt 17:00 to 24:00
	16	export_stmt 24:00 to 08:00
	17	loan 08:00 to 17:00
	18	loan 17:00 to 24:00
	19	loan 24:00 to 08:00
	20	moneymovement 08:00 to 17:00
	21	moneymovement 17:00 to 24:00
	22	moneymovement 24:00 to 08:00
	23	topup 08:00 to 17:00
	24	topup 17:00 to 24:00
	25	topup 24:00 to 08:00
	26	transfer 08:00 to 17:00
	27	transfer 17:00 to 24:00
	28	transfer 24:00 to 08:00

We created visualizations or ‘images’, using each primary event with 24×60 pixels (based on 24 hours per day and 60 minutes per hour). The 60 pixels (1 min per pixel) were too sparse, so we rescaled it to 30 (two min per pixel) leading to 24×30 pixel images. The steps for converting click log data into images are shown in Fig. 2.

The steps in Fig. 2 led to seven 24×30 images, that represented click behavior for each primary event of a single user. We stacked each image into one image leading to an image with seven channels—($24 \times 30 \times 7$) and label them, $\mathcal{I}_{7c}$. Examples of $\mathcal{I}_{7c}$ are in Fig. 3. We can observe some different use pattern in each primary event for each user. According to Fig. 3, this SME user had never used a cardless ATM function, so the cardless ATM image is all black. However, the Non-SME user use this event frequently, so the cardless ATM image from these users had more white pixels. This image showed that this user frequently withdrew money with the cardless ATM function.

However, most computer monitors output a three-channel (RGB) image, for $\mathcal{I}_{7c}$, which has seven channels, we need to extract each channel and code it in a grayscale (one channel) image. Thus, $\mathcal{I}_{7c}$ was still difficult for human eyes to visualize. Therefore, we devised a method that converted $\mathcal{I}_{7c}$ into RGB images, which made it easier to visualize.

We selected the top three frequency channels (‘transfer’ (11,598,530 clicks), ‘moneymovement’ (2,610,249 clicks) and ‘billpay’ (1,203,870 clicks)) from $\mathcal{I}_{7c}$ to be represented in three channels of an RGB image, see Fig. 1. ‘transfer’ is an event when a user transfers money to other bank accounts, ‘moneymovement’ is an event when a user checks the summary of money movements—withdraw from or deposit to—their account, and ‘billpay’ is an event when a user pays the bill with their account. This led to RGB images ($24 \times 30 \times 3$), defined as $\mathcal{I}_{rgb}$. Selection steps for the top three frequency channels, that convert $\mathcal{I}_{7c}$ into $\mathcal{I}_{rgb}$, are shown in Fig. 4 and converted images are shown in Fig. 5. Instead of

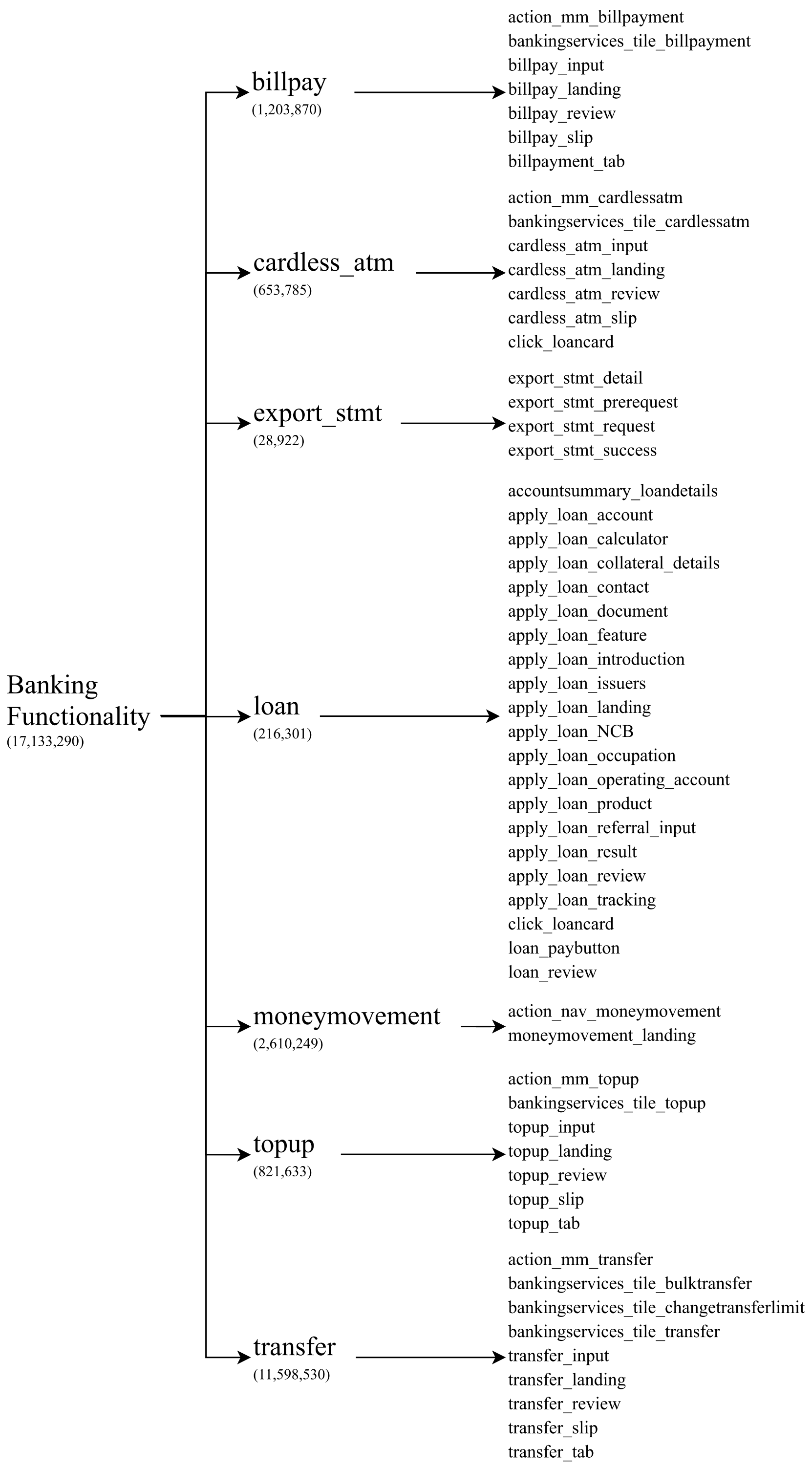

Fig. 1. Banking function click events: There were seven primary events and 57 sub-events considered. Numbers in parentheses are click frequency.

forcing us to distinguish seven gray levels from $\mathcal{I}_{7c}$, $\mathcal{I}_{rgb}$ enabled us to easily visualize a behavior from one RGB image: from Fig. 5, we can see that an SME user used the app frequently and consecutively. On the other hand, a Non-SME user was less active on the app, showing a disjointed use pattern.

The images were found to be suitable for visualizing underlying use patterns present in the click logs. They also enabled a CNN to distinguish between SME and Non-SME users based on images, representing their use behavior.

3.4. Models

3.4.1. XGBoost

XGBoost is a tree-based gradient boosting algorithm [37]. It creates an ensemble of weak prediction models and combines them into a strong prediction model. The strong prediction model is trained sequentially by calculating its error through a loss function and using this information to help the new ensemble model correct its predecessor. It is an optimized version of tree-based gradient boosting [37], designed

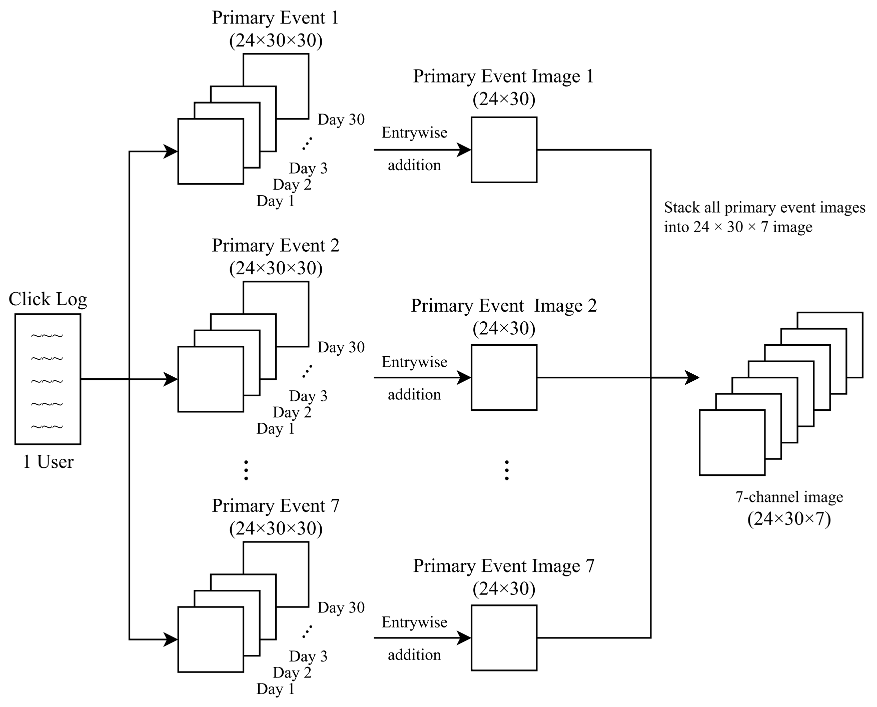

Fig. 2. Steps for encoding user click log—from seven primary events into a seven channel image.

to be efficient, flexible and better regularized to control over-fitting. XGBoost trained faster and achieved better performance than other gradient boosting implementations [38]. XGBoost was used to evaluate the performance of $\mathcal{F}_f$, $\mathcal{F}_t$ and $\mathcal{F}_c$.

3.4.2. ResNet

ResNet is a deep CNN architecture designed to eliminate the problem of vanishing gradient. He et al. introduced a concept of 'skip connection', by adding an input value to the output after the convolution block [39]. This enabled a network to stack more and deeper layers, without affecting training performance. We used an 18-layer version of ResNet, called ResNet-18. He et al.'s default settings [39] were used in an experiment on $\mathcal{I}_{rgb}$. For $\mathcal{I}_{7c}$, we modified ResNet-18 input layer settings to input a seven-channel image, instead of the original three-channel one.

3.4.3. FixMatch

FixMatch is a recent semi-supervised consistency training framework [33]; it combined both consistency training and pseudo labeling techniques as shown in Fig. 6.

FixMatch first trains a model with a labeled image. After that, it augments the unlabeled image, using weak augmentation and strong augmentation. The class of the weak image will be predicted based on the trained model. If the confidence of the prediction exceeds the threshold, the algorithm will convert the prediction into a one-hot encoding that acts as the pseudo-label. Now we have a pseudo-label of the weakly augmented unlabeled data. Then the algorithm predicts their counterparts that are augmented with the strong augmentation. Finally, the algorithm calculates the loss from these two predictions using cross-entropy loss. The model trained by this method aims to make its prediction between weakly and strongly augmented data close to each other. The algorithm minimizes the final loss function which is the combination of supervised loss and unsupervised loss.

The supervised loss $\mathcal{L}_S$ is calculated using cross-entropy loss of the weakly augmented labeled data:

$$\mathcal{L}_S = \frac{1}{B_S}\sum_{i=1}^{B_S} H\left(p_i, p_m(y|\alpha(x_i))\right), \tag{1}$$

where B_S denotes the size of the batch of labeled samples, $\alpha(\cdot)$ denotes weak augmentations, H is the cross-entropy loss, p_i is a one-hot label, and $p_m(y|\alpha(x_i))$ is a model's predicted class distribution from weak augmentation. This step calculates the loss in a standard supervised manner.

For the unsupervised loss, the algorithm uses the unlabeled data to calculate $\mathcal{L}_U$,

$$\mathcal{L}_U = \frac{1}{B_U}\sum_{i=1}^{B_U} \mathbb{1}(\max(q_i) \geq \tau) H(\hat{q}_i, p_m(y\,|\,\mathcal{A}(x_i))), \tag{2}$$

where B_U denotes the size of the batch of unlabeled samples, $\mathcal{A}(\cdot)$ denotes strong augmentations, q_i is $p_m(y|\alpha(x_i))$, a pseudo-label can be defined by $\hat{q}_i = \arg\max(q_i)$ and τ is a threshold that retains the pseudo-label. If a sample prediction confidence is lower than τ, it will not be used to calculate the unsupervised loss in this epoch. The algorithm weakly augments the unlabeled data and predicts these data with the model from the supervised training. Then it converts the prediction into a pseudo label. The pseudo label is used to calculate cross-entropy loss with the strongly augment unlabeled data.

Lastly, FixMatch minimizes the final loss function:

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_U, \tag{3}$$

where λ denotes a parameter that sets the balance between the supervised loss and the unsupervised loss. Increasing λ will increase the effect of unsupervised loss contributing to the overall loss value [32]. The more we increase this value, the more unsupervised loss from unlabeled data will affect the overall loss.

4. Experiment setting

Before starting the experiment, we examined our dataset and found that a large number of the app users were not active users. These users rarely used the app and tended to provide noisy data. Their click log behaviors were random and difficult to distinguish and can lead a model in the wrong direction. Therefore, we plotted user click frequency in Fig. 7: this distribution was significantly right-skewed, with more than 4000 infrequent users.

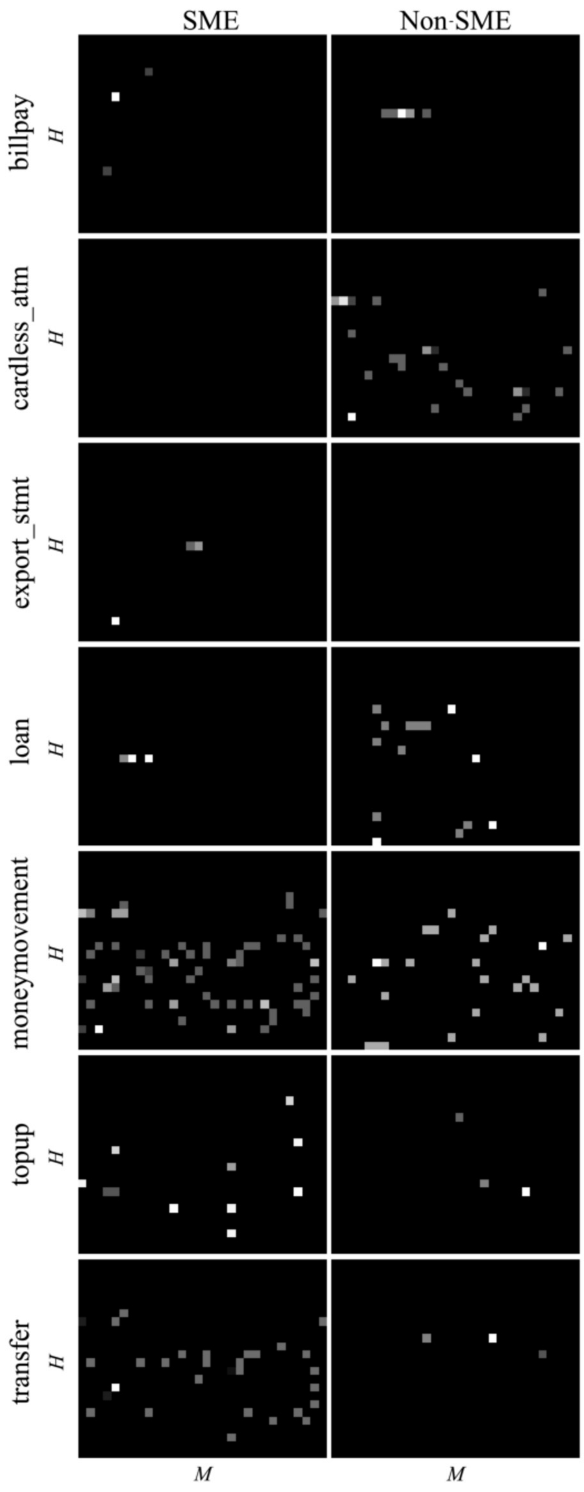

Fig. 3. Example of $\mathfrak{I}_{7c}$: each row represents a primary event. Each column represents $\mathfrak{I}_{7c}$ for one user example. H denotes a 24-hour (one hour per pixel). M denotes a 30-interval (two minutes per pixel).

We assumed that active users were familiar with the app. We wanted to evaluate the model using the active user data, which can show a more meaningful behavior. However, the question still arose that what threshold level should we define to discard non-active users. We decided to sort the user on their usage click frequency and discarded users with lower than 50%, 55%, 60%, 65%, 70% and 75% percentiles. The number of remaining users, after discarding non-active users in each threshold, are shown in Table 2.

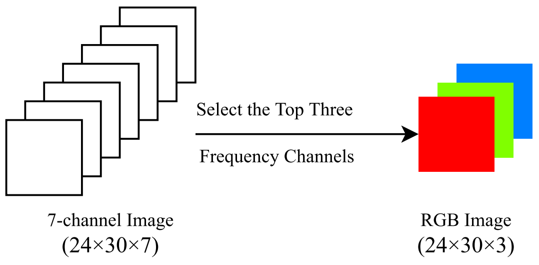

Fig. 4. By selecting only the top three frequency channels, we convert a grayscale seven-channel image ($\mathfrak{I}_{7c}$) into an RGB image ($\mathfrak{I}_{rgb}$).

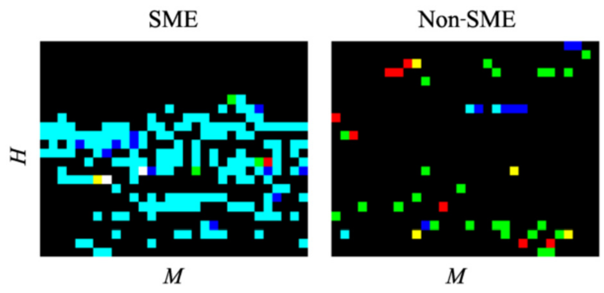

Fig. 5. Examples of $\mathfrak{I}_{rgb}$. Each image represents $\mathfrak{I}_{rgb}$ of SME and Non-SME examples. H denotes hour—24 hours in a day (one hour per pixel). M denotes minute—60 minutes in an hour (here, two minutes per pixel). 'billpay', 'moneymovement' and 'transfer' is represented in red, green, blue channels, respectively. Magenta—'billpay' with 'moneymovement', yellow—'billpay' with 'transfer', cyan—'moneymovement' with 'transfer', and all events together—white. These images represented two types of behavior (SME and Non-SME) when they were using the mobile banking app.

Table 2. Number of remaining users after discarding at each threshold.

Threshold	SME	Non-SME	Total
50	6,303	6,619	12,932
55	5,674	5,949	11,638
60	5,043	5,299	10,341
65	4,414	4,636	9,049
70	3,784	3,972	7,751
75	3,155	3,308	6,454
Total	12,621	13,247	25,868

Here, we performed two tasks—supervised learning and semi-supervised learning tasks.

1. Supervised learning task: We evaluated the SME classification task on five sets of inputs. Hand-crafted features $\mathcal{F}_f$, $\mathcal{F}_t$ and $\mathcal{F}_c$ were trained with the XGBoost. Then we converted the click log data into an image and evaluated the performance of image data ($\mathfrak{I}_{7c}$ and $\mathfrak{I}_{rgb}$). These five sets of input were run with six different discarded thresholds.
2. Semi-supervised learning task: We only evaluated a FixMatch model with a 75 percentile threshold $\mathfrak{I}_{rgb}$ dataset, because it yielded the best performance in the three-channel image in the supervised learning task. We simulated a scenario where the labeled data were limited. The training sets were randomly selected as a set of 50, 125, 250, 500, 1000 and 1500 labeled images per class, as shown in Fig. 8. We used these randomly selected images as labeled images and the remaining images as unlabeled images. Both labeled and unlabeled images in the training sets were used to train the FixMatch models while only labeled images were used to train the CNN models. Note that λ was set to 1 because we wanted both labeled and unlabeled data to equally contribute to overall loss.

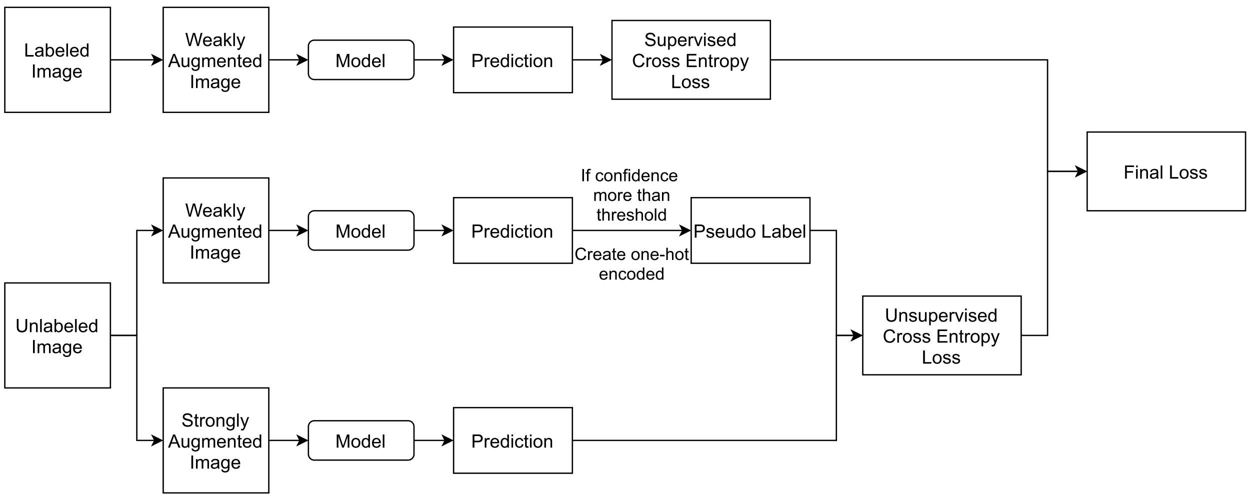

Fig. 6. Flowchart of FixMatch algorithm.

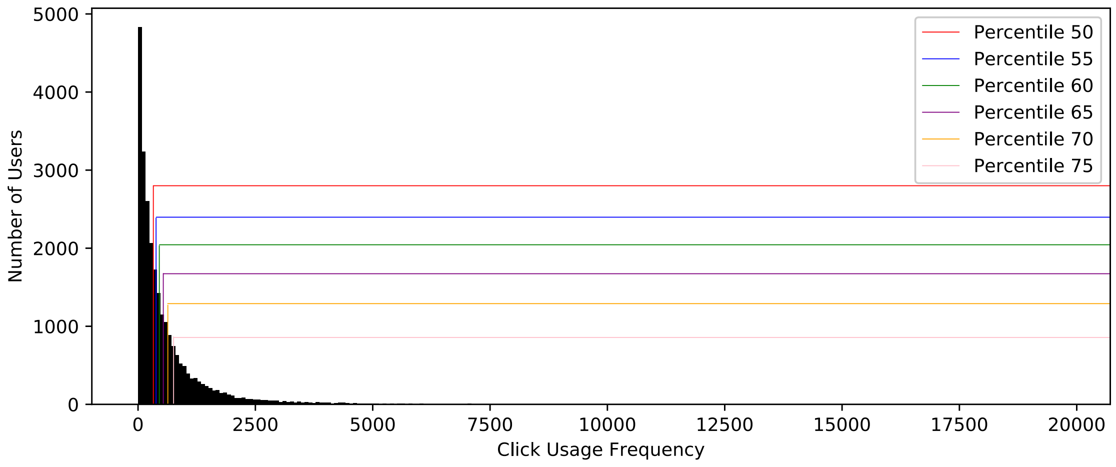

Fig. 7. Histogram showing distribution of user click frequency and the threshold for each discarded percentile considered in this work.

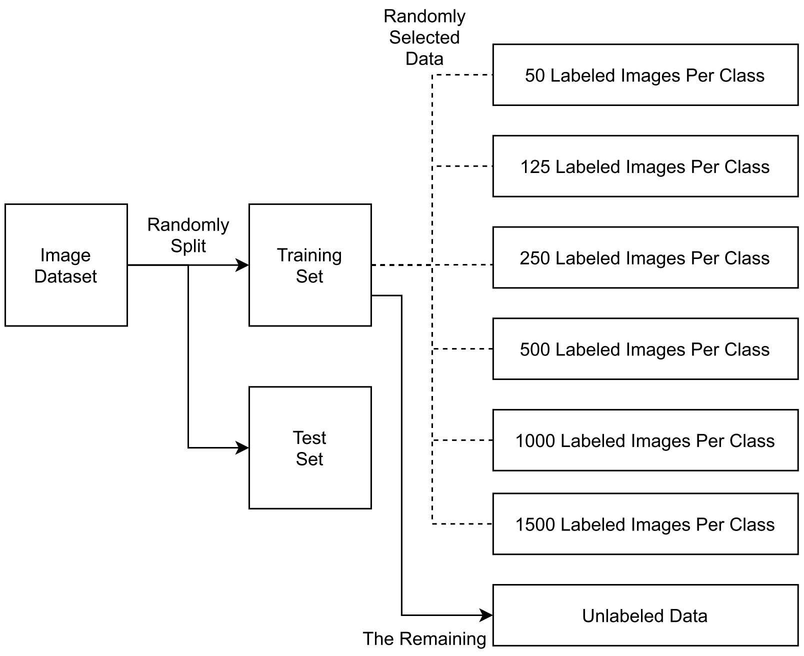
Image Dataset
Randomly Split
Training Set
Test Set
Randomly Selected Data
50 Labeled Images Per Class
125 Labeled Images Per Class
250 Labeled Images Per Class
500 Labeled Images Per Class
1000 Labeled Images Per Class
1500 Labeled Images Per Class
Unlabeled Data
The Remaining

Fig. 8. Data splitting used to prepare for a semi-supervised learning experiment.

Table 3. SME classification accuracy (as %) with XGBoost ($\mathcal{F}_f$, $\mathcal{F}_t$ and $\mathcal{F}_c$) and CNN ($\mathcal{I}_{7c}$ and $\mathcal{I}_{rgb}$) on each discarded threshold.

Discarded threshold	Features				
	$\mathcal{F}_f$	$\mathcal{F}_t$	$\mathcal{F}_c$	$\mathcal{I}_{7c}$	$\mathcal{I}_{rgb}$
50	61.10±0.79	61.10±0.77	61.31±0.89	69.43±0.60	68.81±0.99
55	60.92±0.95	61.10±0.93	61.49±0.56	70.89±0.55	68.59±2.17
60	**62.29±0.80**	**62.34±0.63**	**62.37±1.08**	71.96±1.61	71.50±1.39
65	61.29±0.80	62.32±0.91	62.31±0.53	72.66±0.93	72.54±2.15
70	60.99±0.31	61.69±0.49	62.00±0.58	73.41±0.71	72.70±1.28
75	61.48±1.07	62.19±1.19	62.26±1.39	**73.46±2.43**	**74.31±1.35**
Average	61.35±0.79	61.79±0.82	61.95±0.84	71.97±1.14	71.41±1.56

Note that we randomly split every input ($\mathcal{F}_f$, $\mathcal{F}_t$, $\mathcal{F}_c$, $\mathcal{I}_{7c}$ and $\mathcal{I}_{rgb}$) into training and test sets with an 80:20 ratio. Then 20% of the training set was extracted as a validation set. We selected models with the lowest validation loss and tested them on an unseen test set.

5. Results & discussion

In this section, we evaluated the accuracy of our models based on the two tasks—supervised learning and semi-supervised learning tasks, followed by error analysis. Our goal was to compare the traditional supervised method with the semi-supervised method. We selected the state-of-the-art for each technique. FixMatch represented semi-supervised learning, whereas XGBoost and ResNet represented supervised learning methods.

5.1. Supervised learning task

From Table 3, hand-crafted features achieved the best performance at 61.95% average accuracy with $\mathcal{F}_c$, followed by $\mathcal{F}_t$ at 61.79% and $\mathcal{F}_f$ at 61.35%. Considering each discarded threshold, we found that $\mathcal{F}_c$ achieved higher accuracy than $\mathcal{F}_t$ and $\mathcal{F}_f$. This showed that with the $\mathcal{F}_t$ input, where we added click frequency and times, delivered a more accurate model than those that did not include times.

The results from ResNet-18, trained with the image dataset and six different thresholds, shown in Table 3, $\mathcal{I}_{7c}$ delivered the best performance at 71.97% average accuracy, followed by $\mathcal{I}_{rgb}$ at 71.41%. Thus $\mathcal{I}_{7c}$ outperformed $\mathcal{I}_{rgb}$ at every threshold, except the 75 percentile one. The models trained with $\mathcal{I}_{7c}$ achieved better performance, as they used more information, learning from the seven channels versus only three channels in $\mathcal{I}_{rgb}$ on average. However, $\mathcal{I}_{rgb}$ still achieved the same level of performance differing by only $\sim 0.56\%$ in average accuracy. $\mathcal{I}_{rgb}$ was also more suitable, when considering visualization and error analysis.

Also, the model gained better performance at each threshold after discarding more inactive users. We concluded that inactive users affected the model performance and need to be filtered out.

5.2. Semi-supervised learning task

Here, we compared the performance of CNN and FixMatch trained with $\mathcal{I}_{rgb}$ at 75 percentile threshold. FixMatch models were trained with both labeled and unlabeled images, whereas CNN could only train with labeled images on six training scenarios. Table 4 shows that FixMatch performed better than CNN($\mathcal{I}_{rgb}$) in every training scenario. For example, with only 250 labeled images per class, FixMatch achieved better than 70% accuracy, whereas CNN needed to train with more than 1500 labeled images per class to deliver the same performance. Also, FixMatch trained with 1500 labeled image per class was competitive with CNN with the full dataset on $\mathcal{I}_{7c}$, differing only by $\sim 0.34\%$ accuracy.

5.3. Error analysis

We analyzed the errors for $\mathcal{I}_{rgb}$ and found a hidden insight in $\mathcal{I}_{rgb}$ samples. We focused on samples with high probabilities for either SME or Non-SME - see Fig. 9. According to the figure, users predicted as SMEs had a very similar use behavior: they were extremely active and their time spanned 8:00 to 24:00. They mostly used either 'money-movement' and 'transfer' events together (cyan) or just a consecutive 'transfer' event throughout the day (blue). This showed that SME users regularly checked their transaction movements— whether their customers completed payments and transferred money to other accounts or just transferred money to run their business. They also normally used the 'billpay' event (red)—this applied to SME users that were predicted as Non-SME users. In addition, a group of users that was flagged as Non-SME but was predicted as SME could be users that were not SME at first, but later they turned into one, without notifying the bank. Through our information, a bank can offer this user group services and products for SMEs, *i.e.* business advisor and SME-loans, which could help them grow their business faster.

Table 4. SME classification accuracy on FixMatch and CNN in each training scenario with I_{rgb} (as %).

Labeled per classes	CNN	FixMatch
50	51.90%	67.39%
125	54.68%	69.87%
250	57.01%	70.72%
500	65.89%	71.11%
1000	69.67%	72.19%
1500	72.15%	73.12%
Full Dataset	74.31%	-

Furthermore, users that were predicted as Non-SME also had a similar behavior pattern, *i.e.* irregular use throughout the day and not active continuously. They only used it when it was necessary. In addition, a group of SME users, that was predicted as Non-SME could be SME users, that do not use a mobile banking app or use other banks as their primary bank. From this information, the bank could build a marketing plan or offer promotions for this user group, which can encourage them to become regular mobile banking app users.

6. Conclusion

Mobile banking apps contain a mine of useful data. One of them is the click log data, which can lead to many opportunities for the bank to gain more insight and knowledge of their customer. By converting these logs into images, that capture temporal information, reduced the need for hand-crafted features and made it easier for visualization. In this work, we used five sets of inputs, consisting of hand-crafted features ($\mathcal{F}_f$,$\mathcal{F}_t$ and $\mathcal{F}_c$) and images ($\mathcal{I}_{7c}$ and $\mathcal{I}_{rgb}$) to classify SMEs. We found that using the images outperformed hand-crafted features, due to the time-related features, which they captured. $\mathcal{I}_{rgb}$ achieved the best performance and was easier for visualization, which was helped the management team identify trends behind user click behavior. We also showed that, using a semi-supervised learning technique, FixMatch, we were able to train a model using a smaller amount of labeled data than the traditional CNN method. As a consequence, it reduced the need for human labor to annotate this data.

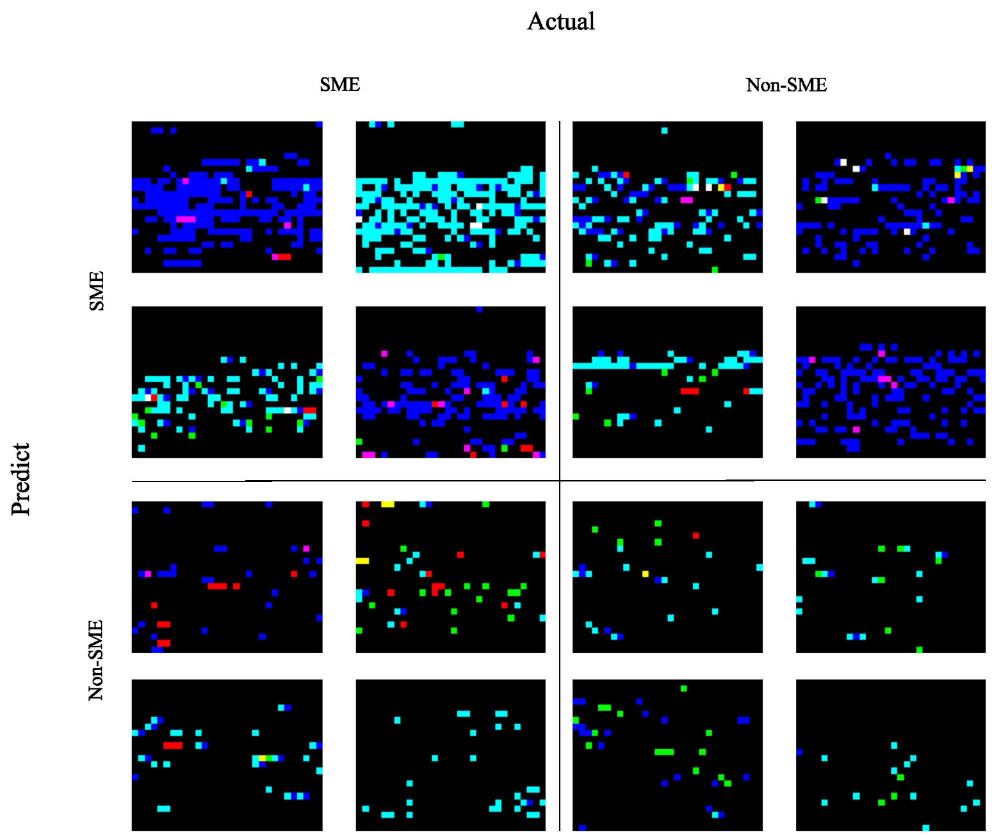

Fig. 9. User samples with high probability prediction from error analysis. 'billpay', 'moneymovement' and 'transfer' is represented in red, green, blue channels, respectively. Magenta—'billpay' with 'moneymovement', yellow—'billpay' with 'transfer', cyan—'moneymovement' with 'transfer', and all events together—white.

Declarations

Author contribution statement

S. Tungjitnob: Performed the experiments; Analyzed and interpreted the data; Wrote the paper.

K. Pasupa: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Wrote the paper.

B. Suntisrivaraporn: Contributed reagents, materials, analysis tools or data.

Funding statement

This work was supported by the Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang and Siam Commercial Bank under a Memorandum of Understanding, since 2019.

Data availability statement

The authors do not have permission to share data.

Declaration of interests statement

The authors declare no conflict of interest.

Additional information

No additional information is available for this paper.

References

[1] S. Kemp, Digital 2021: Global Overview Report, Technical Report, Datareportal, 2021, https://datareportal.com/reports/digital-2021-global-overview-report.
[2] Bank of Thailand, Use of mobile banking and Internet banking (PS_PT_009_S2), https://www.bot.or.th/English/Statistics/PaymentSystems/Pages/StatPaymentTransactions.aspx, 2021.
[3] S. Alavi, V. Ahuja, An empirical segmentation of users of mobile banking apps, J. Internet Commer. 15 (2016) 390–407.
[4] F. Hamka, H. Bouwman, M. de Reuver, M. Kroesen, Mobile customer segmentation based on smartphone measurement, Telemat. Inform. 31 (2014) 220–227.
[5] S.H. Han, S.X. Lu, S.C. Leung, Segmentation of telecom customers based on customer value by decision tree model, Expert Syst. Appl. 39 (2012) 3964–3973.
[6] C. Dullaghan, E. Rozaki, Integration of machine learning techniques to evaluate dynamic customer segmentation analysis for mobile customers, Int. J. Data Min. Knowl. Manag. Process 7 (2017) 13–24.
[7] J. Lee, S.-C. Park, Intelligent profitable customers segmentation system based on business intelligence tools, Expert Syst. Appl. 29 (2005) 145–152.
[8] F.N. Ogwueleka, S. Misra, R. Colomo-Palacios, L. Fernandez, Neural network and classification approach in identifying customer behavior in the banking sector: a case study of an international bank, Human Factors and Ergonomics in Manufacturing & Service Industries 25 (2015) 28–42.
[9] V. Mihova, V. Pavlov, A customer segmentation approach in commercial banks, in: AIP Conference Proceedings, vol. 2025, 2018, p. 030003.
[10] D. Zakrzewska, J. Murlewski, Clustering algorithms for bank customer segmentation, in: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications (ISDA'05), 8–10 September 2005, IEEE, Warsaw, Poland, 2005, pp. 197–202.
[11] K.W.D. Bock, D.V. den Poel, Predicting website audience demographics for web advertising targeting using multi-website clickstream data, Fundam. Inform. 98 (2010) 49–70.
[12] Q. Su, L. Chen, A method for discovering clusters of e-commerce interest patterns using click-stream data, Electron. Commer. Res. Appl. 14 (2015) 1–13.
[13] Z. Du, X. Wang, H. Yang, J. Zhou, J. Tang, Sequential scenario-specific meta learner for online recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2895–2904.
[14] Z. Gharibshah, X. Zhu, A. Hainline, M. Conway, Deep learning for user interest and response prediction in online display advertising, Data Sci. Eng. 5 (2020) 12–26.
[15] J.-H. Kim, B.-G. Kim, P.P. Roy, D.-M. Jeong, Efficient facial expression recognition algorithm based on hierarchical deep neural network structure, IEEE Access 7 (2019) 41273–41285.
[16] D. Jeong, B.-G. Kim, S.-Y. Dong, Deep joint spatiotemporal network (DJSTN) for efficient facial expression recognition, Sensors 20 (2020) 1936.

[17] S. Jaiswal, S. Virmani, V. Sethi, K. De, P.P. Roy, An intelligent recommendation system using gaze and emotion detection, Multimed. Tools Appl. 78 (2018) 14231–14250.
[18] J.-H. Kim, G.-S. Hong, B.-G. Kim, D.P. Dogra, deepGesture: deep learning-based gesture recognition scheme using motion sensors, Displays 55 (2018) 38–45.
[19] Z. Erenel, O.R. Adegboye, H. Kusetogullari, A new feature selection scheme for emotion recognition from text, Appl. Sci. 10 (2020) 5351.
[20] S. Kumar, M. Gahalawat, P.P. Roy, D.P. Dogra, B.-G. Kim, Exploring impact of age and gender on sentiment analysis using machine learning, Electronics 9 (2020) 374.
[21] K. Pasupa, T. Seneewong Na Ayutthaya, Hybrid deep learning models for Thai sentiment analysis, Cogn. Comput. (2021) 1–27.
[22] K. Pasupa, T. Seneewong Na Ayutthaya, Thai sentiment analysis with deep learning techniques: a comparative study based on word embedding, POS-tag, and sentic features, Sustain. Cities Soc. 50 (2019) 101615.
[23] J. Cho, K. Lee, E. Shin, G. Choy, S. Do, How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?, CoRR, arXiv:1511.06348, 2015.
[24] K. Pasupa, W. Sunhem, A comparison between shallow and deep architecture classifiers on small dataset, in: Proceeding of the 8th International Conference on Information Technology and Electrical Engineering (ICITEE 2016), Yogyakarta, Indonesia, 5–6 October 2016, 2016, pp. 390–395.
[25] J. Weston, F. Ratle, R. Collobert, Deep learning via semi-supervised embedding, in: Proceedings of the 25th International Conference on Machine Learning (ICML), ACM Press, Helsinki, Finland, 2008, pp. 639–655.
[26] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, PMLR, 2016, pp. 40–48.
[27] A. Odena, Semi-supervised learning with generative adversarial networks, CoRR, arXiv:1606.01583, 2016.
[28] K. Pasupa, S. Tungjitnob, S. Vatathanavaro, Semi-supervised learning with deep convolutional generative adversarial networks for canine red blood cells morphology classification, Multimed. Tools Appl. 79 (2020) 34209–34226.
[29] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, CoRR, arXiv:1610.02242, 2016.
[30] T. Miyato, S.-I. Maeda, M. Koyama, S. Ishii, Virtual adversarial training: a regularization method for supervised and semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2019) 1979–1993.
[31] M. Sajjadi, M. Javanmardi, T. Tasdizen, Regularization with stochastic transformations and perturbations for deep semi-supervised learning, in: Advances in Neural Information Processing Systems, 2016, pp. 1163–1171.
[32] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, Q.V. Le, Unsupervised data augmentation for consistency training, CoRR, arXiv:1904.12848, 2019.
[33] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E.D. Cubuk, A. Kurakin, H. Zhang, C. Raffel, Fixmatch: simplifying semi-supervised learning with consistency and confidence, CoRR, arXiv:2001.07685, 2020.
[34] D.-H. Lee, Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks, in: Proceedings of the 30th International Conference on Machine Learning (ICML) Workshop on Challenges in Representation Learning (WREPL), Atlanta, USA, 2013, pp. 1–6.
[35] C. Rosenberg, M. Hebert, H. Schneiderman, Semi-supervised self-training of object detection models, in: Proceedings of the 7th IEEE Workshops on Applications of Computer Vision (WACV/MOTION), IEEE, 2005, pp. 29–36.
[36] S. Tungjitnob, K. Pasupa, E. Thamwiwatthana, B. Suntisrivaraporn, SME user classification from click feedback on a mobile banking apps, in: Proceedings of the 27th International Conference on Neural Information Processing (ICONIP2020), 18-22 November 2020, Springer, Springer International Publishing, Bangkok, Thailand, 2020, pp. 256–264.
[37] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (2001) 1189–1232.
[38] T. Chen, C. Guestrin, Xgboost: a scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016), San Francisco, CA, USA, 13–17 August 2016, 2016, pp. 785–794.
[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR 2016), IEEE, Las Vegas, NV, USA, 2016, pp. 770–778.