Leveraging Smartphone Human Interaction Routine Behavior Task Mining and Modeling for Daily Stress Monitoring

HANSOO LEE[*], KAIST and KIST, Republic of Korea
TAEHYEON PARK, KAIST, Republic of Korea
YOUNGJI KOH, KAIST, Republic of Korea
JAE-GIL LEE, KAIST, Republic of Korea
UICHIN LEE[†], KAIST, Republic of Korea

With advancements in mobile sensing technologies, there is a growing need for scalable and interpretable stress monitoring solutions that remain robust over time. Existing smartphone passive sensing approaches rely on statistical app usage features or multi-modal sensor data, making them susceptible to distribution shift and feature evolution (e.g., schema drift), and adding model complexity. To address these challenges, we propose the Smartphone Human Interaction-based Routine Behavior Task Mining, Modeling, and Feature Extraction (SHIRBT-MMF) framework, which models stress-related behaviors by mining fine-grained interaction routine tasks rather than aggregated app usage patterns. SHIRBT-MMF leverages multi-level sequential pattern mining and large language model-based automated task modeling to extract interpretable and stable features from within-app UI state transitions. Unlike traditional methods that require hundreds of apps, SHIRBT focuses on a small, consistent set of routine-based tasks, mitigating covariate shift and feature evolution while improving model robustness. We validated SHIRBT-MMF through one- and four-month in-the-wild datasets with 26 participants, demonstrating that the SHIRBT-based personalized model achieves an average accuracy of 75%, outperforming baseline models by 5% while using only 3–6% of app types. Additionally, SHIRBT features remain stable over time, reducing covariate shift and ensuring reliable performance. With its expandability to other mental health, interpretability, and privacy-conscious design, the SHIRBT-MMF framework lays the foundation for personalized digital mental health monitoring.

CCS Concepts: • **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; *Empirical studies in ubiquitous and mobile computing*; • **Computing methodologies** → *Sequential pattern mining*; *Natural language processing*; • **Applied computing** → Health informatics.

Additional Key Words and Phrases: Smartphone sensing; Routine behavior mining; Sequential pattern mining; Explainable digital phenotyping; LLM-based task annotation; Distribution shift robustness; Feature evolution resilience; Stress monitoring

ACM Reference Format:
Hansoo Lee, Taehyeon Park, Youngji Koh, Jae-Gil Lee, and Uichin Lee. 2025. Leveraging Smartphone Human Interaction Routine Behavior Task Mining and Modeling for Daily Stress Monitoring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 9, 4, Article 187 (December 2025), 45 pages. https://doi.org/10.1145/3770644

[*]Now at the Korea Institute of Science and Technology (KIST), Seoul, Republic of Korea.
[†]Corresponding author.

Authors' Contact Information: Hansoo Lee, KAIST and KIST, Daejeon and Seoul, Republic of Korea, hansoolee@kist.re.kr, hansoo@kaist.ac.kr; Taehyeon Park, KAIST, Daejeon, Republic of Korea, gmdmf@kaist.ac.kr; Youngji Koh, KAIST, Daejeon, Republic of Korea, youngji@kaist.ac.kr; Jae-Gil Lee, KAIST, Daejeon, Republic of Korea, jaegil@kaist.ac.kr; Uichin Lee, KAIST, Daejeon, Republic of Korea, uclee@kaist.ac.kr.

1 INTRODUCTION

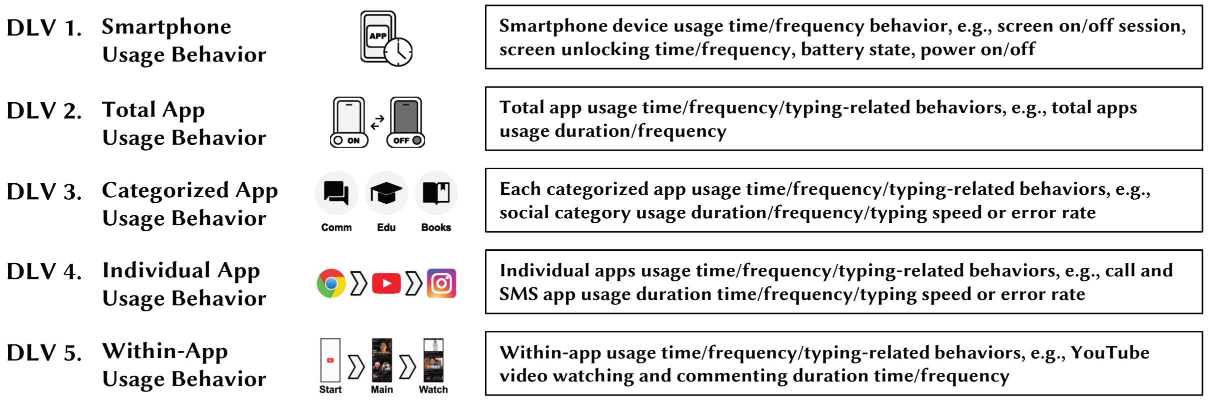
Global daily stress has been steadily rising, especially among young adults in their 20s and 30s facing academic, employment, financial, and future uncertainties [7, 9, 63]. To reduce healthcare and welfare costs and enhance productivity and academic achievement, early stress intervention and diagnostic technologies are crucial [15]. Young adults' stress is closely tied to daily routines and smartphone app use. Notifications from social media, messaging, and email disrupt concentration and heighten stress [41]. Negative self-reflection and the "fear of missing out (FOMO)" amplify stress and encourage addictive app use [41, 78]. Conversely, apps like games, webcomics, video calls, and music can temporarily relieve stress and support work or study [57]. Thus, smartphone use and stress interact in both positive and negative ways. Monitoring app behavior enables digital phenotyping [95] for early stress detection and intervention.

Mobile interaction–based mental health studies use passive sensing over weeks or months to monitor stress in real life, training ML models on smartphone features representing daily and social activities [70, 94]. However, extracting numerous features from hundreds of apps increases model complexity, reducing interpretability and performance [16, 18, 45]. Moreover, whenever the smartphone environment changes, such as through different app usage or operating system (OS) updates, data collection, preprocessing, feature extraction, and model retraining must be repeated, increasing time and cost. To reduce feature dimensionality, prior studies avoided fine-grained features based on specific app usage or within-app behaviors (Depth Level 4–5 in Figure 1). Instead, they extracted aggregated statistical features from time-windowed data, including total smartphone usage and category-level metrics (e.g., social, finance, and system apps) related to duration, frequency, and typing behavior (Depth Level 1–3 in Figure 1) [44, 68, 104]. However, such broad or category-based features capture only surface-level behaviors, and the multifunctionality of modern third-party apps makes single-category classification difficult.

Moreover, when relying on statistical features computed over fixed time windows, newly collected data after model deployment can cause distribution shifts (e.g., covariate shift and concept drift) [30, 65]. In addition, changes in app usage patterns, such as those resulting from app updates, UI/UX revisions, or event schema modifications, can trigger feature evolution (i.e., schema drift), which alters feature types or column structures [39, 69, 86, 105]. These shifts may reorder feature importance, and reusing outdated core features can degrade model performance on new datasets, thereby reducing robustness and reliability [3, 27, 65]. Although prior studies have addressed these problems by detecting drift and applying adaptive learning techniques such as online retraining, incremental learning, and domain adaptation [30, 65, 67], such approaches incur heavy computational and resource costs, making them impractical for smartphones with strict latency and memory constraints [67]. In contrast, this study introduces a robust feature design that, without adaptive learning, is inherently resilient to both covariate shift and schema drift.

In this study, we focus on daily and social activities tasks that individuals routinely perform, such as messaging, financial payments, emailing, and web browsing. These tasks remain stable despite variations in OS, app types, or contextual factors (e.g., environment and time), allowing covariate shift and schema drift to be mitigated without adaptive learning. Accordingly, we propose the **S**martphone **H**uman **I**nteraction-based **R**outine **B**ehavior **T**ask **M**ining, **M**odeling, and **F**eature extraction **(SHIRBT-MMF)** framework (Figure 2).

We utilize within-app user interface (UI) state events derived from app usage logs at Depth Level 5 in Figure 1, which include package names (e.g., com.example.myapp) and class names corresponding to user interactions (e.g., activity, dialog, frame layout, view layout). During smartphone interactions, users transition between UI states through actions such as clicking, typing, and scrolling to accomplish specific tasks (e.g., chatting, browsing, and video watching). We filter out events irrelevant to daily and social activities and focus on **smartphone human interaction–based routine behavior tasks (SHIRBT)** that are more closely related to daily stress. For this, we developed a **multi-level sequential pattern mining (ML-SPM)** algorithm that extracts sequential patterns of daily within-app UI state events. Since these patterns often contain unfamiliar package or class names,



DLV 1. Smartphone Usage Behavior — Smartphone device usage time/frequency behavior, e.g., screen on/off session, screen unlocking time/frequency, battery state, power on/off

DLV 2. Total App Usage Behavior — Total app usage time/frequency/typing-related behaviors, e.g., total apps usage duration/frequency

DLV 3. Categorized App Usage Behavior — Each categorized app usage time/frequency/typing-related behaviors, e.g., social category usage duration/frequency/typing speed or error rate

DLV 4. Individual App Usage Behavior — Individual apps usage time/frequency/typing-related behaviors, e.g., call and SMS app usage duration time/frequency/typing speed or error rate

DLV 5. Within-App Usage Behavior — Within-app usage time/frequency/typing-related behaviors, e.g., YouTube video watching and commenting duration time/frequency

Fig. 1. Depth level (DLV) of smartphone interaction usage behavior and feature examples

switching to a different app providing the same service (e.g., using Bing instead of Chrome for web browsing) requires re-extraction. To address this issue, we develop an LLM-based SHIRBT modeling automation system that unifies within-app behaviors into single, task-level representations (e.g., a web-browsing), independent of app or UI structure. The system was tested 200 times across various apps performing equivalent tasks with different event schemas, achieving 95–100% annotation accuracy consistent with SHIRBT ground-truth labels. Using the extracted SHIRBT and extending prior work on smartphone activities of daily living (S-ADL) [51], we computed time-, frequency-, and typing-based metrics (Figure 2) and built stress detection models using SHIRBT features from the SHIRBT-MMF framework.

To validate our model, we collected in-the-wild data from 26 students and workers in their 20s and 30s. Since the impact of SHIRBT features on stress may vary with the consistency of daily task performance, smartphone usage data were automatically logged via a custom app usage logger, and daily binary stress labels (positive or negative) were obtained using the experience sampling method (ESM) [97], validated in prior work [44]. Data were collected over two periods: a short-term phase (about one month, 16 participants) and a long-term phase (about four months, 10 participants). The average number of SHIRBTs was 7.6 and 6.1, with mean accuracies of 75.9%±8.7 and 73.8%±5.0, respectively, showing minimal performance differences between the two durations. To evaluate performance, we compared SHIRBT features with baseline features corresponding to Depth Levels 1–3 in Figure 1, as used in prior stress studies. The SHIRBT-based classifier achieved 5% higher accuracy while using only 3–6% as many apps as the baseline (Figure 2). Although SHIRBT tasks differ among individuals, essential social tasks such as messaging and financial transactions are largely consistent across age, gender, and occupation. Accordingly, the SHIRBT global model outperformed the baseline model, though its accuracy was slightly lower than that of the personalized model. Moreover, under covariate shift and schema drift conditions, SHIRBT features exhibited lower shift scores and higher structural consistency, confirming their robustness and long-term stability without additional retraining.

The contributions of the paper are as follows:

- We propose **SHIRBT-MMF**, a novel framework for stress monitoring that leverages smartphone interaction patterns instead of conventional app usage metrics. The framework improves scalability, interpretability, and robustness in digital stress monitoring.

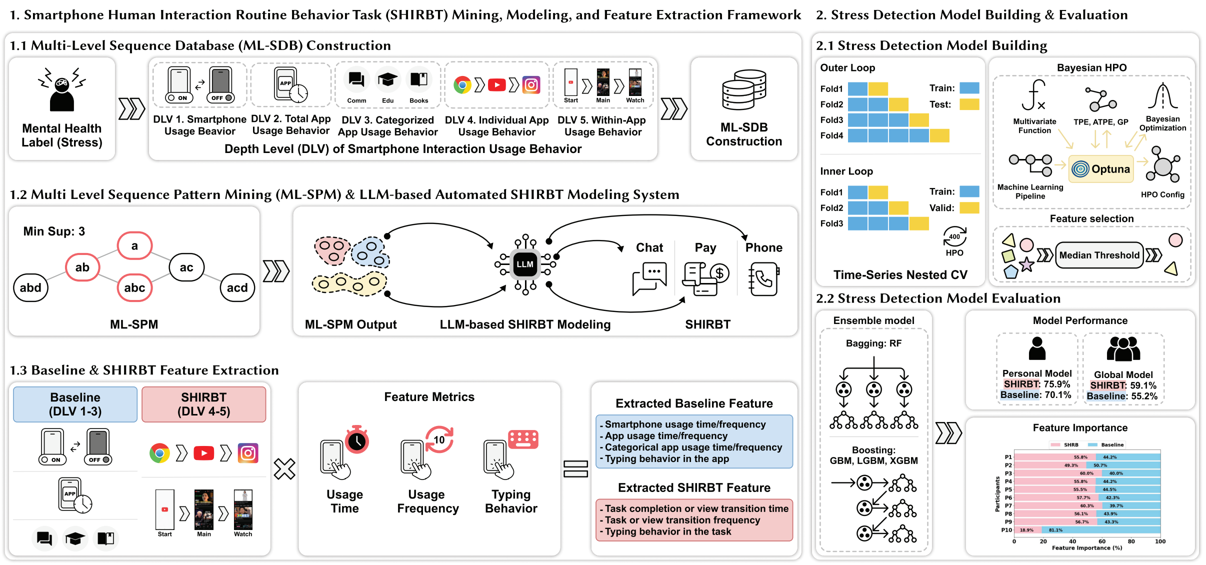

1. Smartphone Human Interaction Routine Behavior Task (SHIRBT) Mining, Modeling, and Feature Extraction Framework
2. Stress Detection Model Building & Evaluation
1.1 Multi-Level Sequence Database (ML-SDB) Construction
Mental Health Label (Stress)
ON
OFF
APP
Comm
Edu
Books
Start
Main
Watch
DLV 1. Smartphone Usage Beavior
DLV 2. Total App Usage Behavior
DLV 3. Categorized App Usage Behavior
DLV 4. Individual App Usage Behavior
DLV 5. Within-App Usage Behavior
Depth Level (DLV) of Smartphone Interaction Usage Behavior
ML-SDB Construction
1.2 Multi Level Sequence Pattern Mining (ML-SPM) & LLM-based Automated SHIRBT Modeling System
Min Sup: 3
a
ab
ac
abd
abc
acd
ML-SPM
ML-SPM Output
LLM
LLM-based SHIRBT Modeling
Chat
Pay
Phone
SHIRBT
1.3 Baseline & SHIRBT Feature Extraction
Baseline (DLV 1-3)
SHIRBT (DLV 4-5)
Feature Metrics
Usage Time
Usage Frequency
Typing Behavior
Extracted Baseline Feature
- Smartphone usage time/frequency
- App usage time/frequency
- Categorical app usage time/frequency
- Typing behavior in the app
Extracted SHIRBT Feature
- Task completion or view transition time
- Task or view transition frequency
- Typing behavior in the task
2.1 Stress Detection Model Building
Outer Loop
Fold1
Fold2
Fold3
Fold4
Train:
Test:
Inner Loop
Valid:
400 HPO
Time-Series Nested CV
Bayesian HPO
Multivariate Function
TPE, ATPE, GP
Bayesian Optimization
Machine Learning Pipeline
Optuna
HPO Config
Feature selection
Median Threshold
2.2 Stress Detection Model Evaluation
Ensemble model
Bagging: RF
Boosting: GBM, LGBM, XGBM
Model Performance
Personal Model
SHIRBT: 75.9%
Baseline: 70.1%
Global Model
SHIRBT: 59.1%
Baseline: 55.2%
Feature Importance
SHIRB
Baseline
Participants
Feature Importance (%)

Fig. 2. Smartphone human interaction-based routine behavior task mining, modeling, feature extraction (SHIRBT-MMF) framework & stress model pipeline

- We develop a **multi-level sequential pattern mining (ML-SPM)** algorithm that captures fine-grained within-app UI event sequences for daily task-level behavior mining. Unlike existing coarse-grained approaches that use statistical methods, this method identifies stable, routine-based behaviors that mitigate covariate shift and schema drift over time.
- We present an **LLM-based automated task classification and labeling** approach that standardizes and categorizes smartphone UI interaction sequences into explainable SHIRBT tasks. This enables the derivation of interpretable, high-quality stress-related features, enhancing both model transparency and explainability.
- We validate **SHIRBT-MMF** on in-the-wild datasets collected from 26 participants over one- and four-month periods. The SHIRBT-based personalized model outperformed the baseline models while using only 3–6% as many apps as the baseline.
- We confirmed that SHIRBT features showed greater robustness to **covariate shift and schema drift** than baseline features, maintaining stable performance and structural consistency over time. This robustness indicates that SHIRBT can serve as a cost-effective framework for long-term digital mental health monitoring without frequent model retraining.

2 BACKGROUND AND RELATED WORK

2.1 Smartphone Interaction Sensing-based Stress Models

We reviewed the past decade that developed stress models using features based on smartphone interaction in Table 1. Most studies incorporated smartphone or wearable-based interactions (e.g., app usage, device state) along with contextual data (e.g., location, physical activity). As noted in Section 1, while multi-modal sensor data improve human behavior analysis, they increase data collection and feature engineering costs. A large feature set reduces model interpretability (e.g., [44] used approximately 3,356 features). Previous research primarily employed aggregated statistical features derived through time windowing, such as total smartphone usage, app category duration, frequency, and typing behavior (e.g., speed and error rate). To reduce model complexity and improve interpretability, studies applied dimensionality reduction [85], feature elimination, or selection techniques, as summarized in Table 1.

However, app-level or category-level features are broad and surface-level, and the multifunctionality of modern third-party apps makes single-category classification challenging. As a result, time-windowed features provide limited insight into specific daily activities (e.g., finance payments, web browsing, messaging), reducing the model's ability to link behaviors to stress. Furthermore, relying on fixed-interval statistical features can cause distribution shifts after deployment, affecting feature selection or importance rankings and compromising robustness [2, 3, 27]. If these shifts are ignored, model performance may degrade on new datasets as outdated features remain in use.

2.2 Distribution Shift in Smartphone Interaction Sensing

In streaming-based ML environments, such as smartphone sensing, distribution shift (also called data drift or dataset shift) occurs when the source distribution p used during training differs from the target distribution f encountered during deployment [36, 65, 67]. Distribution shifts are generally categorized into three types [65, 79]: (1) **Covariate shift**, where the input distribution changes but the input–label relationship remains constant:

$$p_X(x) \neq f_X(x), \quad p_{Y|X}(y \mid x) = f_{Y|X}(y \mid x)$$

Table 1. Review of prior smartphone usage behavior-based stress detection model research. DLV 1: smartphone usage, DLV 2: total app usage, DLV 3: categorical app usage, DLV 4: individual app usage, and DLV 5: within-app usage behavior.

Studies	Smartphone Usage (duration time, frequency, typing) and Context Features	DLV	N	Aged	Duration	Model	Performance
[50]	- Total touch/typing usage (e.g., key press, number of backspace/character) - Location, weather	2	1	30s	2 weeks	Personal	67.5% acc
[10]	- Specific app usage (e.g., call, SMS) - Bluetooth proximity, location	4	7	-	1 month	Personal	Avg. 53% acc
[104]	- Device (e.g., screen on/off), app category, and specific app usage (e.g., call, SMS) - Mic, physical activity, location	1, 3, 4	30	18-30	1 month	Personal	Avg. 63% acc
[99]	- Device (e.g., screen on/off) and app category usage - Location	1, 3	28	20-47	100 days	Personal	59%–70% acc
[64]	- Specific app category (e.g., social and system) and app usage (e.g., call, SMS) - Mic, physical activity, location	3, 4	30	26-40	8 weeks	Personal	67.6%–71.7 acc
[47]	- Total app and app category usage - Physiological/physical activity, location	2, 3	36	19-33	25 days	Personal	Avg. 61-68% acc
[34]	- Total typing usage (e.g., typing duration, number of backspaces/characters)	2	22	24-33	3 weeks	Personal	Avg. 78% AUC
[19]	- Device (e.g., screen on/off/unlock) and app category usage - Total touch/typing usage (e.g., swipe, scroll, and text input) - Physical activity	1–3	25	18-57	1 month	Personal Global	- 77-88% F1 - 63-83% F1
[85]	- Device (e.g., screen on/off) and specific app usage (e.g., call, SMS) - Physiological/physical activity	1, 4	18	28±7.8	5 days	Global	75% acc
[84]	- Device (e.g., screen on/off) and specific app usage (e.g., call, SMS, internet, email) - Physiological/physical activity	1, 4	66	20.1±1.5	1 month	Global	67-92% acc
[71]	- Device (e.g., battery) and specific app usage (e.g., call, calendar) - Mic, physiological/physical activity	1, 4	35	25-62	4 months	Global	55% acc
[13]	- Specific app usage (e.g., call, SMS) - Bluetooth proximity, weather	4	117	-	5 months	Global	72.3% acc
[42]	- Device (e.g., screen on/off) and specific app usage (e.g., call, SMS) - Physiological/physical activity, location	1, 4	20, 48	18-24	1 month	Global	68.5% acc
[44]	- Device usage (e.g., screen on/off/unlock, battery, power, data traffic, network) - Total app, app category, and specific app usage (e.g., call, SMS) - Physiological/physical activity, location	1–4	77	17-38	1 week	Global	66.6% acc

(2) **Label shift**, where the label distribution changes while the conditional feature distribution remains the same:

$$p_Y(y) \neq f_Y(y), \quad p_{X|Y}(x \mid y) = f_{X|Y}(x \mid y)$$

(3) **Concept drift**, where the relationship between features and labels changes over time:

$$p_{Y|X}(y \mid x) \neq f_{Y|X}(y \mid x), \quad p_X(x) = f_X(x)$$

These types of shifts are prevalent in smartphone interaction datasets [1, 65, 67]. In addition, smartphone logs frequently experience **feature evolution** (also called schema drift), where existing features disappear or new ones emerge due to app updates, UI/UX modifications, or event schema changes [39, 69, 86, 105]. Such structural variations can affect feature extraction, alter feature importance, and reduce model reliability.

To address these issues, prior studies have adopted **adaptive learning** strategies, including online retraining, incremental learning, and domain adaptation [30, 36, 65, 67]. For example, Meegahapola et al. [65] applied a Domain-Adversarial Neural Network (DANN)-based Unsupervised Domain Adaptation (UDA) framework to mitigate covariate shift between training and deployment domains in multimodal mobile sensing. Hao et al. [36] proposed a self-supervised lightweight drift detection method that automatically identifies covariate shift on mobile devices and adapts models through by-cause adaptation under resource constraints. Ferjani and Alsaif [30] implemented incremental concept drift detection for smartphone accelerometer-based road anomaly detection. Hou et al. [39] proposed a model-switching approach to recover missing features in streaming data, while Zhang et al. [105] introduced an adaptive deep learning framework addressing both distribution drift and feature evolution simultaneously.

Although these adaptive methods effectively detect and respond to distribution shift or schema drift, they exhibit several inherent limitations. First, they are *reactive*, updating models only after performance degradation occurs due to drift. Second, adaptive learning methods incur significant computational and operational costs associated with continuous monitoring, retraining, and maintaining ensemble models, challenges that are particularly problematic in resource-limited mobile environments [67]. To overcome these challenges, we propose a proactive feature design approach that ensures robustness against both covariate shift and feature evolution. Our proposed SHIRBT features are derived from users' repetitive daily routines, making them inherently stable across OS, app, and contextual variations. These task-level features exhibit lower sensitivity to drift than conventional statistical features, enabling our SHIRBT-based framework to maintain reliable performance without frequent retraining and to offer a scalable, cost-efficient solution for real-world deployment.

2.3 Smartphone Interaction-based Human Behavior Mining

Prior studies have extracted smartphone interaction-based human behaviors using data mining techniques such as Association Rule Mining (ARM) and Sequential Pattern Mining (SPM) [40, 59, 72, 90]. These studies demonstrated that routine behaviors can be mined from passive smartphone sensing data combining contextual (e.g., time, location) and behavioral (e.g., smartphone app usage) information. Mukherji et al. [72] presented a Mobile Sequence Mining (MSM) engine that captures longitudinal on-device usage patterns while preserving privacy. Srinivasan et al. [90] mined app and location-based behaviors to enhance personalization and user experience through predictive shortcuts without significant battery drain. Lu et al. [59] introduced the MASP-Mine algorithm to identify Mobile Application Sequential Patterns (MASPs) and Spatial-Temporal App Usage Paths (STAR), while Hsu et al. [40] proposed a pattern-growth-based algorithm for mining time-constrained sequential patterns.

Collectively, these works showed that personalized app usage patterns or rules derived from SPM and ARM improve behavior interpretability and can reduce data volume, mitigate privacy concerns, and minimize energy consumption [40, 59, 72, 90]. However, these approaches only identified between-app usage patterns at DLV 4 (Figure 2) without offering interpretability for within-app UI state behaviors at DLV 5. They also failed to specify the temporal granularity of extracted patterns, missing daily smartphone interaction routines. Moreover, the derived patterns were not modeled into user-understandable task forms nor extended to broader applications such as mental health detection.

2.4 Smartphone-based Human Behavior Modeling for Mental Health Detection

Previous studies have detected stress and depression by leveraging smartphone multi-modal sensor data (e.g., location, physical activity, app usage) reflecting human routine behavior [12, 99–101]. Berrouiguet et al. [12] applied unsupervised mining of smartphone usage data to identify mobility pattern changes in depressed patients, while Xu et al. [100, 101] used ARM on context-filtered smartphone usage and contextual features to detect depression. Similarly, Vildjiounaite et al. [99] employed an unsupervised long-term stress monitoring approach using smartphone-based multimodal data (e.g., phone usage, physical activity) with Hiden Markov Model (HMM), achieving accuracy comparable to supervised methods.

However, prior studies primarily identified inter-app association rules without analyzing within-app UI trajectory (DLV 5 in Figure 2). They focused on combining multi-modal sensor features (e.g., location, action, and smartphone usage) to derive behavior rules rather than modeling consistent daily routine tasks, thus failing to effectively reduce data collection. In contrast, our study aims to enhance explainability and efficiency by modeling SHIRBT through within-app UI trajectory mining. To derive sequential and frequent UI trajectories, we adopt the SPM technique used in previous studies [40, 59, 72] for SHIRBT modeling.

2.5 Sequential Pattern Mining

Sequential Pattern Mining (SPM) discovers frequent subsequences and meaningful relationships in sequential data, which consists of ordered events over time [4]. It has been applied in diverse domains such as web usage analysis, scientific research, natural disaster tracking, customer behavior, protein structure, and disease treatment [31, 61]. With the expansion of e-commerce and the extensive potential of mobile apps, web/app usage mining has emerged as an essential area of focus [58, 61, 77]. As a critical application of SPM, web/app usage mining centers on extracting users' navigation patterns from various web/app usage data [58, 61].

Depending on the database structure, it is categorized into *page mining* and *content mining* [61]. Page mining identifies interaction sequences across screens (e.g., web access logs, clickstreams, traversals, mobile app usage) using singleton item databases (DBs), where each item represents a screen transition such as wep page or UI state event [28, 31, 54, 58, 77]. Content mining, by contrast, employs k-itemset DBs that record multiple items (e.g., products a, b, and c) generated simultaneously from user interactions with specific content (e.g., add to cart or purchase button) [29, 61]. In this study, we focus on page-level mining to derive SHIRBT by extracting within-app UI state sequential patterns and defining them as tasks. For example, a sequential pattern composed of within-app UI state events can represent a user's behavior in a chat app, such as navigating from the main screen (a) to the chatroom screen (b) and back, using a data structure like ⟨aba⟩.

In general, the key terms and elements (e.g., event set, sequence, sequence database, support, etc.) identified in previous studies in the process of constructing a singleton itemset DB composed of web pages or within-app UI states and performing SPM are as follows [28, 29, 53, 54, 61, 77]:

- **Event**: An event refers to a within-app UI state (e.g., activity, view, layout, dialog, etc.) that changes due to specific inputs during a user's interaction with a smartphone. For example, the main screen or a chatroom screen in a chat app can be considered an event.
- **Event set**: Define E as the set of all events.
- **Sequence**: A sequence S is a list of events arranged in order and is represented as $S = \langle e_1 e_2 \ldots e_n \rangle$, where $e_i \in E$ and $1 \leq i \leq n$.
- **Sequence length**: In the sequence $S = \langle e_1 e_2 \ldots e_n \rangle$, n denotes the length of sequence S. A sequence of length n is called an n-sequence.

- **Subsequence and supersequence**: Given two sequences $S_a = \langle a_1 a_2 \dots a_n \rangle$ and $S_b = \langle b_1 b_2 \dots b_m \rangle$, if there exist integers $1 \leq i_1 < i_2 < \cdots < i_m \leq n$ such that $b_j = a_{i_j}$ for $1 \leq j \leq m$, then S_b is a subsequence of S_a, and S_a is a supersequence of S_b. This relationship is simply denoted as $S_b \subseteq S_a$ when S_b is a subsequence of S_a.
- **Sequence identification (Sid)**: A special value (e.g., timestamp, customer ID, user ID) used to identify each sequence is defined as sequence identification or *sid*.

Table 2. Example of sequence database and sequential pattern mining

(a) Example of Sequence Database (Input)

SID	Sequence
1	<acd>
2	<acbcabae>
3	<ecdabd>
4	<ac>
5	<abc>

(b) Example of Sequential Pattern Mining (Output)

No.	Patterns (minsup: 3)
1	<a>(support: 5)
2	<c>(support: 5)
3	<ac>(support: 4)
4	<b>(support: 3)
5	<ab>(support: 3)

- **Sequence database (SDB)**: A SDB is a set of tuples $\langle sid, S \rangle$, where S is a sequence and *sid* is its identifier.
- **Support**: The support of sequence S_a in SDB is the number of tuples that include S_a. Here, "include" means that S_a is a subsequence of S. The support of sequence S_a in SDB is defined as follows.

$$support_{SDB}(S_a) = |\{\langle sid, S \rangle \mid (\langle sid, S \rangle \in SDB \land (S_a \subseteq S))\}| \tag{1}$$

  If the SDB is clear from context, $support_{SDB}(S_a)$ can be abbreviated as $sup(S_a)$.
- **Sequential Pattern**: A sequential pattern is defined as a sequence in SDB that has support greater than or equal to a minimum support threshold (minsup). The minsup is a user-defined integer greater than 0. Formally, a sequence S is defined as a sequential pattern if the following condition is satisfied.

$$support_{SDB}(S) \geq minsup \tag{2}$$

  For example, in the SDB (Table 2), whereminsup = 3, the frequent patterns are $\langle a \rangle$, $\langle b \rangle$, $\langle c \rangle$, $\langle ab \rangle$, and $\langle ac \rangle$. Importantly, this includes both contiguous subsequences (e.g., $\langle ab \rangle$) and non-contiguous subsequences (e.g., $\langle ac \rangle$ when sid = 5) that meet the minimum support threshold.

3 METHODOLOGY: SHIRBT-MMF FRAMEWORK

We proposed a **Smartphone Human Interaction-based Routine Behavior Task Mining, Modeling, and Feature Extraction (SHIRBT-MMF) framework** process consisting of four phases: Section 3.1.1 explains the limitations of traditional SPM for SHIRBT mining and the need for a multi-level SPM approach. Section 3.1.2 and 3.1.3 presented the construction of a multi-level SDB and the process of multi-level SPM for SHIRBT mining. Section 3.2 explained how the mined SHIRB patterns are modeled into SHIRBT using LLMs for interpretability, and Section 3.3 described SHIRBT-based feature extraction for quantitative stress analysis.

3.1 Development of Multi-Level Sequential Pattern Mining Algorithm

3.1.1 ***Necessity of Multi-Level Sequential Pattern Mining.*** To mine and model SHIRBT, we constructed an SDB and performed SPM by considering two dimensions: temporal context (e.g., time epoch) and in-app action (e.g., sequential pattern based on within-app UI state events). Following previous studies [40, 59, 72, 90], human routine behavior can be mined from user context (e.g., time) and in-app action (e.g., smartphone app usage) domains. Thus, we defined the time epoch as a daily interval to mine within-app UI state events-based sequential

Original Database

Day	Screen On/Off Session	Within-app UI State Event Data
6/14	1	⟨caba⟩
6/14	2	⟨cd⟩
6/15	3	⟨dab⟩
6/15	4	⟨cd⟩
6/16	5	⟨abd⟩

(a) within-app UI State Event-based DB

SDB (Input) ⟶ **SPM (Output)**

SID (Day)	Sequences
6/14	⟨cabacd⟩
6/15	⟨dabcd⟩
6/16	⟨abd⟩

SID = Sequence identification is a day

No.	Patterns (Minsup: 3)
1	⟨a⟩ (sup: 3)
2	⟨b⟩ (sup: 3)
3	⟨d⟩ (sup: 3)
4	⟨ab⟩ (sup: 3)
5	⟨ad⟩ (sup: 3)
6	⟨bd⟩ (sup: 3)
7	⟨abd⟩ (sup: 3)

(b) Single-SID based SDB and SPM Outputs

SDB (Input) ⟶ **SPM (Output)**

SID1 (Day)	SID2 (Session)	Sequences
6/14	1	⟨caba⟩
6/14	2	⟨cd⟩
6/15	3	⟨dab⟩
6/15	4	⟨cd⟩
6/16	5	⟨abd⟩

Session = (Screen On/Off) Session

No.	Patterns (Minsup: 3)
1	⟨d⟩ (sup: 4)
2	⟨a⟩ (sup: 3)
3	⟨b⟩ (sup: 3)
4	⟨c⟩ (sup: 3)
5	⟨ab⟩ (sup: 3)

(c) Double-SID based SDB and SPM Outputs

ML-SDB (Input) ⟶ **ML-SPM (Output)**

SID ⟨Day, Session⟩	Sequences
⟨Day: 6/14, Session: 1⟩	⟨caba⟩
⟨Day: 6/14, Session: 2⟩	⟨cd⟩
⟨Day: 6/15, Session: 3⟩	⟨dab⟩
⟨Day: 6/15, Session: 4⟩	⟨cd⟩
⟨Day: 6/16, Session: 5⟩	⟨abd⟩

SID = Sequence identification is a tuple consisting of Day and (Screen On/Off) Session

No.	Patterns (Minsup: 3)
1	⟨a⟩ (sup: 3)
2	⟨b⟩ (sup: 3)
3	⟨d⟩ (sup: 3)
4	⟨ab⟩ (sup: 3)

(d) Multi-Level-SID based ML-SDB and ML-SPM Output

Fig. 3. Limitations of sequential pattern mining based on existing sequence database structures and the proposed multi-level sequence database and multi-level sequential pattern mining method

patterns that occur daily. Furthermore, to extract only the actions during smartphone interactions, we focused on within-app UI state event data corresponding to app usage sequences between screen-on and screen-off events (screen on/off session). Based on this, we constructed an SDB that incorporated three dimensions of data: day, screen on/off session, and within-app UI state event.

We adopted the tuple structure ⟨sid,S⟩ commonly used in web log-based SPM studies [31, 38, 40, 61, 72, 77] based on the SDB structure presented in Figure 3b. The *sid* represents the day for daily support counting, and Sequence *S* denotes the within-app UI state events occurring during the first screen on/off session of the day. However, the single-*sid* based SDB structure can consider one screen on/off session among multiple sessions within a day, preventing the mining of sequential patterns for individual screen on/off sessions across the day. For example, the sequence ⟨caba⟩ associated with *sid* (Day) for June 14 corresponds to the first screen on/off session of that day, whereas the sequence ⟨cd⟩ from the second screen on/off session on the same day (June 14) could not be included. Therefore, the single-*sid*-based SDB structure includes only sequences from specific sessions within a day, excluding those from other same-day sessions and preventing a comprehensive analysis of the entire day.

To address this limitation, we defined *sid* as the day for daily support counting and *S* as all within-app UI state events occurring from the first screen-on to the last screen-off within that day. In this configuration, *S* represents a continuous chronological sequence that ignores session boundaries, causing independent sessions separated by hours to be mined as one. For instance, the sequence ⟨*cabacd*⟩ for June 14 combines two sessions, allowing subsequences such as ⟨*acd*⟩, ⟨*ac*⟩, and ⟨*ad*⟩ to appear as a single pattern even though they originate from different sessions. This highlights the need for an SDB structure and SPM method that accounts for multiple screen on/off sessions within a day and extracts sequential patterns solely from within-session events.

To extract sequential patterns from within-app UI state events in each screen on/off session, we performed SPM using Double-*sid*-based SDB containing two identifiers: sid_1 (day) and sid_2 (screen on/off), as illustrated in

Figure 3c. However, conventional SPM algorithms [31, 35, 61, 77] could not simultaneously process dual identifiers, leading to support-counting inconsistencies. When considering sid_1 (day), extracting session-level sequential patterns became difficult, whereas using sid_2 (screen on/off) prevented accurate daily support aggregation. For instance, with a minimum support threshold (minsup) of 3, seven patterns (e.g., $\langle a\rangle$, $\langle b\rangle$, $\langle c\rangle$, $\langle d\rangle$, $\langle ab\rangle$, $\langle ac\rangle$, $\langle ad\rangle$) were mined. Yet, under sid_2, the pattern $\langle d\rangle$ appeared across three days (June 14–16) and should have a support of 3, but duplicate sessions on June 15 increased it to 4. Similarly, $\langle c\rangle$ spanned two days (June 14–15) but was redundantly counted due to multiple sessions on June 14, yielding a support of 3. To resolve these inconsistencies, we proposed the **Multi-Level Sequence Database (ML-SDB)** and **Multi-Level Sequential Pattern Mining (ML-SPM)** algorithm (Figure 3d), which accounts for the hierarchical relationship between day and session. Detailed procedures for ML-SDB construction and ML-SPM were presented in Sections 3.1.2 and 3.1.3.

3.1.2 *Multi-Level Sequence Database Construction.*

We preprocessed the within-app UI state event data before constructing the ML-SDB (Figure 3d). To capture transitions within-app usage states during user–app interactions within each screen on/off session, we logged *WindowStateChanged* events using Android's built-in APIs: Accessibility Event [21] and Accessibility Service [22], as recommended by recent Android sensing surveys [52]. These events indicate the app's UI structure changes, such as transitions between activities, layouts, or dialogs (Figure 4). Each *WindowStateChanged* event was recorded as a Fully Qualified Class Name (FQCN), for example, `com.kakao.talk.activity.main.MainActivity` or `com.kakao.talk.android.widget.FrameLayout`, comprising the package name (a unique app identifier, e.g., `com.kakao.talk`) and the class name (e.g., `ChatRoomHolderActivity` or `FrameLayout`), which represent UI elements that users directly interact with.

To abbreviate *WindowStateChanged* events, we removed the top-level domain (e.g., `com`, `org`) and used the first letter of the package name, which represents the company, app, or service, combined with the initials of each word in the class name (e.g., `com.kakao.talk.activity.main.MainActivity` $\rightarrow$ `KT.MA`). In some cases, *WindowStateChanged* events contained two package names, such as `com.naver.whale.org.chromium.chrome.browser.ChromeTabbedActivity`. This occurred when Android libraries (e.g., Chromium) or modularized code were reused. For instance, a browser app (e.g., `com.naver.whale`) could use APIs from an open-source library (e.g., `org.chromium.chrome` [93]), resulting in a combined structure of two package names and a class name (e.g., `com.naver.whale.org.chromium.chrome.browser.ChromeTabbedActivity`, abbreviated as `NW.CCB.CTA`). We excluded events that were not generated during user interactions within screen on/off sessions. Additionally, automatically generated system events within sessions (e.g., `com.android.systemui.android.widget.FrameLayout`) were also excluded.

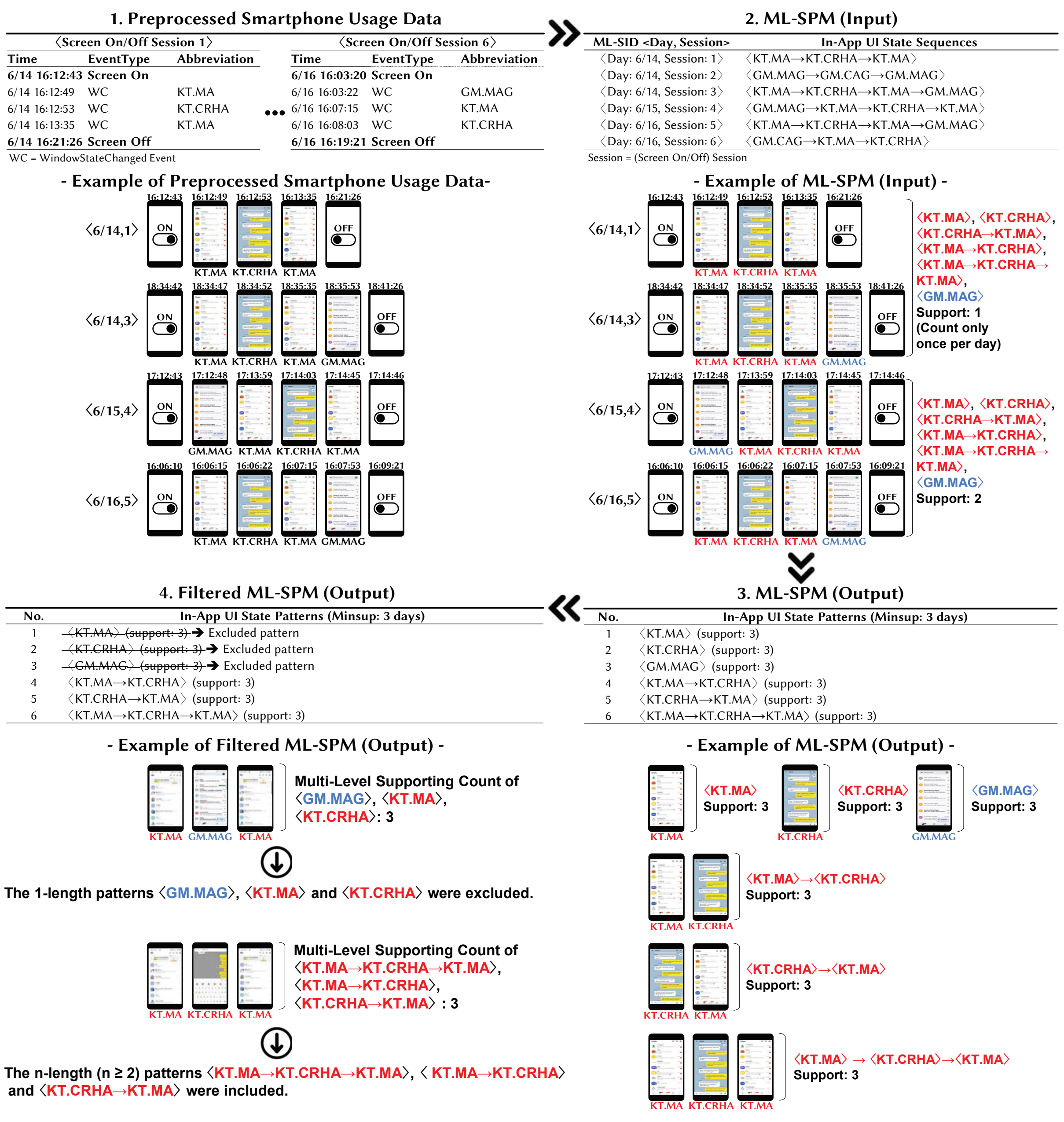
We constructed an ML-SDB based on Multi-Level sequence identification (ML-sid) to perform Multi-Level Sequential Pattern Mining (ML-SPM) using the preprocessed smartphone usage data, as shown in Figure 3d. The definitions of ML-sid and ML-SDB are as follows.

- **Multi-Level sequence identification**: ML-sid is defined in the form of a tuple ⟨day, screen on/off session⟩ (e.g., ⟨June 14, 1⟩, ⟨June 14, 2⟩) to simultaneously account for the hierarchical structure between day and screen on/off session during the support counting process.
- **Multi-Level sequence database**: The ML-SDB (Figure 3d) is defined as a set of tuples ⟨ML-sid, sequence⟩, where each tuple consists of an ML-sid and a corresponding within-app UI state event-based sequence.

The ML-SDB constructed from preprocessed smartphone usage data to perform ML-SPM can be represented as shown in the example in Figure 4. Specifically, in the example ML-SDB presented in Figure 4, the "→" symbol is used to distinguish each *WindowStateChanged* event and clearly indicate the chronological order of within-app UI state events.

3.1.3 *Multi-Level Sequential Pattern Mining Algorithm.*

We proposed an ML-SPM algorithm to extract within-app UI state events-based sequential patterns performed daily from the constructed ML-SDB. The existing



1. Preprocessed Smartphone Usage Data

⟨Screen On/Off Session 1⟩

Time	EventType	Abbreviation
6/14 16:12:43	**Screen On**	
6/14 16:12:49	WC	KT.MA
6/14 16:12:53	WC	KT.CRHA
6/14 16:13:35	WC	KT.MA
6/14 16:21:26	**Screen Off**	

•••

⟨Screen On/Off Session 6⟩

Time	EventType	Abbreviation
6/16 16:03:20	**Screen On**	
6/16 16:03:22	WC	GM.MAG
6/16 16:07:15	WC	KT.MA
6/16 16:08:03	WC	KT.CRHA
6/16 16:19:21	**Screen Off**	

WC = WindowStateChanged Event

2. ML-SPM (Input)

ML-SID <Day, Session>	In-App UI State Sequences
⟨Day: 6/14, Session: 1⟩	⟨KT.MA→KT.CRHA→KT.MA⟩
⟨Day: 6/14, Session: 2⟩	⟨GM.MAG→GM.CAG→GM.MAG⟩
⟨Day: 6/14, Session: 3⟩	⟨KT.MA→KT.CRHA→KT.MA→GM.MAG⟩
⟨Day: 6/15, Session: 4⟩	⟨GM.MAG→KT.MA→KT.CRHA→KT.MA⟩
⟨Day: 6/16, Session: 5⟩	⟨KT.MA→KT.CRHA→KT.MA→GM.MAG⟩
⟨Day: 6/16, Session: 6⟩	⟨GM.CAG→KT.MA→KT.CRHA⟩

Session = (Screen On/Off) Session

- Example of Preprocessed Smartphone Usage Data-
⟨6/14,1⟩ 16:12:43 ON 16:12:49 KT.MA 16:12:53 KT.CRHA 16:13:35 KT.MA 16:21:26 OFF
⟨6/14,3⟩ 18:34:42 ON 18:34:47 KT.MA 18:34:52 KT.CRHA 18:35:35 KT.MA 18:35:53 GM.MAG 18:41:26 OFF
⟨6/15,4⟩ 17:12:43 ON 17:12:48 GM.MAG 17:13:59 KT.MA 17:14:03 KT.CRHA 17:14:45 KT.MA 17:14:46 OFF
⟨6/16,5⟩ 16:06:10 ON 16:06:15 KT.MA 16:06:22 KT.CRHA 16:07:15 KT.MA 16:07:53 GM.MAG 16:09:21 OFF
- Example of ML-SPM (Input) -
⟨KT.MA⟩, ⟨KT.CRHA⟩, ⟨KT.CRHA→KT.MA⟩, ⟨KT.MA→KT.CRHA⟩, ⟨KT.MA→KT.CRHA→KT.MA⟩, ⟨GM.MAG⟩ Support: 1 (Count only once per day)
⟨KT.MA⟩, ⟨KT.CRHA⟩, ⟨KT.CRHA→KT.MA⟩, ⟨KT.MA→KT.CRHA⟩, ⟨KT.MA→KT.CRHA→KT.MA⟩, ⟨GM.MAG⟩ Support: 2

3. ML-SPM (Output)

No.	In-App UI State Patterns (Minsup: 3 days)
1	⟨KT.MA⟩ (support: 3)
2	⟨KT.CRHA⟩ (support: 3)
3	⟨GM.MAG⟩ (support: 3)
4	⟨KT.MA→KT.CRHA⟩ (support: 3)
5	⟨KT.CRHA→KT.MA⟩ (support: 3)
6	⟨KT.MA→KT.CRHA→KT.MA⟩ (support: 3)

4. Filtered ML-SPM (Output)

No.	In-App UI State Patterns (Minsup: 3 days)
1	~~⟨KT.MA⟩ (support: 3)~~ ➔ Excluded pattern
2	~~⟨KT.CRHA⟩ (support: 3)~~ ➔ Excluded pattern
3	~~⟨GM.MAG⟩ (support: 3)~~ ➔ Excluded pattern
4	⟨KT.MA→KT.CRHA⟩ (support: 3)
5	⟨KT.CRHA→KT.MA⟩ (support: 3)
6	⟨KT.MA→KT.CRHA→KT.MA⟩ (support: 3)

- Example of ML-SPM (Output) -
⟨KT.MA⟩ Support: 3
⟨KT.CRHA⟩ Support: 3
⟨GM.MAG⟩ Support: 3
⟨KT.MA⟩→⟨KT.CRHA⟩ Support: 3
⟨KT.CRHA⟩→⟨KT.MA⟩ Support: 3
⟨KT.MA⟩ → ⟨KT.CRHA⟩→⟨KT.MA⟩ Support: 3
- Example of Filtered ML-SPM (Output) -
KT.MA GM.MAG KT.MA
Multi-Level Supporting Count of ⟨GM.MAG⟩, ⟨KT.MA⟩, ⟨KT.CRHA⟩: 3
KT.MA KT.CRHA KT.MA
Multi-Level Supporting Count of ⟨KT.MA→KT.CRHA→KT.MA⟩, ⟨KT.MA→KT.CRHA⟩, ⟨KT.CRHA→KT.MA⟩ : 3

The 1-length patterns ⟨GM.MAG⟩, ⟨KT.MA⟩ and ⟨KT.CRHA⟩ were excluded.

The n-length (n ≥ 2) patterns ⟨KT.MA→KT.CRHA→KT.MA⟩, ⟨ KT.MA→KT.CRHA⟩ and ⟨KT.CRHA→KT.MA⟩ were included.

Fig. 4. Example of smartphone usage data preprocessing and the input & output of multi-Level sequential pattern mining

Abbreviations:
KT.MA = `com.kakao.talk.activity.main.MainActivity`,
KT.CRHA = `com.kakao.talk.activity.chatroom.ChatRoomHolderActivity`,
GM.MAG = `com.google.android.gm.ui.MailActivityGmail`,
GM.CAG = `com.google.android.gm.ui.ComposeActivityGmail`

SPM method counts the support of identical subsequences across multiple screen on/off sessions on the same day redundantly, and generates both non-contiguous and contiguous sequential patterns as explained in section 2.5 and 3.1.1. To count the support of subsequences appearing in multiple screen-on/off sessions on the same day without redundancy and to reflect the continuity of users' within-app usage behavior, we proposed a new support counting method called multi-level support (ML-sup). The definitions were described as follows.

- **Contiguous subsequence:** When a sequence of within-app UI state $S_b = \langle b_1, b_2, \ldots, b_m \rangle$ is a subsequence of a sequence of within-app UI state $S_a = \langle a_1, a_2, \ldots, a_n \rangle$, if integers $i_1, i_2, \ldots, i_m$ satisfy $i_{k+1} - i_k = 1$ for $(1 \le k < m)$, then S_b is a contiguous subsequence of S_a, denoted as $S_a \preceq S_b$. For example, for the sequence ⟨abc⟩, the subsequences ⟨ab⟩ and ⟨bc⟩ are contiguous subsequences of ⟨abc⟩, while ⟨ac⟩ is not a contiguous subsequence.
- **Contiguous sequential pattern:** A contiguous sequential pattern is a sequential pattern that qualifies as a contiguous subsequence.
- **Multi-Level support:** Multi-Level support is a support counting method that counts the support of a subsequence across multiple screen-on/off sessions on the same day without redundancy. To extract only contiguous within-app UI state events-based sequential patterns from the constructed ML-SDB, we defined Multi-Level support$_{\text{ML-SDB}}$ (ML-sup) as follows.

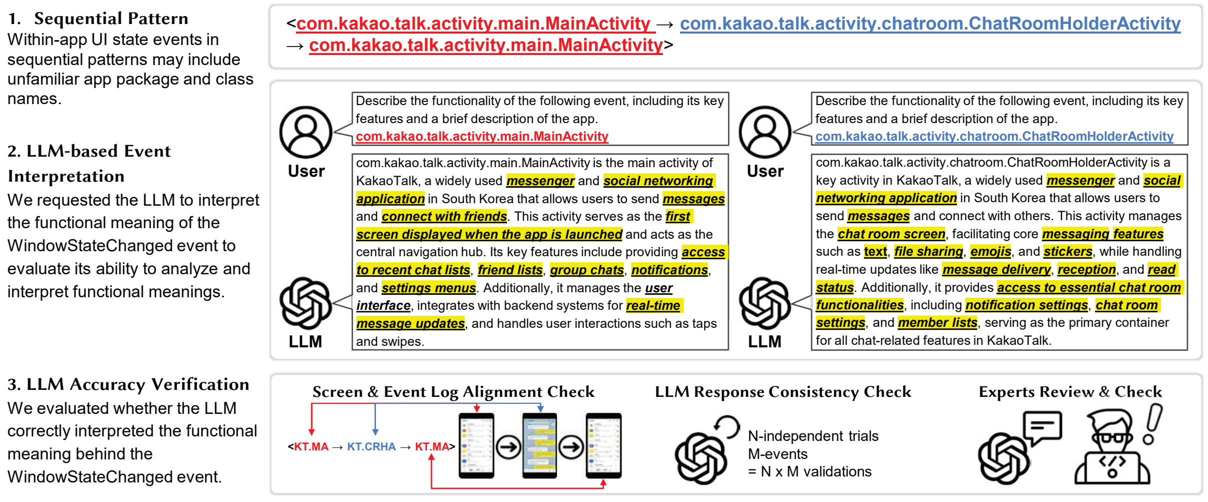
$$\text{Multi-level support}_{\text{ML-SDB}}(S_a) = \left|\{\text{day} \mid (\langle\langle \text{day, screen on/off session}\rangle, S\rangle \in \text{ML-SDB} \wedge (S_a \preceq S))\}\right|$$

For example, when ML-SPM is applied to the ML-SDB in Figure 4, the contiguous within-app UI state events-based sequential patterns ⟨KT.MA⟩, ⟨KT.CRHA⟩⟨KT.CRHA⟩, and ⟨KT.MA → KT.CRHA⟩ appear in 5 sequences (i.e., session 1 & 3 on June 14, 4 on June 15, and 5 & 6 on June 16). Similarly, ⟨KT.CRHA → KT.MA⟩ and ⟨KT.MA → KT.CRHA → KT.MA⟩ appear in 4 sequences (i.e., session 1 & 3 on June 14, 4 on June 15, and 5 on June 16). ⟨GM.MAG⟩ appears in 4 sequences (i.e., session 2 & 3 on June 14, 3 on June 15, and 5 on June 16). For sequential patterns ⟨KT.MA⟩, ⟨KT.CRHA⟩, ⟨KT.MA → KT.CRHA⟩, and ⟨GM.MAG⟩ that appear in both session 1 & 3 or 2 & 3 on June 14, the ML-sup for June 14 is counted as 1 without redundant counting. Similarly, for sequential patterns ⟨KT.MA⟩, ⟨KT.CRHA⟩, and ⟨KT.MA → KT.CRHA⟩ that appear in both session 5 & 6 on June 16, the ML-sup for June 16 is also counted as 1, not 2. To extract the SHIRB tasks from the mined contiguous within-app UI state events-based sequential patterns, it is necessary to interpret the meaning of human routine tasks based on both the within-app UI state events and their order. However, a sequence with a length of 1 (1-sequence) is difficult to interpret as a human routine behavior based on the order of events. Therefore, sequential patterns with a length of 1 were excluded from the extracted sequential patterns, such as ⟨GM.MAG⟩, ⟨KT.MA⟩, and ⟨KT.CRHA⟩, as shown in Figure 4.

Within-app UI state events-based sequential pattern data is characterized by a large number of sequences, the presence of very long sequences, and a wide variety of events. Therefore, considering the nature of the data, we applied a relatively effective pattern-growth method (e.g., PrefixSpan [35]) compared to traditional apriori algorithms (e.g., Apriori All/Some, GSP, SPAM, SPADE [4, 8, 89, 102]). The pattern-growth approach eliminates candidate generation and leverages database projection to efficiently reduce the search space and the number of database scans, thereby minimizing memory consumption and improving processing speed. The detailed implementation of the pattern-growth-based ML-SPM algorithm is described in Appendix A.

3.2 Development of LLM-based Automated SHIRBT Modeling System

We modeled sequential patterns derived from ML-SPM, which are composed of within-app UI state events, as **Smartphone Human Interaction–based Routine Behavior Tasks (SHIRBT)**. This modeling adopted the concept of smartphone-based Activities of Daily Living (S-ADL) [51] to generate explainable and human-interpretable features for users and service providers. Each within-app UI state event in the sequential patterns consists of an app package and a class name. While some events can be intuitively understood from their names, many are unfamiliar to non-technical users or system operators. Furthermore, the large number of patterns and events makes it impractical for service providers to manually assign heuristic SHIRBT labels. To address this challenge, we developed an **LLM-based SHIRBT annotation system** that automatically maps the functional meaning of event sequences to human-understandable SHIRBT.



1. Sequential Pattern
Within-app UI state events in sequential patterns may include unfamiliar app package and class names.

<com.kakao.talk.activity.main.MainActivity → com.kakao.talk.activity.chatroom.ChatRoomHolderActivity → com.kakao.talk.activity.main.MainActivity>

2. LLM-based Event Interpretation
We requested the LLM to interpret the functional meaning of the WindowStateChanged event to evaluate its ability to analyze and interpret functional meanings.

User: Describe the functionality of the following event, including its key features and a brief description of the app. com.kakao.talk.activity.main.MainActivity

LLM: com.kakao.talk.activity.main.MainActivity is the main activity of KakaoTalk, a widely used ***messenger*** and ***social networking application*** in South Korea that allows users to send ***messages*** and ***connect with friends***. This activity serves as the ***first screen displayed when the app is launched*** and acts as the central navigation hub. Its key features include providing ***access to recent chat lists***, ***friend lists***, ***group chats***, ***notifications***, and ***settings menus***. Additionally, it manages the ***user interface***, integrates with backend systems for ***real-time message updates***, and handles user interactions such as taps and swipes.

User: Describe the functionality of the following event, including its key features and a brief description of the app. com.kakao.talk.activity.chatroom.ChatRoomHolderActivity

LLM: com.kakao.talk.activity.chatroom.ChatRoomHolderActivity is a key activity in KakaoTalk, a widely used ***messenger*** and ***social networking application*** in South Korea that allows users to send ***messages*** and connect with others. This activity manages the ***chat room screen***, facilitating core ***messaging features*** such as ***text***, ***file sharing***, ***emojis***, and ***stickers***, while handling real-time updates like ***message delivery***, ***reception***, and ***read status***. Additionally, it provides ***access to essential chat room functionalities***, including ***notification settings***, ***chat room settings***, and ***member lists***, serving as the primary container for all chat-related features in KakaoTalk.

3. LLM Accuracy Verification
We evaluated whether the LLM correctly interpreted the functional meaning behind the WindowStateChanged event.

Screen & Event Log Alignment Check
<KT.MA → KT.CRHA → KT.MA>
LLM Response Consistency Check
N-independent trials
M-events
= N x M validations
Experts Review & Check

Fig. 5. Validating LLM's capability of understanding functional meaning of UI state events

Abbreviations:
KT.MA = `com.kakao.talk.activity.main.MainActivity`,
KT.CRHA = `com.kakao.talk.activity.chatroom.ChatRoomHolderActivity`

3.2.1 *Validating LLM's Capability of Understanding Functional Meaning of UI State Events.*

Before performing clustering, we verified whether the LLM could accurately interpret the functional meaning of within-app UI state events. Each event consisted of a package name and class name, and we used 20–30 representative examples from the most frequently used apps among young adults in Korea (e.g., KakaoTalk, Chrome, Naver, Gmail) [91, 92] (Figure 5). We used prompts related to interpreting the functional meaning of each event (e.g., KT.MA, KT.CRHA) corresponding to contiguous within-app UI state events-based sequential patterns (e.g., ⟨KT.CRHA → KT.MA⟩) derived through the ML-SPM process. These prompts were applied to the LLM (GPT-4o [75]) to extract the characteristics of app package names and the functional meanings of class names within the app, as shown in Figure 5.

We evaluated the LLM's interpretive accuracy through three tests (Figure 5). First, we manually performed various app tasks while logging events and recording screens to confirm that the LLM's interpretations matched the actual functions observed at the same timestamps. Second, we assessed consistency by reinterpreting 20 representative events per app five times, obtaining identical results in all cases (100% consistency). Third, three experts—including app developers and researchers familiar with usage log analysis—verified the correctness of the LLM's interpretations. Finally, we confirmed that the model also accurately and consistently interpreted events from both system apps (e.g., `com.android.launcher3.proxy.ProxyActivityStarter`) and sideloaded apps not available in official app stores (e.g., `com.example.speaker.controller.MainActivity`).

3.2.2 *Developing and Standardizing SHIRBT Naming Rules for LLM Training.*

We established standards and rules for task labeling to derive human-understandable and consistent SHIRBTs from the sequential patterns identified through ML-SPM using LLMs. We conducted a heuristic analysis based on expert and user evaluations across various cases to develop these standards and rules. The expert and user evaluations for SHIRBT labeling were conducted using eight evaluation criteria: Generality, Specificity, Comprehensibility, Accuracy, Comprehensiveness, Distinctiveness, Consistency, and Universality. Each criterion was rated on a 5-point Likert scale (with higher scores indicating better compliance with the criteria). Detailed descriptions of the evaluation criteria for SHIRBT modeling are as follows.

- **Generality:** How broadly a task name can be applied across various apps and services without being restricted to specific apps or app services?
- **Specificity:** How precisely does a task name describe the function of a class name?
- **Comprehensibility:** How clear and understandable is the task name from the perspective of general users?
- **Accuracy:** How accurately does the task name reflect the actual meaning of the function?
- **Comprehensiveness:** How well does the task name encompass the diverse functions or scenarios of a class name within the app?
- **Distinctiveness:** How clearly can the task name be distinguished from those of class names with different functions?
- **Consistency:** How consistently are naming conventions and classification standards applied throughout the entire task framework?
- **Universality:** Whether the task name is commonly used to describe tasks in general smartphone usage.

We proposed three naming rule candidates for SHIRBT modeling based on eight evaluation criteria: (1) general behavior-based tasks, (2) service-specific behavior-based tasks, and (3) app-specific behavior-based tasks. Through evaluations by experts and general users, we selected the naming rule with the highest score, as shown in Table 3 and Appendix B.1. For example, as illustrated in Figure 5, the sequence ⟨KT.CRHA → KT.MA⟩, interpreted by the LLM, represents the action of chatting in a chatting room of the KakaoTalk app (`com.kakao.talk`), a messenger service, followed by exiting the chatting room to navigate to the friends' list screen. Based on the three naming rules and considering a syntactic structure, this sequence can be labeled as *chatting task*, *messenger chatting task*, or *KakaoTalk chatting task* in Table 3.

Table 3. The naming rule examples of smartphone human interaction routine behavior task

Naming Rules	**Syntactic Structures**	**Task Naming (KT.CRHA→KT.MA)**
General behavior-based task	Behavior (gerund) + Task (noun)	Chatting Task
Service-specific behavior-based task	Service (noun) + Behavior (gerund) + Task (noun)	Messenger Chatting Task
App-specific behavior-based task	App name (noun) + Behavior (gerund) + Task (noun)	Kakaotalk Chatting Task

As shown in Appendix B.1, a general behavior-based task naming rule (e.g., *chatting task*) received high scores in consistency, universality, generality, comprehensibility, and comprehensiveness. This naming rule was concise and encompasses all activities without being tied to specific platforms or apps. However, this rule scored relatively low in specificity, distinctiveness, and accuracy (average total score: 32.7) because this rule did not indicate the type of service in which the chatting occurs. In contrast, an app-specific behavior-based task naming rule (e.g., *KakaoTalk chatting task*) received high scores in specificity, accuracy, and distinctiveness as this rule provided clear information about the app and highlighted app-specific differences. However, this rule received lower scores in consistency, comprehensibility, generality, and universality (average total score: 29.3), as it was less informative to users unfamiliar with the app and required that the app name be changed or replaced by another. A service-specific behavior-based task naming rule (e.g., *messenger chatting task*) had scores similar to a general behavior-based task naming rule in most criteria but achieved the highest scores in specificity, distinctiveness, and accuracy (average total score: 33.7). This was because it allowed user behavior to be contextualized based on the characteristics of the app service and enabled more detailed analysis of user actions. Given these findings, the service-specific naming rule was selected for SHIRBT modeling.

3.2.3 ***Establishing SHIRBT Categorization and Labeling Criteria for LLM Training.*** Additionally, we defined SHIRBT clustering labeling criteria to account for cases where sequential patterns are composed of within-app UI state events with the same package and class names but differ in order or include repetitions. According to case 1 in Figure 6, the sequential pattern ⟨KT.MA → KT.CRHA⟩ represents the behavior task of navigating from the main screen (i.e., friend list screen) to the chatting screen in the KakaoTalk app, which can be labeled as the *messenger browsing chat lists task*. On the other hand, ⟨KT.CHRA → KT.MA⟩ represents the behavior task of exiting from the chatting screen to the main screen, which is labeled as the *messenger chatting task*. As such, even when sequential patterns are composed of the same events, the performed task can differ depending on the order of events. To reflect this distinction, we defined criteria to differentiate task names based on the order of events, as shown in Figure 6.

In contrast, sequential patterns where the same event repeats at least once, such as ⟨KT.MA → KT.CRHA → KT.MA⟩, can be interpreted as a combination of the *messenger browsing chat lists task* and *messenger chatting task*, which may be labeled as the *messenger browsing chat list & chatting task*. However, since the *messenger chatting task* encompasses the meaning of the *messenger browsing chat lists task* in this case, it can be merged into the broader *messenger chatting task*. Accordingly, we defined task labeling criteria that consolidate tasks into a single representative name when one task meaningfully encompasses the others within a sequential pattern. On the other hand, as shown in case 2 in Figure 6, sequential patterns like ⟨GAG.MAG → GAG.CAG → GAG.MGA⟩ can be labeled as *email reading task* and *email composing task*. However, one task cannot be interpreted as encompassing the other in terms of behavioral semantics. In such cases, instead of selecting only one task, we defined task labeling criteria to represent them as a combined label, such as *email reading & composing task*.

Furthermore, even when the package and class names of the events in a sequential pattern are not identical, consistent task naming can be achieved if the app service domain is the same and the functional meaning of the class names aligns. For instance, as illustrated in case 3 of Figure 6, apps like Google, Naver, Microsoft Bing, and Daum differ in package names but share a common service domain as search engines and internet portals. Although their class names vary slightly (e.g., main activity, search activity, and browser activity), they can be consistently labeled as *web search task* or *web browsing task*. Additionally, since *web search & browsing task* can be broadly categorized under *web browsing task*, all such patterns are ultimately labeled as *web browsing task*. Therefore, we established task labeling criteria to consistently represent tasks when the functional meaning of the class names and the app service domain are aligned, even if package and class names differ.

3.2.4 ***Few-shot Prompting and Chain-of-Thought (CoT) Reasoning for LLM Training.*** Based on the previously defined task naming rules and labeling criteria, sequential patterns can be effectively modeled into SHIRBTs that are understandable for both users and service providers. However, in real-world scenarios, it is impractical to model all sequential patterns into SHIRBTs after model deployment manually. Therefore, we developed a system that automates the modeling of sequential patterns into SHIRBTs through prompt engineering by training an LLM on the task naming rules and labeling criteria, along with the examples previously used, as detailed in Appendix B.2. Since SHIRBT modeling requires sufficient examples and contextual understanding to correctly interpret the rules, we employed GPT-4o [75], which is known for its strong reasoning capabilities.

To further improve accuracy, we integrated *few-shot prompting* and *Chain-of-Thought (CoT) reasoning*. As described in "4. Few-shot Prompting and Chain-of-Thought Reasoning in Applying Naming Rules and Labeling Criteria" in Appendix B.2, we provided representative sequence patterns (e.g., ⟨KT.CHRA → KT.MA⟩) along with ground truth SHIRBT labels (e.g., *messenger chatting task*) that were defined by three domain experts in accordance with the established rules and criteria. Each example also included information about which labeling

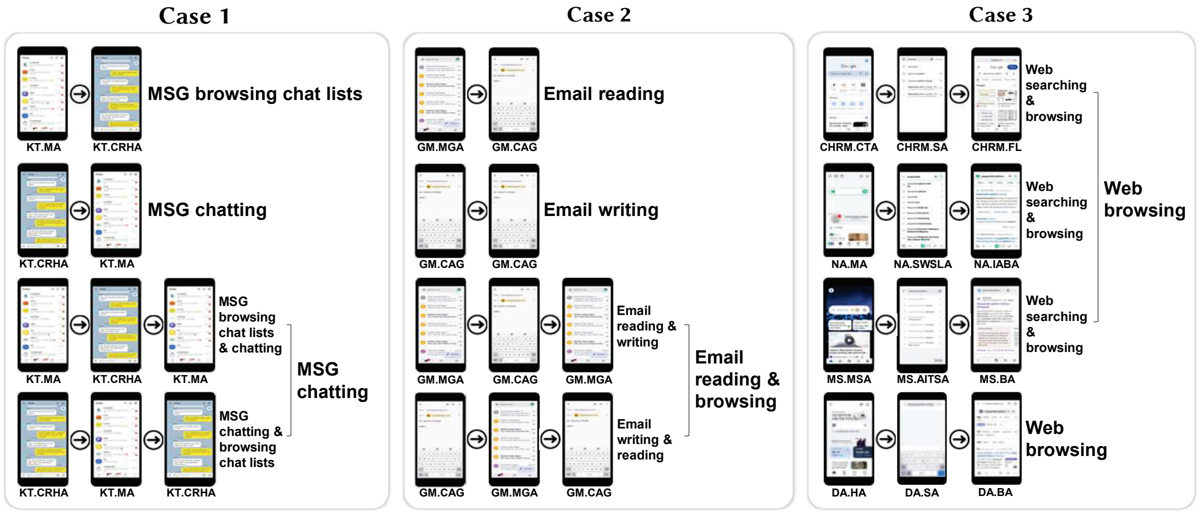


Fig. 6. Smartphone human interaction-based routine behavior task labeling criteria (case 1: task naming based on event order, case 2: merging tasks when one encompasses other tasks, case 3: unified task naming across apps based on equivalent functionality)

Abbreviations:
KT.MA = `com.kakao.talk.activity.main.MainActivity`,
KT.CRHA = `com.kakao.talk.activity.chatroom.ChatRoomHolderActivity`,
CHRM.CTA = `org.chromium.chrome.browser.ChromeTabbedActivity`,
CHRM.SA = `org.chromium.chrome.browser.searchwidget.SearchActivity`,
CHRM.FL = `com.android.chrome.android.widget.FrameLayout`,
NA.MA = `com.nhn.android.search.proto.MainActivity`,
NA.SWSLA = `com.nhn.android.search.browser.control.searchwindow.suggest.SearchWindowSuggestListActivity`,
NA.IABA = `com.nhn.android.search.browser.InAppBrowserActivity`
MS.MSA = `com.microsoft.sapphire.app.main.MainSapphireActivity`,
MS.AITSA = `com.microsoft.sapphire.app.search.autosuggest.activity.AIToolsSuggestActivity`,
MS.BA = `com.microsoft.bing.com.microsoft.sapphire.app.browser.BrowserActivity`,
DA.MA = `net.daum.android.daum.ui.main.MainActivity`,
DA.SA = `net.daum.android.daum.features.entrypage.SearchActivity`,
DA.BA = `net.daum.android.daum.browser.BrowserActivity`,
GM.MAG = `com.google.android.gm.ui.MailActivityGmail`,
GM.CAG = `com.google.android.gm.ui.ComposeActivityGmail`

criterion it corresponds to, as well as a CoT reasoning explanation (e.g., "Moving from the chatroom to the main activity implies exiting a chat session"). This structure allows the model to learn not only surface-level name mapping but also rule-based reasoning for task inference, thereby improving the interpretability and generalizability of the automated SHIRBT modeling system.

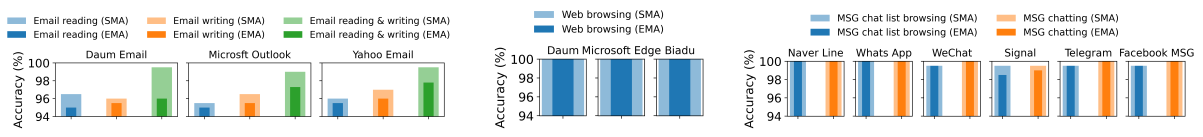
3.2.5 *Evaluating the Performance of the LLM-Based SHIRBT Modeling System*.

To comprehensively and quantitatively evaluate the actual performance, reliability, robustness, and generalizability of the LLM-based SHIRBT modeling system, we assessed whether the system maintains consistent performance across different applications and within-app UI state events for the same SHIRBT tasks (e.g., chatting, chat list browsing, web browsing, email reading, email writing, and email reading & writing). Additionally, we conducted an ablation study to quantitatively analyze the impact of few-shot prompting and Chain of Thought (CoT) reasoning on the system's accuracy.

To evaluate the accuracy of the experimental results, we used Exact Match Accuracy (EMA) and Semantic Match Accuracy (SMA). EMA measures whether the SHIRBT name inferred by the LLM exactly matches the ground truth task name at the string level. In contrast, SMA considers cases where the inferred SHIRBT name satisfies the naming rules and task labeling criteria, and conveys the same meaning despite differences in expression (e.g., "Messaging chatting task" vs. "Messenger chatting task") as semantically equivalent to the ground truth. SMA is calculated based on the Semantic Textual Similarity (STS) between the LLM-generated SHIRBT name

Table 4. Few-shot prompting apps and test dataset apps used to evaluate the consistency of LLM-based task labeling across diverse apps

SHIRBT Tasks	Few-shot Apps	Test Dataset Apps
Email reading, Email writing, Email reading & writing	Gmail	Daum Mail, Yahoo Email, Microsoft Outlook
Web browsing	Naver, Chrome	Daum, Microsoft Edge, Baidu
MSG chatting , MSG chat list browsing	KakaoTalk	WhatsApp, WeChat, Telegram, Naver Line, Signal, Facebook



(a) Email Apps (b) Browser Apps (c) Messenger Apps

Fig. 7. LLM's task clustering and labeling performance evaluation for each apps measured by semantic match accuracy (SMA) and exact match accuracy (EMA)

and the ground truth task name. To ensure precise STS measurement, we used OpenAI's text-embedding-3-large[76] model to generate embeddings for each SHIRBT name, and computed the cosine similarity between these embeddings[81, 98]. For accurate evaluation, the LLM was independently run 200 times per SHIRBT task for each app, and the inference accuracy was reported as a percentage (e.g., 190 correct out of 200 = 95%), as shown in Figure 7 and Table 5.

As shown in Table 4, the test dataset was constructed using within-app UI state events from apps involved in few-shot prompting and CoT reasoning, as well as from other apps performing functionally equivalent tasks across three service types, such as email, web browser, and messenger (see Appendix B.3's Table 15). As shown in Figure 7, both EMA and SMA achieved high accuracy rates exceeding 95% (e.g., 190 correct out of 200 trials) across all service types. Furthermore, the difference between EMA and SMA remained relatively small, ranging from 0% to 3.5%. These results demonstrate that the LLM-based SHIRBT modeling system consistently delivers strong performance in both naming and string-level accuracy across diverse sequence patterns of within-app UI state events for apps performing the same tasks. The small gap between EMA and SMA indicates that the system produces semantically consistent labeling results even when the inferred SHIRBT names differ in expression, which supports the high reliability and robustness of the model. Detailed SHIRBT name inference results for each sequence pattern in the test dataset, along with their corresponding EMA and SMA scores, are provided in Appendix B.3's Table 15.

Furthermore, to quantitatively analyze the impact of including few-shot prompting and Chain of Thought (CoT) reasoning on SHIRBT-specific accuracy, we designed and conducted an ablation study. In addition to the full prompt, which includes both few-shot prompting and CoT reasoning as described in "3. Few-shot Prompting and Chain of Thought Reasoning in Applying Naming Rules and Labeling Criteria" of Appendix B.2, we also evaluated two reduced variants, a few-shot only prompt that excludes CoT reasoning, and a rule only prompt that excludes both few-shot examples and CoT reasoning. According to the ablation results presented in Table 5, which report the average EMA and SMA accuracy (in percentages) across test dataset apps for each task, the full prompt consistently achieved the highest accuracy for both metrics. The few-shot only prompt showed only a slight decrease of 2.3% in both EMA and SMA for the web browsing task compared to the full prompt. However, a substantial performance drop of 20.4% in EMA was observed for the email reading task. The rule only prompt, which provides no examples for the LLM to learn from, produced significantly lower performance,

Table 5. Ablation study of the LLM's task clustering and labeling performance measured by exact match accuracy (EMA) and semantic match accuracy (SMA)

Tasks	**Apps**		**Full Prompt**		**Few-shot Only**		**Rule Only**	
	Few-shot	**Test Dataset**	**EMA**	**SMA**	**EMA**	**SMA**	**EMA**	**SMA**
Email reading	Gmail	Daum Mail, Microsoft Outlook, Yahoo Email	95.2±0.3	96.0±0.5	74.8±1.3	87.8±2.5	0.0±0.0	0.2±0.3
Email writing			95.7±0.3	96.5±0.5	81.5±3.1	89.2±0.8	0.0±0.0	1.2±0.3
Email reading & writing			97.0±2.2	99.3±0.3	88.1±4.3	94.9±2.5	0.0±0.0	0.3±0.6
Web browsing	Chrome, Naver	Daum, Microsoft Edge, Baidu	100.0±0.0	100.0±0.0	97.7±0.3	97.7±0.3	33.3±3.2	45.5±3.5
MSG chat list browsing	KakaoTalk	Naver Line, Whats App, WeChat, Signal, Telegram, Facebook MSG	99.5±0.5	99.7±0.3	84.8±3.1	86.8±1.9	0.0±0.0	0.2±0.4
MSG chatting			99.8±0.4	99.9±0.2	93.1±5.4	94.8±4.2	0.0±0.0	1.9±1.3

EMA = Exact Match Accuracy, SMA = Semantic Match Accuracy, Few-shot Only = prompt without Chaint of Thoughts (CoT) but including few-shot prompting, Rule Only = prompt excluding few-shot train dataset and CoT

with EMA ranging from 0 to 33% and SMA from 0.2 to 45.5%. These findings confirm that removing either few-shot prompting or CoT reasoning leads to a notable decline in SHIRBT modeling performance. Moreover, they quantitatively demonstrate that both few-shot prompting and CoT reasoning play important roles in improving the generalization performance of the SHIRBT modeling system.

3.3 SHIRBT and Baseline Feature Extraction

We implemented the feature extraction pipeline to extract baseline features (DLV 1–3: general smartphone usage and categorical app usage) and SHIRBT features (DLV 4–5: within app usage trajectories) for comparing the performance of the stress detection model. These features are based on three metrics (i.e., usage duration, usage frequency, and typing behavior) to quantify human behavior, deriving insights from previous research on smartphone interaction-based digital phenotyping [34, 47, 51, 68, 83, 94] and prior stress monitoring research (Table 1). The extracted SHIRBT features include SHIRBT completion time/frequency-related features and SHIRBT generic/calculated typing features, as shown in Table 6. Baseline features (DLV 1–3) were derived from three quantifiable metrics (i.e., usage duration/frequency, and typing behavior) as well, using prior stress detection model studies in Table 1. In particular, general typing-related features are obtained by categorizing backspace, characters, and keystroke typing, then counting their daily frequency or duration time for each SHIRBT, as detailed in Table 6. Calculated typing features offer complex user typing behavior metrics to measure details such as error rate, characters/keystrokes per time, and time per character/keystroke. While not used in existing stress detection models, calculated typing features have been validated in studies on mobile typing measures [62] and have proven useful in smartphone-based substance use disorder detection research (e.g., alcohol detection) [51]. Each DLV 1 to 5 stage measured these quantifiable metrics with five statistical measures (average, median, min, max, sum) within daily time windows, as detailed processes are described in Appendix C.

4 DATA COLLECTION AND SHIRBT FEATURE EXTRACTION

4.1 Stress Label Data Collection

To evaluate the performance of the proposed SHIRBT-based stress model against baseline features from previous smartphone interaction studies (Table 1), we conducted an in-the-wild data collection with stress labels. As shown in Table 7, we recruited 40 young adults (23 males and 17 females) in their 20s and 30s, including university students and workers. Data were collected over two periods: a short-term (about 1 month) and a long-term (about 4–5 months) phase. This design allowed us to examine whether the influence of SHIRBT features on stress varies by the duration of daily behavior. The number of participants and the study duration were comparable to or greater than previous personal stress modeling research (Table 1), ensuring sufficient demographic and behavioral diversity. Two types of stress measurements were collected: (1) a one-time pre-survey using the

Table 6. Types of features in depth level of SHIRB

DLV of SUB	Metrics	Features
DLV 1: Overall Smartphone Usage Behavior	Time	• Smartphone usage duration time (SUD) • Screen unlocking time (UKT)
	Frequency	• Smartphone usage frequency (SUF) • Screen unlocking frequency (UKF)
	Typing	• None
DLV 2: Overall App Usage Behavior	Time	• App usage duration time (AUD) • App noti response time (NRTA)
	Frequency	• Overall app usage frequency (AUF)
	Typing	• Generic typing: NTB, NTC, NTK, TDT • Calculated typing: CPS, KPS, SPC, SPK, KSPC
DLV 3: Categorical App Usage Behavior	Time	• App usage duration time (AUD) • App noti response time (NRTA)
	Frequency	• App usage frequency (AUF)
	Typing	• Generic typing: NTB, NTC, NTK, TDT • Calculated typing: CPS, KPS, SPC, SPK, KSPC
DLV 4-5: Individual App and Within App Usage Behavior (Data Type Used in SHIRBT)	Time	• Task completion time (TCT) • View transition time (VTT) • App noti response time (NRTA)
	Frequency	• Task Frequency (TF) • View transition frequency (VTF)
	Typing	• Generic typing: NTB, NTC, NTK, TDT • Calculated typing: CPS, KPS, SPC, SPK, KSPC

– DLV of SUB: Depth Level of Smartphone Interaction Usage Behavior
– Generic typing: Number of typed backspaces (NTB), Number of typed characters (NTC), Number of typed keystrokes (NTK), typing duration time (TDT)
– Calculated typing: Characters per second (CPS), keystrokes per second (KPS), seconds per character (SPC), seconds per keystroke (SPK), keystrokes per character (KSPC)

10-item Perceived Stress Scale (PSS-10) [20] to assess baseline stress, and (2) repeated daily stress surveys using Experience Sampling Method (ESM) triggers during the study. The mean PSS score was 21.15±6.70 out of 40 (0–13: low, 14–26: moderate, 27–40: high), indicating that most participants experienced moderate to high stress [20]. Two participants (P35, P39) with very low stress scores were excluded during pre-screening.

Participants self-reported their daily stress through smartphone and smart-speaker ESM triggers [55]. We adopted a micro-ESM design consisting of a single 5-point Likert item ("How was your overall stress level today?") to minimize burden [48, 66]. Daily stress ratings were binarized following Zhang et al. [103]: scores of 4–5 and 3 were labeled as "stress-positive," while 1–2 were "non-stress." This binarization method is commonly used in long-term monitoring [45, 97, 103] to balance sensitivity and user compliance. Twelve of the remaining 38 participants were excluded: eight due to withdrawal or insufficient ESM data (≤10 valid days) and four due to extreme label imbalance (≤1% positive samples). Such imbalance would require excessive augmentation (e.g., SMOTE) and risk model distortion. The final dataset included 26 participants, with 43.8±20.7% stress-positive and 56.2±20.7% non-stress samples (Table 7). Only days containing both valid stress labels and smartphone logs were analyzed. The short-term group (n=16) provided an average of 30.5 days (range: 25–36), and the long-term group (n=10) provided 105 days (range: 53–148). All procedures were approved by the Institutional Review Board (IRB).

4.2 Smartphone Data Collection

To extract SHIRBT and baseline features, we collected smartphone interaction data using a custom logger based on Android APIs [22–24, 26] during the same period with stress label collection in real life. The data include three types (Figure 1): (1) DLV 1: smartphone usage behavior (e.g., screen on/off, unlock), (2) DLV 2–4: app usage behavior (e.g., categories, total/individual app usage), and (3) DLV 5: within app usage behavior (within-app UI state, including active events such as *WindowStateChanged* event, *ViewTextChanged* event, and passive events such as *NotificationStateChanged* [21]). As a result, we collected an average of 132.35 app types (range: 69–340), an average of 975.12 *WindowStateChanged* event types (range: 501–2,974), an average of 55.73 *ViewTextChanged* event types (range: 23–156), and an average of 75.81 *NotificationStateChanged* event types (range: 48–145) per participant. To address privacy concerns, sensitive personal data was hashed and stored an encrypted. For example, app usage logs did not record specific characters typed, only whether typing occurred and the type of input (e.g., character, backspace). Message or call sender/receiver information was also hashed for anonymity. Sensitive information was excluded or encrypted, and the data collection methods were shared with participants, ensuring transparency and obtaining consent under IRB approval.

4.3 SHIRBT Modeling and Feature Extraction Result

We utilized smartphone interaction data collected in the wild to derive individual SHIRBTs using the SHIRBT-MMF framework (Section 3) and extracted both SHIRBT features and baseline features used in previous studies. First, to compare SHIRBT features with traditional categorized app usage behavior (DLV 3 in Figure 2), we classified the 132.35 apps that participants used daily on average into categories. Following the Google Play Store's categorization based on primary app functionality, we defined 42 app categories. However, since participants performed an average of only seven SHIRBTs daily (range: 2–10) during the study, we merged the app categories into 11 broader categories (Table 8) to ensure a fair comparison between the SHIRBT-based stress model and the baseline model. As a result, participants engaged with an average of 6.5 app categories daily, corresponding to an average of 145.3 apps per participant (Tables 10 and 11). Across all participants, 29 unique SHIRBTs were identified through the SHIRBT-MMF framework, with individuals performing an average of seven tasks per day (range: 2–10) (Table 9). Notably, the number of apps associated with these seven SHIRBT tasks averaged only 4.40 per participant, amounting to just 3–6% of the apps used in the categorized app features (Table A7 and A8 in Supplement: D).

As shown in Table 9, SHIRBTs offer a more precise, consistent, and interpretable representation of smartphone interaction behaviors compared to app categories (DLV 3) or individual apps (DLV 4). Unlike app-dependent naming conventions, SHIRBTs are based on user behavior tasks, ensuring feature stability even when users switch or adopt new apps. This characteristic helps mitigate covariate shift and schema drift. To further evaluate the performance advantages of SHIRBT-based stress models over baseline models, we extracted a greater number of features (averaging 195.3 vs. 151.8) (Tables 10 and 11). While feature count is adjustable, an excessive number of features can negatively impact model performance. Therefore, we deliberately generated more SHIRBT features than baseline features to test the model under less favorable conditions (Table A9 in Supplement: D). Conversely,

Table 7. Demographic information and stress label collection

Periods	**Demographic Information**				**Pre-survey**	**Daily-survey (ESM)**			
	Total N	Selected N	Avg. Age	Sex	Avg. PSS Score (1–40)	Avg. Samples (day)	Avg. Stress Level Score (1–5)	Avg. Proportion Pos (%)	Avg. Proportion Neg (%)
Short-term	20	16	27.4±4.2	M: 6, F: 10	23.4±4.7	30.5±3.3	2.8±0.8	45.9±20.4	54.1±20.4
Long-term	20	10	24.6±2.3	M: 9, F: 1	17.6±5.9	105.1±29.5	2.5±1.1	40.5±21.7	59.5±21.7

Table 8. Used app categories for each participant

App Categories (DLV 3)	Short-Term Periods (16 participants)	Long-Term Periods (10 participants)
SOC & COMM	All participants	All participants
FIN	All participants	All participants
MM	15 participants (except P3)	All participants
TL & SYS	All participants	All participants
ENT	P2, P14, P18	P23, P24, P25, P30, P32, P37
HLTH & BEA	P16, P19	P24, P28
SHOP & LIFEST	All participants	P24, P27, P29, P30, P32, P37
LOC & TRANS	P9, P10, P11, P15, P18	P24, P27, P28, P29, P30
WRK & PROD	P1, P2, P5, P6, P9, P11, P14, P15, P16, P19	8 participants (except P23, P40)
BKS & EDU	P16, P19	None

Note: SOC & COMM = Social & Communication; FIN = Finance; MM = Multimedia; TL & SYS = Tools & System; ENT = Entertainment; HLTH & BEA = Health & Beauty; SHOP & LIFEST = Shopping & Lifestyle; LOC & TRANS = Location & Transportation; WRK & PROD = Work & Productivity; BKS & EDU = Books & Education. Values indicate the participants who used apps in each category during the short-term and long-term logging periods.

to ensure that the superior performance was not merely due to the number of features, we also evaluated models using only universally performed SHIRBTs, such as *messenger chatting* and *finance paying*, resulting in fewer features than the baseline. Additionally, by leveraging these commonly performed SHIRBTs, we validated personalized SHIRBT feature-based models and a global model to assess their effectiveness across participants.

5 MODEL EXPERIMENT SETUP

5.1 Comparison Methods

We evaluated the stress detection model's performance using SHIRBT features compared with baseline features in four conditions:

- **Section 6.1. Personal SHIRBT vs. Baseline**: How do personalized SHIRBT features compare to baseline features in model performance?
- **Section 6.2. Personal One/two SHIRBT vs. SHIRBTs vs. Baseline**: How do the one or two SHIRBT features most frequently and consistently used by each individual compare to baseline and the full set of SHIRBT features? We aimed to verify that using only one or two SHIRBT features could achieve comparable performance, even with fewer data and feature types than baseline features.
- **Section 6.3. Feasibility of Global Model** Can SHIRBT features be applied to global models as well as personal models? If they can, how does their performance differ from baseline features?
- **Section 6.4. Feature Importance in Personal Model** Which feature types have the greater impact on explaining the model among SHIRBT and Baseline?

5.2 Model Building

5.2.1 ***Ensemble ML Model***. We built a binary stress classifier to validate the superiority of the proposed SHIRBT features compared to existing baseline features in Table 6. While the model performance is crucial in most prior mental health models, interpretability is emphasized, leading to a preference for traditional machine learning models over deep learning models. Therefore, to compare and validate against existing research, we utilized ensemble models such as Random Forest (RF) [14], Gradient Boosting Machine (GBM) [32], eXtreme GBM (XGBM) [17], and Light GBM (LGBM) [46], which have shown superior performance in prior mental health models [13, 47, 51, 94] as shown in Figure 2. Ensemble models can reduce bias and variance, thereby preventing

Table 9. Used SHIRBTs for each participant

App Categories (DLV 3)	Types of Apps (DLV 4)	Types of SHIRBT (DLV 5)	Short-Term Periods (16 participants)	Long-Term Periods (10 participants)
SOC & COMM	KT	MSG chat list browsing	15 participants (except P14)	9 participants (except P24)
	KT	MSG chatting	All participants	8 participants (except P24, P40)
	KT	MSG profile viewing	P9, P10, P11, P15, P18	None
	KT	MSG media viewing	None	P30
	KT, SBI	MSG unlocking	P19	P25, P37
	GCH, SIB, NA, GQS, FAB	Web browsing	12 participants (except P1, P2, P6, P17, P19)	P24, P27, P28, P29, P37
	IN, TW	SNS browsing	P9, P10, P11, P15, P16, P18, P19, P20	P28
	SM	SMS messaging	P3, P12, P17, P20	None
	SIC, SPD, WW	Call managing	P5, P17, P19	None
	ET	Community browsing	P14	P37
	BD, VB	Dating board browsing	P3	None
	BD	Dating profile viewing	P3	None
	GM	Email reading	P6	None
FIN	SP	Finance paying	15 participants (except P16)	9 participants (except P40)
	SC, TO	Finance managing	None	P24
MM	YT	Video browsing	P2, P10, P12, P14, P16, P18	P23, P25, P28, P30
	YT	Video watching	P2, P10, P12, P14, P16, P18	None
	CA	Camera shooting	P12	None
TL & SYS	GF	Device searching	P3, P15	P27, P30
	NS	System noti checking	P18	P32, P37
	SCP	System controlling	P12	None
	ZP	Document reading	P14	None
	CL	Clock time checking	None	P24
ENT	NW	Webtoon browsing	None	P32
	KP	Entertainments browsing	None	P23
HLTH & BEA	CW, CHA	Health tracking	P19	P24, P28
SHOP & LIFEST	DA	Shoping	None	P29
WRK & PROD	TS, SCA, GCA	Time scheduling	P2, P9, P16	None
BKS & EDU	SID	ID checking	P6	None
Overall Categories	**Overall apps**	**Overall tasks**	All participants	All participants

Note: KakaoTalk (KT), Samsung Biometrics (SBI), Google Chrome (GCH), Samsung Internet Browser (SIB), Naver (NA), Google Quick Searchbox (GQS), Free AD Blockerbrowser (FAB), Samsung Messages (SM), Samsung In Call (SIC), Samsung Phone Dialer (SPD), Who Who (WW), Every Time (ET), Blind Date (BD), Vanilla Bridge (VB), Gmail (GM), Instagram (IN), Twitter (TW), Samsung Card (SC), Toss (TO), YouTube (YT), Camera (CA), Galaxy Finder (GF), NotiStar (NS), Samsung Control Panel (SCP), Zipper Plus (ZP), Clock (CL), Naver Webtoon (NW), Kakao Page (KP), Cash Walk (CW), Challengers (CHA), Danggeun (DA), Time Spread (TS), Samsung Calender (SCA), Google Calnendar (GCA), Stuent ID card (SID)

over/underfitting, and outperform single classifier models (e.g., logistic regression, decision tree) [17, 32, 46]. In the overall process, we derived results for both short-term and long-term models and adopted macro accuracy and AUC-ROC, which were used with prior most stress model studies in Table 1 as ML performance metrics to compare model performance.

5.2.2 ***TSNCV based Bayesian HPO and Feature Selection.*** To predict future data from past data in our daily-sampled time-series dataset, we used time series nested cross-validation (TSNCV) as shown in Figure 2. Most personal or global digital phenotyping-based mental health (e.g., stress) models used K-fold or leave-one-subject-out (LOSO) cross-validation to address imbalanced label datasets and improve performance [6, 19, 64, 94]. However, these methods mix past and future data, leading to learning past data from future data. Our study arranges training data chronologically to ensure learning occurs only from past data. We prevent complete imbalance (e.g., 0% or 100% positive samples) in train and test folds by setting a minimum sample size. Considering the amount of short-term (average of 30.5 days) and long-term (average of 105.1 days) samples, Each test fold has at least three days of samples for short-term personal models and seven days (one week) samples for long-term personal models in both outer and inner loops to maintain balance. The first training fold of the outer loop was

Table 10. Performance results of personal and global short-term stress detection models in SHIRBT vs. baseline

Average of 16 Personal Models									Global Model			
ML Methods	**All SHIRBTs**		**Two SHIRBTs (Chatting & Paying)**		**One SHIRBT (Chatting)**		**Baseline**		**Two SHIRBT (Chatting & Paying)**		**Two Baseline (SOC & FIN)**	
# of Apps	5.3±1.6		2		1		89.4±18.7		2		119	
# of Features	209.4±52.2		60.3±14.7		49.8±14.3		150.3±7.4		67		69	
ML Metrics	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
RF	72.4±10.5	63.0±10.6	66.2±14.2	57.1±7.9	68.3±13.6	60.0±10.0	67.8±11.6	57.0±8.5	50.1±7.8	48.3±6.9	**48.8±10.2**	**50.1±8.5**
GBM	**75.9±8.7**	**66.8±10.1**	**73.2±8.8**	**63.4±9.6**	**73.5±13.9**	**64.3±10.3**	**70.4±11.5**	**62.5±8.5**	**54.2±8.6**	**52.4±8.4**	47.5±7.6	46.7±9.4
XGBM	68.3±11.3	56.7±12.8	61.2±15.2	51.7±3.8	61.7±15.7	54.2±12.9	64.2±13.9	52.9±9.6	52.7±9.3	51.6±9.7	45.0±7.7	46.1±7.3
LGBM	56.6±13.8	52.6±5.8	58.8±16.0	51.0±3.3	58.3±13.3	51.8±4.3	56.6±13.8	50.0±0.0	53.8±10.3	52.1±9.4	46.0±9.8	46.4±9.5

Table 11. Performance results of personal and global long-term stress detection models in SHIRBT vs. baseline

Average of 10 Personal Models									Global Model			
ML Methods	**All SHIRBTs**		**Two SHIRBTs (Chatting & Paying)**		**One SHIRBT (Chatting)**		**Baseline**		**Two SHIRBTs (Chatting & Paying)**		**Two Baseline (SOC & FIN)**	
# of App	4.4±1.7		2		1		145.3±72.0		2		117	
# of Features	182.6±57.2		66.4±1.7		55.4±1.7		154.2±10.5		67		69	
ML Metrics	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
RF	69.7±6.4	60.8±9.8	67.2±9.0	62.2±8.1	66.0±6.5	60.2±9.8	68.1±8.2	59.3±9.6	**59.1±7.0**	**51.7±4.4**	54.9±11.4	49.1±4.0
GBM	**73.8±5.0**	**70.0±8.1**	**73.8±7.5**	**69.0±10.0**	**72.1±6.3**	**67.5±7.5**	**70.3±6.7**	**65.1±8.4**	53.1±8.5	48.4±3.2	53.3±10.6	48.6±3.2
XGBM	67.1±7.0	57.9±4.9	61.7±12.7	56.8±5.6	61.3±11.7	56.4±4.3	60.6±9.7	56.5±5.1	53.1±9.3	47.4±5.1	51.6±10.3	49.4±3.7
LGBM	60.4±6.5	53.3±4.7	55.8±6.6	51.1±3.2	56.2±6.5	52.8±2.9	56.6±5.3	52.1±4.1	50.7±10.2	45.3±3.8	**55.2±12.0**	**51.1±3.5**

set to a minimum of three days for short-term models and one week for long-term models, increasing the training dataset by three days (short-term models) and one week (long-term models) per fold, respectively. On the other hand, for the Global model, since it is difficult to follow the chronological order of the entire participants' data, we used LOSO cross-validation, where one participant is used as the training set, another participant as the validation set, and the remaining participants as the test set.

To enhance model performance, we implemented TSNCV, with 400 iterations run in the inner loop to determine the optimal hyperparameters and select the features for training in the outer loop in Figure 2. We used Bayesian hyperparameter optimization with a tree-structured Parzen estimator (TPE), a state-of-the-art (SOTA) algorithm. Unlike other black-box optimization methods (e.g., grid and random search), Bayesian-HPO creates a probabilistic model mapping hyperparameters to the objective's score probability function [11, 74]. Among Bayesian-HPO, TPE is faster at identifying HPO than other methods (e.g., Gaussian processes and random forest) [11]. We used the Optuna library, an advanced automatic HPO framework [5]. We used an ensemble model (RF) embedded method for feature selection, selecting features with importance above the median value in Figure 2.

5.2.3 ***Running Environment.*** The experiment was run on a 64-bit machine with Ubuntu 22.04 LTS, AMD Ryzen 96 5900X CPU, 64GB RAM, RTX3070Ti GPU, VRAM 8GB, Ubuntu 22.04 LTS operating system with installed Python version 3.11.9, NumPy version 1.26.4, Pandas version 2.2.2, and Scikit-learn version 1.5.0.

6 RESULTS

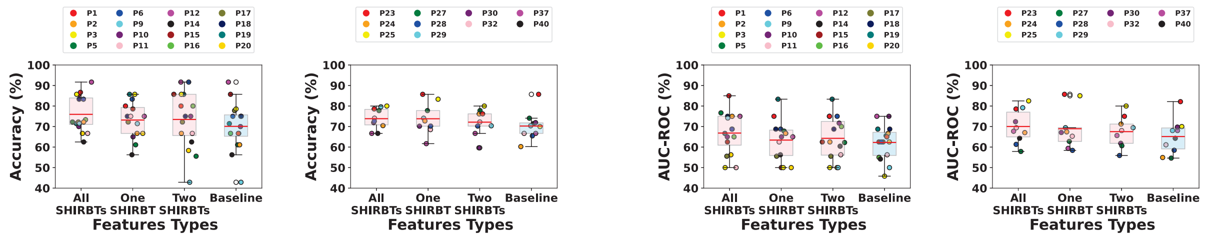
6.1 Personal Model: SHIRBT vs. Baseline

Among four ensemble models, the GBM model achieved the best results in all conditions (total SHIRBT vs. two SHIRBT vs. one SHIRBT vs. baseline). The average performance of personal SHIRBT models outperformed by 5.8% (short-term) and 3.9% (long-term) accuracy baseline models as shown in Table 10 and 11. The SHIRBT-based models also showed an average AUC-ROC improvement of 4.6% (short-term) and 5.4% (long-term) over the baseline models. These results demonstrate that SHIRBT consistently outperforms the baseline model, regardless of whether the data collection and SHIRBT modeling period was short or long. Additionally, these results verified that extracted SHIRBT features can achieve better performance even though the app types required is only an average of 5.0% (short-term: 3.2%, long-term: 6.1%) of that required for baseline features extraction across 26 participants as shown in Table 10, 11, and Table A7,A8 of Supplement: D. The performance results for each of the 26 personal stress models are presented in Figure 8.

6.2 Personal Model: One/Two SHIRBT(s) vs. All SHIRBTs vs. Baseline

The extracted SHIRBT features involve fewer app data volume than the baseline feature extraction process, but the number of features was kept superior to compare model performance under the same conditions in the total SHIRBT model in Section 6.1. However, as the feature count increases, users and service providers may find it hard to understand their importance and incur higher time and costs for feature engineering and ML system maintenance. Therefore, we controlled the number of features and validated whether extracted SHIRBT features still show superior performance for stress modeling with fewer features than baseline features. To achieve a small number of SHIRBT features, we developed the model using only the most frequently or extensively used SHIRBT feature sets among all participants: the messenger chatting task (chat) and the mobile paying task (pay) features. These one or two SHIRBTs-based extract input features account for between an average of 8.0% and 37.9% of the baseline input features in both short- and long-term personal stress models, in Table 10, 11 and Supplement: D's Table A9. The detailed performance results for each of the 26 personal stress models are presented in Figure 8.

The two SHIRBTs (paying and chatting tasks) features showed an **approximately average of 3% accuracy performance improvement** over the baseline model in both short and long-term personal models in Table 10, 11 and Figure 8. Moreover, single SHIRBT (chat) based features showed an even greater **average of 2% and 3% accuracy performance enhancement** over the baseline model in both short and long-term models, respectively. Additionally, one or two SHIRBT-based features showed a greater **average of 1%–4% AUC-ROC performance improved** than baseline features in both short and long-term models, respectively. This shows that stress models based on one or two frequently used SHIRBT features per participant can achieve similar or slightly higher performance than baseline models, even with significantly fewer app data types (between the average of 0.6% and 2.2% compared to baseline features' app data types per participant) and extracted number of features (between

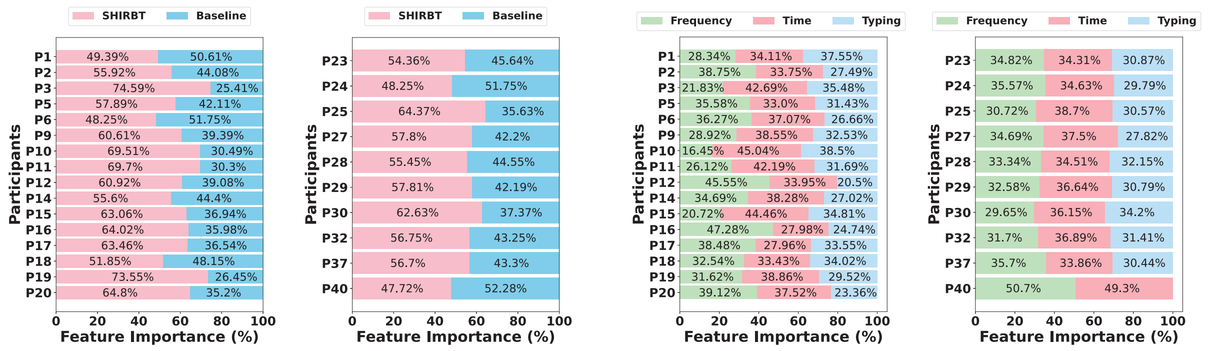


(a) Short-term (b) Long-term (c) Short-term (d) Long-term

(A) Accuracy results (B) AUC-ROC results

Fig. 8. Accuracy and AUC-ROC performance of personal models: 16 short-term personal models and 10 long-term personal models

]



(a) Short-term (b) Long-term (c) Short-term (d) Long-term

(A) SHIRBT vs. Baseline (B) Time vs. Frequency vs. Typing

Fig. 9. Comparison of feature importance: (A) comparison of selected SHIRBT vs. baseline features in best combined SHIRBT and baseline features based 16 short-term and 10 long-term personal models, (B) comparison of selected time vs. frequency vs. typing features in best SHIRBT features based 16 short-term and 10 long-term personal models

average of 34.1% and 41.7% compared to baseline features per participant). Therefore, utilizing one or two SHIRBT features reduces the privacy burden of app data collection for users.

6.3 Global Model: SHIRBT vs. Baseline

We developed a global stress model for both short and long-term data to evaluate whether the SHIRBT-based model outperformed the baseline model. Using the most frequently used SHIRBT (messenger chatting task) features (Table 9) among all participants, we built global stress models using demographic factors as the features (e.g., sex, age, jobs). For comparison, the baseline features were derived from apps categorized as Social & Communication (SOC & COMM) and Finance (FI), which correspond to matching *messenger chatting task* and *finance paying task*. The SHIRBT-based global model outperformed the 5.4% (short-term) and 3.9% (long-term) accuracy of the baseline model as shown in Table 10 and 11. Furthermore, the SHIRBT-based global model demonstrated **4% and 5% AUC-ROC higher** compared to the baseline model in both the short and long term. This indicated that SHIRBT can enhance interpretability and performance while reducing app data collection and feature extraction in the global model as well. Additionally, the frequent use of SHIRBT or app categories related to communication likely reflects fundamental human routine tasks essential for social activities in both the short and long term. However, the performance of the global model was significantly lower compared to the personal model. This is likely because the personal model more precisely captures an individual's unique smartphone usage patterns, daily routines, and stress responses, leading to superior performance compared to the global model.

6.4 Selected Feature Importance of SHIRBT and Baseline

The proposed SHIRBT features are performed daily without exception for at least a month to several months as part of an individual's routine activities and social interactions, resulting in low feature variability. Additionally, all daily tasks inherently have a significant impact on the model. Furthermore, the average number of SHIRBTs is relatively small (mean: 7, range: 2–10), and their importance increases with more frequent and prolonged use throughout the day. Therefore, it is unnecessary to separately determine which SHIRBTs are crucial for model performance using feature importance analysis techniques such as SHapley Additive exPlanations (SHAP) [87], Local Interpretable Model-agnostic Explanations (LIME) [82], or ensemble-based feature importance evaluation [96] [16, 18, 45]. However, the importance of time, frequency, and typing behavior-related features may vary within each SHIRBT. Moreover, while the proposed SHIRBT features outperform baseline features, it is necessary to verify their superiority by analyzing which specific features contribute most to improving model performance when SHIRBT and baseline features are used together.

To compare the selected features' impact on extracted SHIRBT and baseline features, we developed personal stress models based on both SHIRBT and baseline features. The best short and long-term combined personal models showed an average of 74.7% and 71.9% accuracy, respectively. The feature selection based on embedded feature importance using a median threshold in the ensemble model resulted in an average of 25.0% of features being selected (86.1±31.7) compared to the average input features per participant (351.74±57.57). Considering the difference in the number of input features between SHIRBT and the baseline, we calculated the ratio of selected features relative to the number of input features, rather than simply counting the number of selected features in the best model. This ratio was normalized to 100% for the total selected SHIRBT and baseline features for each fold, as shown in Figure 9a and 9b. For example, P40's stress model used 32 SHIRBTs and 137 baseline input features. In P40's stress model, 31.70% of SHIRBT features and 34.72% of baseline features were selected from the input features. After normalization, SHIRBT features accounted for 47.72% and baseline features for 52.28%. As shown in Figure 9a and 9b, this method was used to compare the importance of SHIRBT and baseline features across all 26 participants. As a result, in the top 26 models based on combined SHIRBT and baseline features, SHIRBT features were selected on average 19% more (SHIRBT: 59.5%, baseline: 40.5%) than baseline features.

Additionally, we analyzed which selected total SHIRBT feature types among time, frequency, and typing behavior metrics were most influential in 26 total SHIRBT-based personal models (see Section 6.1). We used the same feature importance calculation method from the process of comparing SHIRBT and baseline features in Figure 9a, 9b. Taking into account the difference in the number of input features among time, frequency, and typing-related features, we calculated the proportion of selected features relative to the total number of input features, instead of merely counting the selected features in the best model, and this ratio was normalized to 100% for the total selected SHIRBT's time, frequency, and typing features for each fold, separately. As shown in Figure 9c, 9d, **the feature types most frequently selected in proportion to the number of input features were time, frequency, and typing behavior, in order among most participants**. After normalizing the total number of selected feature types to 100%, the average selection rates in the short-term models were time (36.8%±5.2%), frequency (32.6%±8.7%), and typing (30.6%±5.4%), while in the long-term models, they were time (37.2%±4.4%), frequency (34.9%±5.9%), and typing (30.9%±1.6%).

7 Comparison of Covariate Shift and Schema Drift

We previously demonstrated through experiments that SHIRBT-based features outperformed baseline features in explaining and predicting stress levels. This performance superiority is not merely due to differences in model architecture or training methods, but it is also closely related to how robust the SHIRBT features are against covariate shift and schema drift (i.e., feature evolution). In this section, we go beyond model performance comparison and quantitatively compare the degree of input distributional change (covariate shift) and feature structure change (schema drift) between SHIRBT and baseline features. We aim to examine whether SHIRBT's improved robustness arises from its greater distributional stability and structural persistence. To this end, we utilized data from 10 long-term participants (P23–P40) whose data spans were sufficiently long to enable daily routine mining (average collection period: 105.1 days). Each participant's data was chronologically sorted and split evenly into a training interval (source distribution) and a test interval (target distribution). Using the

Table 12. Univariate and multivariate covariate shift evaluation metrics

Metric	Explanation	Formula	Symbol Definitions
Kolmogorov–Smirnov Test (KS Test)	Measures the maximum difference between two cumulative distribution functions to test distributional equality	p-value $= \frac{\text{\# of permutations with } D_n \geq d}{\binom{2n}{n}}$	- $F_n(x), G_n(x)$: Cumulative distribution functions of two different samples, each of size n - D_n: $\sup_x \lvert F_n(x) - G_n(x) \rvert$ - d: Observed value of D_n
Probability Stability Index (PSI)	Quantifies the shift in feature distributions by comparing binned frequencies.	$PSI = \sum_{i=1}^{n}(r_i - m_i) \cdot \log\left(\frac{r_i}{m_i}\right)$	- r_i: $\frac{\text{\# of reference data in i-th bin}}{\text{\# of data points}}$ - m_i: $\frac{\text{\# of monitored data in i-th bin}}{\text{\#of data points}}$
Jensen-Shannon Divergence (JS)	Measures the similarity between two probability distributions in a symmetric way.	$JS(P \parallel Q) = \frac{1}{2}D_{KL}(P \parallel M) + \frac{1}{2}D_{KL}(Q \parallel M)$	- P, Q: Two different probability distributions - M: $\frac{1}{2}(P+Q)$ - $D_{KL}(P \parallel Q)$: $\sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$
Maximum Mean Discrepancy (MMD)	Compares distributions by computing kernel-based differences in mean embeddings.	$\text{MMD}^2 = \mathbb{E}_{x,x'}[k(x,x')] + \mathbb{E}_{y,y'}[k(y,y')] - 2\,\mathbb{E}_{x,y}[k(x,y)]$	- $x, x' \sim P$: Two samples from train dataset - $y, y' \sim Q$: Two samples from test dataset - $k(x,y)$: Gaussian kernel function
Domain Classifier (DC)	Evaluates how well a classifier distinguishes source from target data domains.	$AUC = \int_0^1 TPR(FPR^{-1}(t))\,dt$	- TPR: $\frac{TP}{TP+FN}$ - FPR: $\frac{FP}{FP+TN}$
Data Reconstruction Error (DRE)	Measures the squared euclidean distance based reconstruction error after projecting and recovering input data.	Error $= \lVert X - \hat{X} \rVert_2^2, \quad \hat{X} = W_k W_k^T X$	- X: Input dataset of PCA - $\hat{X}$: Reconstructed data - $\lVert \cdot \rVert_2^2$: Squared L_2 distance (squared euclidean distance) - W_k: Top k principal components

SHIRBT-MMF framework, we extracted SHIRBT and baseline features from each interval and compared the robustness of the two feature types by measuring their respective degrees of covariate shift and schema drift.

7.1 Univariate & Multivariate Covariate Shift Comparison

As described in Section 2.2, covariate shift refers to changes in the distribution of input features between the training and deployment environments. Because such distributional changes can lead to performance degradation in deployed models, it is important to evaluate and anticipate them in advance. Covariate shift can be categorized into two types of data distribution changes [79]. First, univariate covariate shift refers to changes in the marginal distribution of individual features and is assessed through statistical hypothesis testing of each feature. Second, multivariate covariate shift captures shifts in the joint distribution across features—cases in which no single feature's distribution changes significantly, but their interdependence does. This type requires more complex metrics that consider feature relationships. To quantitatively measure both types of covariate shift, we employed a set of evaluation metrics validated by recent covariate shift detection studies [49, 60, 65, 79, 88], as shown in Table 12. For univariate covariate shift analysis, we used the two-sample Kolmogorov–Smirnov (KS) test, the Population Stability Index (PSI), and the Jensen–Shannon (JS) divergence. For multivariate covariate shift analysis, we used maximum mean discrepancy (MMD), domain classifier (DC), and data reconstruction error

Table 13. Definitions of abbreviations for SHIRBT and baseline feature subsets used for covariate shift comparison

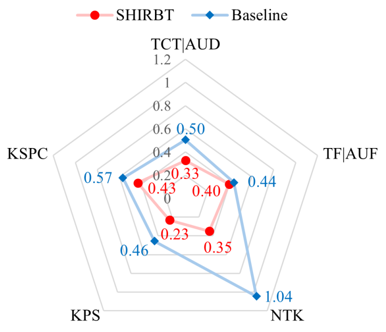
Shift Type	Abbreviation	Full Name
Univariate	TCT|AUD	Mean Statistic of TCT in One SHIRBT (Chatting) vs. AUD in One category (SOC & COMM)
	TF|AUF	Sum Statistic of TF in One SHIRBT (Chatting) vs. AUF in One category (SOC & COMM)
	NTK	Mean Statistic of NTK in One SHIRBT (Chatting) vs. NTK in One category (SOC & COMM)
	KPS	Mean Statistic of KPS in One SHIRBT (Chatting) vs. KPS in One category (SOC & COMM)
	KSPC	Mean Statistic of KSPC in One SHIRBT (Chatting) vs. KSPC in One category (SOC & COMM)
Multivariate	All	All SHIRBTs (DLV 4–5) vs. all baseline (DLV 1–3) related features
	One	One SHIRBT (Chatting) vs. one app category (SOC & COMM) related features
	Two	Two SHIRBTs (Chatting and Paying) vs. Two categories (SOC & COMM and FIN) related features
	One (TIM)	One SHIRBT (Chatting) vs. One category (SOC & COMM) related time features
	One (FRQ)	One SHIRBT (Chatting) vs. One category (SOC & COMM) related frequency features
	One (TYP)	One SHIRBT (Chatting) vs. One category (SOC & COMM) related typing features

– Features: Task completion time (TCT), App usage duration time (AUD), Task frequency (TF), App usage frequency (AUF), Number of typed keystrokes (NTK), 'KPS, KSPC
– Metrics: Time (TIM), Frequency (FRQ), Typing (TYP)

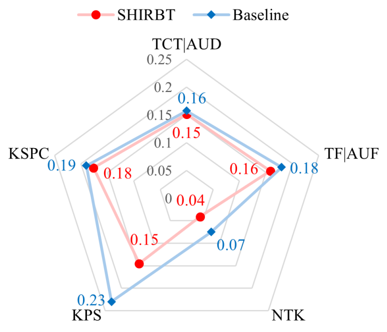
(DRE). For DRE, dimensionality reduction was performed using PCA to preserve 90–95% of the total variance before calculating reconstruction error.

7.1.1 Univariate Covariate Shift Comparison. In the univariate covariate shift analysis, we compared five representative features: time (TCT vs. AUD), frequency (TF vs. AUF), general typing (NTK), and calculated typing (KPS and KSPC). To ensure consistent feature definitions across participants, we extracted the SHIRBT features from a commonly used routine task ("Messenger chatting task") and the baseline features from the "Social & Communication category," as shown in Table 13. To assess the statistical significance of feature distribution changes, we applied the KS test at a threshold of $p = 0.05$. The results showed that SHIRBT features yielded statistically significant shifts in only 1–2 participants for most features, while baseline features exhibited more widespread distributional shifts. For example, in the case of NTK, only two participants showed significant changes in SHIRBT, whereas 5 participants showed such changes in baseline features, as detailed results in Supplement: E.1's Table A11. This trend was also evident in the numerical shift score analysis. JS divergence and PSI were used to compute average participant shift scores. As illustrated in the radar chart (Figure 10), SHIRBT exhibited lower average scores than baseline features across all five features. Participant-level comparisons in Figure 11 show that SHIRBT features had lower shift scores than baseline features for almost all participants (highlighted in red where SHIRBT had lower scores), as detailed in Supplement: E.1's Table A12 and A13. These findings suggest that SHIRBT features are more stable in terms of univariate distributional changes.

7.1.2 Multivariate Covariate Shift Comparison. In the multivariate analysis, we evaluated covariate shift across six feature-set configurations used in the stress prediction model: All, One, Two, One with Time, One with Frequency, and One with Typing, as shown in Table 13. For each configuration, we reconstructed both SHIRBT and baseline feature sets for the training and test intervals, and applied the three multivariate covariate shift metrics (MMD, DC, and DRE) to calculate shift scores. The radar chart in Figure 10 shows that SHIRBT features yielded lower participant-averaged shift scores than baseline features across all six configurations. Figure 11 further shows that, except for 1–2 participants, SHIRBT consistently exhibited lower shift scores than baseline features, as detailed in the results in Supplement: E.2's Table A14, A15, and A16. These findings indicate that SHIRBT features are more resilient to covariate shift across a range of feature combinations, which supports their suitability for robust deployment in real-world settings.

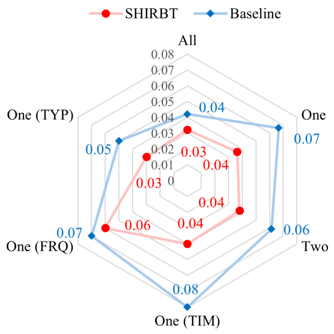


(a) PSI

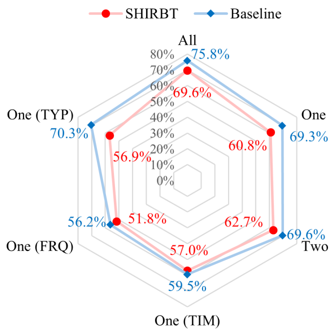


(b) JS

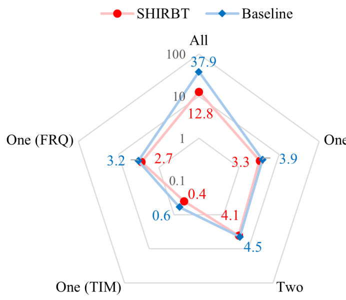
(A) Univariate



(c) MMD



(d) DC



(e) DRE

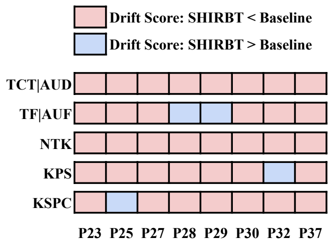
(B) Multivariate

Fig. 10. Comparison of average covariate shift detection scores across feature subsets of SHIRBT and baseline (see Table 13 for abbreviation definitions)

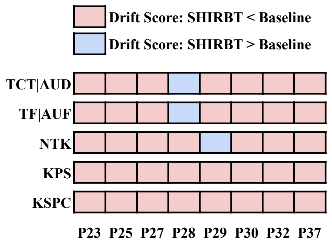
7.2 Schema Drift Comparison

Furthermore, we quantitatively analyzed the robustness of SHIRBT features against schema drift structural changes (i.e., feature evolution) in feature composition over time compared to conventional baseline features. This analysis addresses the issue of model performance degradation when the structure of input features changes between the training and testing phases. To measure schema drift, we split the long-term participant data into training (source) and testing (target) intervals, and quantitatively assessed the similarity and dissimilarity of the feature sets generated in each interval. For similarity, we used the Jaccard Similarity (expressed as a percentage), which measures the ratio of the intersection to the union of features between the two intervals; higher values indicate more structural consistency and lower drift. For dissimilarity, we computed the Symmetric Difference (in counts), representing the number of features that appear in only one of the two intervals, to quantify the absolute magnitude of structural change.

To evaluate SHIRBT's structural stability, we compared three conditions. First, following prior smartphone-based stress detection studies (Table 1), we considered the set of all apps that were used at least once during each

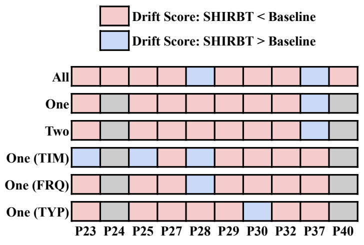


(a) PSI

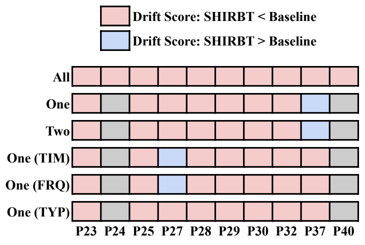


(b) JS

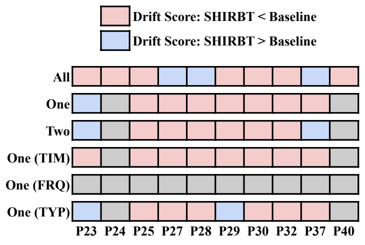
(A) Univariate



(c) MMD



(d) DC



(e) DRE

(B) Multivariate

Fig. 11. Per-participant comparison of covariate shift detection results between SHIRBT and baseline features across 5 detection methods (red: drift score where SHIRBT < baseline, blue: drift score where SHIRBT > baseline, gray: participant not available, see Table 13 for abbreviation definitions)

phase ("Apps ever used during each phase"). Second, following our baseline configuration, we considered only the apps used daily in each phase ("Apps used daily throughout each phase"). Third, for SHIRBT, we included only SHIRBT-based routine tasks that were observed daily in each phase ("SHIRBT's used daily throughout each phase"). The results of the Jaccard Similarity and Symmetric Difference comparisons across these three conditions are summarized in Table 14. The analysis showed that, except for participants P28 and P30, SHIRBT consistently exhibited higher Jaccard Similarity scores than both baseline conditions. For Symmetric Difference, SHIRBT showed equal or fewer structural changes than the two baselines in all participants except P28. These findings indicate that SHIRBT features undergo less structural variation over time and maintain greater consistency between training and testing phases. In conclusion, SHIRBT demonstrates superior robustness to structural drift compared to baseline features.

8 DISCUSSION

8.1 Summary of Major Contributions

This study developed **S**martphone **H**uman **I**nteraction-based **R**outine **B**ehavior **T**ask **M**ining, **M**odeling, and **F**eature extraction **(SHIRBT-MMF)** framework (Figure 2) to build a robust stress monitoring model based on daily smartphone usage patterns. This approach detects stress by deriving features from sequential interaction routine behaviors performed daily. While traditional stress models detect stress based on general smartphone

Table 14. Schema drift detection results on long-term users

Schema Drift Metrics	Used Apps or SHIRBTs	P23	P24	P25	P27	P28	P29	P30	P32	P37	P40
Jaccard Similarity (%)	Apps ever used during each phase	62.1	57.5	59.8	53.3	58.8	50.0	60.7	62.4	53.8	59.3
	Apps used daily throughout each phase	44.4	75.0	66.7	50.0	100.0	57.1	80.0	66.7	100.0	16.7
	SHIRBTs used daily throughout each phase	71.4	71.4	71.4	83.3	87.5	71.4	75.0	71.4	100.0	66.7
Symmetric Difference (# of apps or SHIRBTs)	Apps ever used during each phase	53	107	45	78	56	80	59	59	157	48
	Apps used daily throughout each phase	5	2	2	4	0	3	1	2	0	5
	SHIRBTs used daily throughout each phase	2	2	2	1	1	2	1	2	0	1

usage, overall app usage, and app category usage statistics captured over a specific period, SHIRBT-MMF extracts smartphone within-app behavioral sequential patterns that individuals perform daily. These patterns or routines are structured into a Smartphone Human Interaction Routine Behavior Task (SHIRBT), which is both consistent and easily interpretable. By quantifying these behaviors, SHIRBT-MMF enhances the explainability of daily stress monitoring from the perspectives of both users and service providers.

To achieve this, we developed a **Multi-Level Sequential Pattern Mining** technique to more accurately mine sequential patterns composed of within-app UI state events based on smartphone user interactions and daily routines. Additionally, we introduced an **LLM-based automated SHIRBT modeling** approach. To implement this effectively, we established naming, classification, and labeling criteria for automatic labeling by evaluating whether tasks met eight key criteria: Generality, Specificity, Comprehensibility, Accuracy, Comprehensiveness, Distinctiveness, Consistency, and Universality. Furthermore, to validate the high consistency and reproducibility of the LLM-based SHIRBT automation system, we conducted 200 repeated evaluations of the auto-labeling performance across different apps and their within-app UI sequences corresponding to the same task types in three major service categories (messaging, search, and email). When compared against ground truth labels, the system achieved reliable accuracy ranging from 95% to 100% based on both EMA and SMA criteria.

The extracted SHIRBT tasks were used to derive features through time, frequency, and typing behavior-related metrics, enabling a more quantitative assessment of stress. This approach improved stress detection accuracy compared to existing baseline models (accuracy 75.0% vs. 70.0%). Moreover, traditional stress models statistically capture app usage, app category usage, and overall smartphone usage over a specific period. As a result, the importance of learned features may change over time, leading to potential instability. In contrast, the proposed SHIRBT approach leverages task-based features that reflect users' daily and social routine activities, and was quantitatively shown through shift score analysis to more effectively mitigate covariate shift and feature evolution (especially schema drift) compared to baseline features, without requiring additional feature reconstruction or adaptive learning. In particular, SHIRBT features exhibit minimal structural change and high distributional consistency, thereby reducing the resource costs associated with data recollection, repeated model retraining, and feature selection, while enabling more stable and long-term operation in real-world service environments.

Additionally, the SHIRBT-based model offers interpretability. While traditional models rely on simple app usage statistics or category-based features, SHIRBT considers the context of user routine behavior, enabling a clearer understanding of its relationship with stress. For example, a user experiencing stress in a work environment may exhibit *delays in composing emails or responding to messages*, while a student preparing for exams may *frequently switch between study-related apps*. By analyzing such behaviors through SHIRBT, stress can be detected with greater precision, offering valuable insights for interpreting the user's state.

8.2 Possibility of Reducing Model Complexity and Addressing Privacy Concerns

The SHIRBT-MMF framework-based model offers privacy benefits by reducing the types and volume of data and model complexity without requiring feature selection or dimensionality reduction. Prior app-based stress detection studies have mainly relied on features derived from overall app usage or app category usage (Table 1), and even apps used only once during the training or evaluation period were often included as features. As a result, despite the reduced number of features, thousands of app usage logs still had to be collected, and continuous data collection was required even after model deployment. This increases the privacy burden on users, raises the likelihood of distribution shift or schema drift (i.e., feature evolution), and necessitates repeated retraining, leading to high resource and maintenance costs [3, 27, 39, 65].

We propose SHIRBT features, grounded in users' everyday and social routine tasks. SHIRBT reduces both the type and amount of data required without additional feature selection or dimensionality reduction, thereby alleviating model complexity and privacy concerns. On average, SHIRBT relies on only 5.0±1.63 app types, significantly fewer than the 110.9±53.4 apps used in baseline models (Table 10, 11). While diverse data are needed during initial model training, only a small set of frequently performed tasks needs to be monitored after deployment. We observed that the apps involved in SHIRBT remained largely unchanged over several months. Even when new apps were introduced, the underlying routines persisted, naturally mitigating covariate shift and feature evolution. Our shift score analysis confirms that SHIRBT is more robust than baseline features, and even a small number of core SHIRBT tasks can maintain high predictive performance.

SHIRBT features are generated by an LLM-based labeling system that automatically clusters sequential UI interaction patterns into task-level SHIRBTs, independent of app names or UI identifiers. This task-centric approach protects user privacy by avoiding the collection of sensitive UI elements (e.g., passwords) and enhances users' data literacy and trust in digital mental health tools. The compact structure of SHIRBT is well-suited for on-device and adaptive learning, reducing computational and transmission costs while improving privacy and scalability. In particular, when SHIRBT is configured based on globally shared tasks (e.g., messenger chatting) rather than personalized routines, the processes of data collection and preprocessing can be simplified. This enables interpretable behavior modeling without repeated data collection, supporting more efficient service deployment and maintenance. These characteristics greatly enhance the practicality and scalability of data-driven mental health care systems.

8.3 Robustness of Data Collection: Empirical Analysis across Devices, OS Version, and App Types

While the SHIRBT framework proposed in this study relies on Android accessibility events, various real-world deployment challenges—such as device- and OS-specific customization, app-level data collection restrictions, and diverse UI architectures—can potentially hinder consistent data acquisition [25, 33]. To address these concerns, we empirically evaluated the robustness and generalizability of SHIRBT across multiple device and application environments, as detailed in Supplement: A and B. First, we analyzed the devices and OS versions used by 26 participants, all of whom used Samsung Galaxy smartphones running Android OS versions 9–13. We confirmed that the key accessibility events required to construct SHIRBT—namely, `TYPE_WINDOW_STATE_CHANGED; WC`, `TYPE_VIEW_FOCUSED; VF`, `TYPE_VIEW_TEXT_CHANGED; TC`, and `TYPE_VIEW_TEXT_SELECTION_CHANGED; TS`—were consistently captured across all devices and OS versions without structural variation (Supplement: A).

We conducted experiments under four scenarios known to potentially restrict accessibility logging: (1) security-sensitive apps (e.g., Pay, Card, Bank), (2) pre-installed system apps (e.g., Phone, Gallery, Settings, Clock), (3) progressive web apps (PWAs, e.g., Pinterest, Flipboard), and (4) custom launchers and UIs (e.g., One UI, Nova, Microsoft Launcher). These experiments were carried out on a Galaxy S21 (Android 13) using a custom logger while recording the task execution on screen (see Supplement: B.1 and B.2). For security-sensitive apps, text input events were partially restricted for password fields, but task-level events (WC, VF) were consistently logged. In pre-installed apps, all key events were reliably recorded, and previously reported logging failures [33] were not observed. For PWAs, key events were captured; however, only the search engine domain was logged, not the actual services used. Under custom UI environments, event structures were consistently recorded across both system and third-party apps, indicating that SHIRBT construction was not affected.

The SHIRBT approach is practical and robust across various Android devices, operating systems, custom UIs, and even apps with restricted logging capabilities. For example, while sensitive apps may not log detailed inputs such as account numbers or passwords, high-level task behaviors like *"money transfer attempt," "mobile payment,"* or *"search"* can still be reliably inferred via window state change events. Similarly, in PWA environments, although URLs or specific content are not logged, SHIRBT focuses on extracting task-level behaviors such as *"web browsing,"* which not only reduces the impact of missing data but also enhances user privacy. We also empirically confirmed that the LLM can reliably interpret UI state events even for system apps or privately distributed apps with limited public documentation. Most participants regularly used fewer than five apps, which were typically well-known, minimizing the risk of unrecognized apps. However, in cases where the LLM might struggle with unfamiliar apps, we propose leveraging contextual UI information, offering user validation prompts, or temporarily labeling such instances as *"unknown"* for future refinement via active learning. Finally, although not observed in our experiments, certain devices (e.g., Xiaomi) may block accessibility logs entirely. In such cases, fallback strategies involving auxiliary sensors, APIs, or user-reported data may serve as effective alternatives.

8.4 Battery Consumption and User Behavior Implications of SHIRBT Derivation

SHIRBTs derivation require continuous collection of UI state events. This raises potential concerns about battery consumption and its impact on users' daily interaction patterns. Prior research has shown that continuous smartphone sensor logging can lead to significant battery drain, which in turn may cause cognitive and privacy burdens, behavioral distortions, increased charging frequency, and strategic user adaptations [37, 56, 73, 80]. However, these studies typically involved high-frequency sensor data collection (e.g., GPS, accelerometers, Bluetooth) in addition to app usage logs, which fundamentally differs from the logging strategy used for SHIRBT derivation. Our framework adopts an event-driven logging approach, in which logs are only recorded when user interaction occurs, such as changes in foreground activity or UI state events. It avoids real-time sensor sampling or continuous streaming, thereby minimizing resource consumption and improving energy efficiency.

To empirically validate this energy efficiency, we conducted a battery consumption experiment under three logging conditions: (1) logging disabled, (2) logging of app usage and accessibility events required for SHIRBT derivation, and (3) logging of app usage along with high-frequency sensor data (e.g., GPS, accelerometer), commonly used in prior behavioral monitoring studies. For each condition, three user scenarios—YouTube viewing, automated 5-minute app switching (via macro), and idle state—were executed for 4 hours on the same device and configuration, repeated three times (Supplement: C.1). Results showed no significant difference in battery usage between conditions (1) and (2), while condition (3) exhibited a slight increase in consumption (Supplement: C.2). These findings suggest that SHIRBT-related logging incurs virtually no additional battery cost compared to a non-logging baseline, indicating that it is a sustainable framework unlikely to distort user behavior in real-world settings. Furthermore, once SHIRBT routines are mined, there is no need to continuously collect all app usage data. Instead, selective logging of a small number of core apps (typically fewer than five) is sufficient to support the service. This design offers significant benefits in terms of energy efficiency and privacy preservation, and is well-suited for integration with on-device learning models. If future extensions of the SHIRBT framework involve additional sensor data, strategies such as context-aware adaptive logging, on-demand logging, and optimized sampling based on user context (e.g., time of day, battery level) can be employed to further minimize battery consumption.

8.5 Limitation and Future Works

8.5.1 Need for Diverse Demographics. This study primarily focuses on young adults (in their 20s and 30s). Future research should expand a more diverse range of age groups, occupations, cultural backgrounds, and smartphone operating system users (iOS vs. Android). This approach will allow for an analysis of how cultural and national differences influence stress prediction based on SHIRBT and account for variations in smartphone usage patterns across demographic factors. Furthermore, it is essential to examine the applicability of SHIRBT across different operating systems to ensure cross-platform consistency.

8.5.2 Long-Term Study and Routine Change Detection. While the SHIRBT feature-based model has been empirically shown to be more robust to covariate shift and feature evolution compared to baseline feature models, adapting to long-term changes in user behavior remains important. Although this study was conducted over a relatively long period (e.g., 4–5 months), a long-term study (over one year) may be necessary to fully validate SHIRBT's long-term stability. Such a study would enable us to assess how SHIRBT's performance evolves and detect distribution shifts caused by life events (e.g., job changes, seasonal factors, relocation, or marriage). As daily routines naturally evolve, future work investigates mechanisms that automatically detect new behavior patterns and incorporate them into the SHIRBT model. These needs could be addressed using lightweight distribution shift detection or anomaly detection techniques, enabling adaptive learning strategies such as online retraining, incremental learning, or domain adaptation [30, 36, 65, 67]. Although our study achieved strong performance without such adaptation by leveraging robust features, we plan to further investigate concept drift i.e., changes in the relationship between features and labels over time due to evolving routines in future work.

8.5.3 Potential for Expansion to Overall Mental Health. Although this study focuses on stress detection, SHIRBT-MMF can also be applied to other mental health issues such as depression, anxiety, and cognitive decline. For example, behavioral patterns associated with depression may manifest as reduced routine variability, decreased app-switching frequency, and increased passive content consumption, such as scrolling through social media. In the case of anxiety, it is likely to be characterized by increased irregular app switching and frequent notification-checking behaviors. For cognitive decline (e.g., dementia), long-term tracking of SHIRBT features should be conducted on a weekly or monthly basis rather than a daily basis to detect gradual pattern changes. Considering the older age demographic in this context, it is necessary to design a user interface and SHIRBT types (e.g., physical activity or mobility routine) that accommodate their needs.

8.5.4 Causal Analysis of SHIRBT and Stress. To clarify the causal relationship between SHIRBT and stress, future research should incorporate causal inference techniques [43]. For example, applying a pseudo-experiment design would enable the analysis of SHIRBT's direct impact on stress levels. Additionally, utilizing counterfactual analysis would allow for the evaluation of how hypothetical behavioral changes affect stress levels. For instance, by simulating a scenario in which a user reduces smartphone app-based social activities and work-related tasks (e.g., messenger chatting tasks and email reading tasks) before bedtime, it would be possible to determine whether specific SHIRBT patterns contribute to stress alleviation or exacerbation.

9 CONCLUSION

The SHIRBT-MMF framework provides a scalable, interpretable, and robust digital stress monitoring approach. While traditional stress models have limitations due to the instability and low interpretability of features, SHIRBT-based features improve reliability and interpretability through routine-based behavior modeling. Future research can advance this framework by expanding the user population, conducting long-term studies, broadening its application to various mental health issues, and incorporating causal inference techniques to enhance the explanatory power of stress prediction. Through these advancements, the SHIRBT-MMF framework has the potential to serve as a foundation for developing personalized mental healthcare solutions.

Acknowledgments

This research was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2025-02305705) and by the National Research Foundation (NRF) funded by the Korean government (MSIT) (RS-2022-NR068758, RS-2025-00559234).

References

[1] Zahraa S Abdallah, Mohamed Medhat Gaber, Bala Srinivasan, and Shonali Krishnaswamy. 2018. Activity recognition with evolving data streams: A review. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[2] Samuel Ackerman, Parijat Dube, Eitan Farchi, Orna Raz, and Marcel Zalmanovici. 2021. Machine learning model drift detection via weak data slices. In *2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*. IEEE, 1–8.

[3] Samuel Ackerman, Eitan Farchi, Orna Raz, Marcel Zalmanovici, and Parijat Dube. 2020. Detection of data drift and outliers affecting machine learning model performance over time. *arXiv preprint arXiv:2012.09258* (2020).

[4] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering*. IEEE, 3–14.

[5] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[6] Joost Asselbergs, Jeroen Ruwaard, Michal Ejdys, Niels Schrader, Marit Sijbrandij, Heleen Riper, et al. 2016. Mobile phone-based unobtrusive ecological momentary assessment of day-to-day mood: an explorative study. *Journal of medical Internet research* 18, 3 (2016), e5505.

[7] American Psychological Association et al. 2019. Stress in America, United States, 2007-2018. (2019).

[8] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. 2002. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 429–435.

[9] Joanne Banks and Emer Smyth. 2015. 'Your whole life depends on it': Academic stress and high-stakes testing in Ireland. *Journal of youth studies* 18, 5 (2015), 598–616.

[10] Gerald Bauer and Paul Lukowicz. 2012. Can smartphones detect stress-related changes in the behaviour of individuals?. In *2012 IEEE international conference on pervasive computing and communications workshops*. IEEE, 423–426.

[11] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *25th annual conference on neural information processing systems (NIPS 2011)*, Vol. 24. Neural Information Processing Systems Foundation.

[12] Sofian Berrouiguet, David Ramírez, María Luisa Barrigón, Pablo Moreno-Muñoz, Rodrigo Carmona Camacho, Enrique Baca-García, Antonio Artés-Rodríguez, et al. 2018. Combining continuous smartphone native sensors data capture and unsupervised data mining techniques for behavioral changes detection: a case series of the evidence-based behavior (eB2) study. *JMIR mHealth and uHealth* 6, 12 (2018), e9472.

[13] Andrey Bogomolov, Bruno Lepri, Michela Ferron, Fabio Pianesi, and Alex Pentland. 2014. Daily stress recognition from mobile phone data, weather conditions and individual traits. In *Proceedings of the 22nd ACM international conference on Multimedia*. 477–486.

[14] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[15] Alison Brunier. 2016. Investing in treatment for depression and anxiety leads to fourfold return. Retrieved August 1, 2025 from https://www.who.int/news/item/13-04-2016-investing-in-treatment-for-depression-and-anxiety-leads-to-fourfold-return

[16] Richard Chen, Filip Jankovic, Nikki Marinsek, Luca Foschini, Lampros Kourtis, Alessio Signorini, Melissa Pugh, Jie Shen, Roy Yaari, Vera Maljkovic, et al. 2019. Developing measures of cognitive impairment in the real world from consumer-grade multimodal sensor streams. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2145–2155.

[17] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[18] Prerna Chikersal, Afsaneh Doryab, Michael Tumminia, Daniella K Villalba, Janine M Dutcher, Xinwen Liu, Sheldon Cohen, Kasey G Creswell, Jennifer Mankoff, J David Creswell, et al. 2021. Detecting depression and predicting its onset using longitudinal symptoms captured by passive sensing: a machine learning approach with robust feature selection. *ACM Transactions on Computer-Human Interaction (TOCHI)* 28, 1 (2021), 1–41.

[19] Matteo Ciman and Katarzyna Wac. 2016. Individuals' stress assessment using human-smartphone interaction analysis. *IEEE Transactions on Affective Computing* 9, 1 (2016), 51–65.

[20] Sheldon Cohen. 1988. Perceived stress in a probability sample of the United States. (1988).

[21] Android Developer. 2025. AccessibilityEvent. Retrieved August 1, 2025 from https://developer.android.com/reference/android/view/accessibility/AccessibilityEvent

[22] Android Developer. 2025. AccessibilityService. Retrieved August 1, 2025 from https://developer.android.com/reference/android/accessibilityservice/AccessibilityService

[23] Android Developer. 2025. NotificationListenerService. Retrieved August 1, 2025 from https://developer.android.com/reference/android/location/LocationListener

[24] Android Developer. 2025. NotificationManager. Retrieved August 1, 2025 from https://developer.android.com/reference/android/app/NotificationManager

[25] Android Developer. 2025. Understand the UI Layer. Retrieved August 1, 2025 from https://developer.android.com/topic/architecture/ui-layer

[26] Android Developer. 2025. UsageStatsManager. Retrieved August 1, 2025 from https://developer.android.com/reference/android/app/usage/UsageStatsManager#constants_1

[27] Sijie Dong, Qitong Wang, Soror Sahri, Themis Palpanas, and Divesh Srivastava. 2024. Efficiently Mitigating the Impact of Data Drift on Machine Learning Pipelines. *Proceedings of the VLDB Endowment* 17, 11 (2024), 3072–3081.

[28] Maged El-Sayed, Carolina Ruiz, and Elke A Rundensteiner. 2004. FS-Miner: efficient and incremental mining of frequent sequence patterns in web logs. In *Proceedings of the 6th annual ACM international workshop on web information and data management*. 128–135.

[29] Christie I Ezeife, Yi Lu, and Yi Liu. 2005. PLWAP sequential mining: open source code. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*. 26–35.

[30] Imen Ferjani and Suleiman Ali Alsaif. 2024. Dynamic road anomaly detection: Harnessing smartphone accelerometer data with incremental concept drift detection and classification. *Sensors* 24, 24 (2024), 8112.

[31] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.

[32] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[33] Julien Gamba, Mohammed Rashed, Abbas Razaghpanah, Juan Tapiador, and Narseo Vallina-Rodriguez. 2020. An analysis of pre-installed android software. In *2020 IEEE symposium on security and privacy (SP)*. IEEE, 1039–1055.

[34] Surjya Ghosh, Sumit Sahu, Niloy Ganguly, Bivas Mitra, and Pradipta De. 2019. EmoKey: An emotion-aware smartphone keyboard for mental health monitoring. In *2019 11th international conference on communication systems & networks (COMSNETS)*. IEEE, 496–499.

[35] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*. IEEE Piscataway, NJ, USA, 215–224.

[36] Wei Hao, Zixi Wang, Lauren Hong, Lingxiao Li, Nader Karayanni, AnMei Dasbach-Prisk, Chengzhi Mao, Junfeng Yang, and Asaf Cidon. 2025. Nazar: Monitoring and Adapting ML Models on Mobile Devices. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*. 746–761.

[37] Gabriel M Harari, Nicholas D Lane, Rui Wang, Benjamin S Crosier, Andrew T Campbell, and Samuel D Gosling. 2016. Using smartphones to collect behavioral data in psychological science: Opportunities, practical considerations, and challenges. *Perspectives on Psychological Science* 11, 6 (2016), 838–854.

[38] Marwan Hassani and Thomas Seidl. 2011. Towards a mobile health context prediction: Sequential pattern mining in multiple streams. In *2011 IEEE 12th International Conference on Mobile Data Management*, Vol. 2. IEEE, 55–57.

[39] Bo-Jian Hou, Lijun Zhang, and Zhi-Hua Zhou. 2017. Learning with feature evolvable streams. *Advances in Neural Information Processing Systems* 30 (2017).

[40] Kuo-Wei Hsu. 2017. Effectively mining time-constrained sequential patterns of smartphone application usage. In *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*. 1–8.

[41] Emma C Hurley, Ian R Williams, Adrian J Tomyn, and Lena Sanci. 2024. Social media use among Australian university students: Understanding links with stress and mental health. *Computers in Human Behavior Reports* 14 (2024), 100398.

[42] Natasha Jaques, Sara Taylor, Asaph Azaria, Asma Ghandeharioun, Akane Sano, and Rosalind Picard. 2015. Predicting students' happiness from physiology, phone, mobility, and behavioral data. In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 222–228.

[43] Gyuwon Jung, Sangjun Park, and Uichin Lee. 2024. DeepStress: Supporting Stressful Context Sensemaking in Personal Informatics Systems Using a Quasi-experimental Approach. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–18.

[44] Soowon Kang, Woohyeok Choi, Cheul Young Park, Narae Cha, Auk Kim, Ahsan Habib Khandoker, Leontios Hadjileontiadis, Heepyung Kim, Yong Jeong, and Uichin Lee. 2023. K-emophone: A mobile and wearable dataset with in-situ emotion, stress, and attention labels. *Scientific data* 10, 1 (2023), 351.

[45] Soowon Kang, Cheul Young Park, Auk Kim, Narae Cha, and Uichin Lee. 2022. Understanding emotion changes in mobile experience sampling. In *Proceedings of the 2022 Chi conference on human factors in computing systems*. 1–14.

[46] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017), 3146–3154.

[47] Taewan Kim, Haesoo Kim, Ha Yeon Lee, Hwarang Goh, Shakhboz Abdigapporov, Mingon Jeong, Hyunsung Cho, Kyungsik Han, Youngtae Noh, Sung-Ju Lee, et al. 2022. Prediction for retrospection: Integrating algorithmic stress prediction into personal informatics systems for college students' mental health. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. 1–20.

[48] Zachary D King, Judith Moskowitz, Begum Egilmez, Shibo Zhang, Lida Zhang, Michael Bass, John Rogers, Roozbeh Ghaffari, Laurie Wakschlag, and Nabil Alshurafa. 2019. Micro-stress EMA: A passive sensing framework for detecting in-the-wild stress in pregnant mothers. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 3, 3 (2019), 1–22.

[49] Jeomoan Francis Kurian and Mohamed Allali. 2024. Detecting drifts in data streams using Kullback-Leibler (KL) divergence measure for data engineering applications. *Journal of Data, Information and Management* 6, 3 (2024), 207–216.

[50] Hosub Lee, Young Sang Choi, Sunjae Lee, and IP Park. 2012. Towards unobtrusive emotion recognition for affective social communication. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 260–264.

[51] Hansoo Lee, Auk Kim, SangWon Bae, and Uichin Lee. 2024. S-ADL: Exploring Smartphone-based Activities of Daily Living to Detect Blood Alcohol Concentration in a Controlled Environment. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–25.

[52] Hansoo Lee, Joonyoung Park, and Uichin Lee. 2022. A systematic survey on android api usage for data-driven analytics with smartphones. *Comput. Surveys* 55, 5 (2022), 1–38.

[53] Yue-Shi Lee and Show-Jane Yen. 2008. Incremental and interactive mining of web traversal patterns. *Information Sciences* 178, 2 (2008), 287–306.

[54] Florian Lettner, Christian Grossauer, and Clemens Holzmann. 2014. Mobile interaction analysis: towards a novel concept for interaction sequence mining. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*. 359–368.

[55] Jieun Lim, Youngji Koh, Auk Kim, and Uichin Lee. 2024. Exploring Context-Aware Mental Health Self-Tracking Using Multimodal Smart Speakers in Home Environments. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–18.

[56] Jason I Lin, Julie Liu, Nicholas D Lane, Fan Chen, Tanzeem Choudhury, and Andrew T Campbell. 2012. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 501–510.

[57] Jake Linardon, Joseph Firth, John Torous, Mariel Messer, and Matthew Fuller-Tyszkiewicz. 2024. Efficacy of mental health smartphone apps on stress levels: a meta-analysis of randomised controlled trials. *Health psychology review* 18, 4 (2024), 839–852.

[58] Eric Hsueh-Chan Lu, Yi-Wei Lin, and Jing-Bin Ciou. 2014. Mining mobile application sequential patterns for usage prediction. In *2014 IEEE International Conference on Granular Computing (GrC)*. IEEE, 185–190.

[59] Eric Hsueh-Chan Lu and Ya-Wen Yang. 2018. Mining mobile application usage pattern for demand prediction by considering spatial and temporal relations. *GeoInformatica* 22 (2018), 693–721.

[60] Sheng-Chieh Lu, Wenye Song, Andre Pfob, and Chris Gibbons. 2025. Assessing the representativeness of large medical data using population stability index. *BMC Medical Research Methodology* 25, 1 (2025), 44.

[61] Nizar R Mabroukeh and Christie I Ezeife. 2010. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)* 43, 1 (2010), 1–41.

[62] I Scott MacKenzie and Kumiko Tanaka-Ishii. 2010. *Text entry systems: Mobility, accessibility, universality*. Elsevier.

[63] Stephanie Marken. 2024. Mental Health, Stress Top Reasons Students Consider Leaving. Retrieved August 1, 2025 from https://news.gallup.com/poll/644645/mental-health-stress-top-reasons-students-consider-leaving.aspx

[64] Alban Maxhuni, Pablo Hernandez-Leal, Eduardo F Morales, L Enrique Sucar, Venet Osmani, and Oscar Mayora. 2020. Unobtrusive stress assessment using smartphones. *IEEE Transactions on Mobile Computing* 20, 6 (2020), 2313–2325.

[65] Lakmal Meegahapola, Hamza Hassoune, and Daniel Gatica-Perez. 2024. M3BAT: Unsupervised domain adaptation for multimodal mobile sensing with multi-branch adversarial training. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 2 (2024), 1–30.

[66] Abhinav Mehrotra, Fani Tsapeli, Robert Hendley, and Mirco Musolesi. 2017. MyTraces: Investigating correlation and causation between users' emotional states and mobile phone interaction. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21.

[67] Shikha Mehta et al. 2017. Concept drift in streaming data classification: algorithms, platforms and issues. *Procedia computer science* 122 (2017), 804–811.

[68] Jennifer Melcher, Ryan Hays, and John Torous. 2020. Digital phenotyping for mental health of college students: a clinical review. *BMJ Ment Health* 23, 4 (2020), 161–166.

[69] Microsoft. 2025. How to handle schema drift in Azure Data Factory Data Flow. Retrieved August 1, 2025 from https://learn.microsoft.com/en-us/azure/data-factory/concepts-data-flow-schema-drift

[70] Ivan Moura, Ariel Teles, Davi Viana, Jean Marques, Luciano Coutinho, and Francisco Silva. 2022. Digital phenotyping of mental health using multimodal sensing of multiple situations of interest: A systematic literature review. *Journal of Biomedical Informatics* (2022), 104278.

[71] Amir Muaremi, Bert Arnrich, and Gerhard Tröster. 2013. Towards measuring stress with smartphones and wearable devices during workday and sleep. *BioNanoScience* 3 (2013), 172–183.
[72] Abhishek Mukherji, Vijay Srinivasan, and Evan Welbourne. 2014. Adding intelligence to your mobile device via on-device sequential pattern mining. In *Proceedings of the 2014 acm international joint conference on pervasive and ubiquitous computing: Adjunct publication*. 1005–1014.
[73] Elizabeth L Murnane, Dan Cosley, Patrick Chang, Shion Guha, Ellen Frank, Geri Gay, and Mark Matthews. 2016. Self-monitoring practices, attitudes, and needs of individuals with bipolar disorder: Implications for the design of technologies to manage mental health. *Journal of the American Medical Informatics Association* 23, 3 (2016), 477–484.
[74] Skogby Steinholtz Olof. 2018. A comparative study of black-box optimization algorithms for tuning of hyper-parameters in deep neural networks.
[75] OpenAI. 2024. GPT-4o. Retrieved August 1, 2025 from https://openai.com/index/hello-gpt-4o/
[76] OpenAI. 2024. text-embedding-3-large. Retrieved August 1, 2025 from https://platform.openai.com/docs/models/text-embedding-3-large
[77] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Hua Zhu. 2000. Mining access patterns efficiently from web logs. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 396–407.
[78] Andrew K Przybylski, Kou Murayama, Cody R DeHaan, and Valerie Gladwell. 2013. Motivational, emotional, and behavioral correlates of fear of missing out. *Computers in human behavior* 29, 4 (2013), 1841–1848.
[79] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. 2019. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems* 32 (2019).
[80] Mashfiqui Rabbi, Myok Ko Aung, Ming Zhang, and Tanzeem Choudhury. 2015. MyBehavior: Automatic personalized health feedback from user behaviors and preferences using smartphones. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 707–718.
[81] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).
[82] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.
[83] John Rooksby, Alistair Morrison, and Dave Murray-Rust. 2019. Student perspectives on digital phenotyping: The acceptability of using smartphone data to assess mental health. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–14.
[84] Akane Sano, Andrew J Phillips, Z Yu Amy, Andrew W McHill, Sara Taylor, Natasha Jaques, Charles A Czeisler, Elizabeth B Klerman, and Rosalind W Picard. 2015. Recognizing academic performance, sleep quality, stress level, and mental health using personality traits, wearable sensors and mobile phones. In *2015 IEEE 12th international conference on wearable and implantable body sensor networks (BSN)*. IEEE, 1–6.
[85] Akane Sano and Rosalind W Picard. 2013. Stress recognition using wearable sensors and mobile phones. In *2013 Humaine association conference on affective computing and intelligent interaction*. IEEE, 671–676.
[86] Stefanie Scherzinger, Meike Klettke, and Uta Störl. 2013. Managing schema evolution in NoSQL data stores. *arXiv preprint arXiv:1308.0514* (2013).
[87] M Scott, Lee Su-In, et al. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017), 4765–4774.
[88] Jaime Cespedes Sisniega, Vicente Rodríguez, German Molto, and Álvaro López García. 2024. Efficient and scalable covariate drift detection in machine learning systems with serverless computing. *Future Generation Computer Systems* 161 (2024), 174–188.
[89] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining sequential patterns: Generalizations and performance improvements. In *International conference on extending database technology*. Springer, 1–17.
[90] Vijay Srinivasan, Saeed Moghaddam, Abhishek Mukherji, Kiran K. Rachuri, Chenren Xu, and Emmanuel Munguia Tapia. 2014. MobileMiner: Mining Your Frequent Patterns on Your Phone. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Seattle, Washington) *(UbiComp '14)*. Association for Computing Machinery, New York, NY, USA, 389–400.
[91] Statista. 2020. Most frequently used smartphone apps in South Korea as of September 2020, by category. Retrieved August 1, 2025 from https://www.statista.com/statistics/897227/south-korea-frequently-used-smartphone-apps-by-category/
[92] Statista. 2020. Most used smartphone functions in South Korea from 2014 to 2017. Retrieved August 1, 2025 from https://www.statista.com/statistics/953819/south-korea-mainly-used-smartphone-functions/
[93] The Chromium Projects. 2008. The Chromium Projects. Retrieved August 1, 2025 from https://www.chromium.org/chromium-projects/
[94] Anja Thieme, Danielle Belgrave, and Gavin Doherty. 2020. Machine learning in mental health: A systematic review of the HCI literature to support the development of effective and implementable ML systems. *ACM Transactions on Computer-Human Interaction (TOCHI)* 27, 5 (2020), 1–53.
[95] John Torous, Mathew V Kiang, Jeanette Lorme, Jukka-Pekka Onnela, et al. 2016. New tools for new research in psychiatry: a scalable and customizable platform to empower data driven smartphone research. *JMIR mental health* 3, 2 (2016), e5165.

[96] Eugene Tuv, Alexander Borisov, George Runger, and Kari Torkkola. 2009. Feature selection with ensembles, artificial variables, and redundancy elimination. *The Journal of Machine Learning Research* 10 (2009), 1341–1366.
[97] Niels Van Berkel, Denzil Ferreira, and Vassilis Kostakos. 2017. The experience sampling method on mobile devices. *ACM Computing Surveys (CSUR)* 50, 6 (2017), 1–40.
[98] Dilip Venkatesh and Sundaresan Raman. 2024. Bits pilani at semeval-2024 task 1: Using text-embedding-3-large and labse embeddings for semantic textual relatedness. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*. 865–868.
[99] Elena Vildjiounaite, Johanna Kallio, Vesa Kyllönen, Mikko Nieminen, Ilmari Määttänen, Mikko Lindholm, Jani Mäntyjärvi, and Georgy Gimel'farb. 2018. Unobtrusive stress detection on the basis of smartphone usage data. *Personal and Ubiquitous Computing* 22 (2018), 671–688.
[100] Xuhai Xu, Prerna Chikersal, Afsaneh Doryab, Daniella K Villalba, Janine M Dutcher, Michael J Tumminia, Tim Althoff, Sheldon Cohen, Kasey G Creswell, J David Creswell, et al. 2019. Leveraging routine behavior and contextually-filtered features for depression detection among college students. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 3, 3 (2019), 1–33.
[101] Xuhai Xu, Prerna Chikersal, Janine M Dutcher, Yasaman S Sefidgar, Woosuk Seo, Michael J Tumminia, Daniella K Villalba, Sheldon Cohen, Kasey G Creswell, J David Creswell, et al. 2021. Leveraging collaborative-filtering for personalized behavior modeling: a case study of depression detection among college students. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–27.
[102] Mohammed J Zaki. 2001. SPADE: An efficient algorithm for mining frequent sequences. *Machine learning* 42 (2001), 31–60.
[103] Panyu Zhang, Gyuwon Jung, Jumabek Alikhanov, Uzair Ahmed, and Uichin Lee. 2024. A reproducible stress prediction pipeline with mobile sensor data. *Proceedings of the ACM on interactive, mobile, wearable and ubiquitous technologies* 8, 3 (2024), 1–35.
[104] Xiao Zhang, Wenzhong Li, Xu Chen, and Sanglu Lu. 2018. Moodexplorer: Towards compound emotion detection via smartphone sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (2018), 1–30.
[105] Zhen-Yu Zhang, Peng Zhao, Yuan Jiang, and Zhi-Hua Zhou. 2020. Learning with feature and distribution evolvable streams. In *International Conference on Machine Learning*. PMLR, 11317–11327.

A Multi-Level Sequential Pattern Mining Algorithm: Implementation

Algorithm 1 Multi-Level Sequential Pattern Mining Algorithm

Notation
- e: Within-app UI state event
- S: Within-app UI state event-based sequence
- SessionID: Screen on/off session ID
- S_{prefix}: Prefix sequence
- S_{suffix}: Suffix sequence
- ML-SDB$|_{S_{\text{prefix}}}$: S_{prefix}-projected ML-SDB

```
Procedure: ML-SPM
Input: ML-SDB
Output: PatternSet // A set of output sequential patterns
 1: /*Identifies daily frequent events*/
 2: SupportDict ← Empty Multi-Level Support Dictionary
 3: for all Day, SessionID, S in ML-SDB do
 4:     for all $e$ in S do
 5:         SupportDict[$e$] ← SupportDict[$e$] ∪Day
 6:     end for
 7: end for
 8: EventSet ← $\{e \mid e|\text{SupportDict}[e]|/\textit{TotalNumerOfDates} = 1.0\}$
 9: /*Mines contiguous sequential patterns*/
10: PatternSet ←Empty Set // Initialize output sequential pattern set
11: for all $e$ in EventSet do
12:     $S_{\text{prefix}} \leftarrow \langle e \rangle$ // Create prefix sequence
13:     ML-SDB$|_{S_{\text{prefix}}}$ ← Empty Set // Initialize new projected database
14:     PatternSet ← PatternSet ∪ $S_{\text{prefix}}$
15:     // Below codes project ML-SDB by every frequent event
16:     for all Day, SessionID, S ∈ ML-SDB do
17:         $S_{\text{suffix}} \leftarrow S[j+1:]$ if $S[j] = e$ else $\langle\rangle$
18:         ML-SDB$|_{S_{\text{prefix}}}$ ← ML-SDB$|_{S_{\text{prefix}}}$ ∪ $\langle\langle$Day, SessionID$\rangle, S_{\text{suffix}}\rangle$
19:     end for
20:     ML-SPM_sub(ML-SDB$|_{S_{\text{prefix}}}$, $S_{\text{prefix}}$) // Call subroutine function ►
21: end for
```

```
Function: ML-SPM_sub(ML-SDB$|_{S_{\text{prefix}}}$, $S_{\text{prefix}}$)
 1: /*Finds and extends contiguous sequential patterns*/
 2: SupportDict ← Empty Multi-Level Support Dictionary
 3: for all Day, SessionID, S ∈ ML-SDB$|_{S_{\text{prefix}}}$ do
 4:     FirstEvent ← S[1]
 5:     SupportDict[FirstEvent] ← SupportDict[FirstEvent] ∪Day
 6: end for
 7: EventSet ← $\{e \mid e|\text{SupportDict}[e]|/\textit{TotalNumerOfDates} = 1.0\}$
 8: for all $e$ in EventSet do
 9:     $S'_{\text{prefix}}$ ← Extend($S_{\text{prefix}}$, $e$) // Create extended sequential pattern
10:     PatternSet ← PatternSet ∪ $S'_{\text{prefix}}$ // Add new sequential pattern
11:     ML-SDB$|_{S'_{\text{prefix}}}$ ← Empty Set // Initialize new projected database
12:     for all Day, SessionID, S ∈ ML-SDB$|_{S_{\text{prefix}}}$ do
13:         $S_{\text{suffix}} \leftarrow S[2:]$ if $S[1] = e$ else $\langle\rangle$
14:         ML-SDB$|_{S'_{\text{prefix}}}$ ← ML-SDB$|_{S'_{\text{prefix}}}$ ∪ $\langle\langle$Day, SessionID$\rangle, S_{\text{suffix}}\rangle$
15:     end for
16:     ML-SPM_sub(ML-SDB$|_{S'_{\text{prefix}}}$, $S'_{\text{prefix}}$) // Call subroutine recursively
17: end for
```

We developed the Multi-Level Sequential Pattern Mining (ML-SPM) Algorithm based on the pattern-growth method (i.e., PrefixSpan [35]) to mine only daily frequent contiguous sequential patterns. The ML-SPM algorithm first finds all frequent event e such that ML-sup$(\langle e \rangle)/|d_{total}| = 1.0$ from Multi-Level Sequence Database (ML-SDB) as shown in lines 1–8 of the procedure ML-SPM in Algorithm 1. Then, the algorithm projects ML-SDB using the frequent event e to create an $\langle e \rangle$-projected ML-SDB (denoted as ML-SDB$|_{\langle e \rangle}$) and calls the subroutine function ML-SPM_sub(ML-SDB$|_{\langle e \rangle}$, $\langle e \rangle$) to find and extract sequential patterns as shown in lines 9–21 of the procedure ML-SPM in Algorithm 1. In sequential pattern mining, a prefix sequence refers to the initial subsequence of a given sequence, which serves as the basis for projected databases. Here, the sequence $\langle e \rangle$ is called the prefix sequence of the $\langle e \rangle$-projected ML-SDB, and the prefix sequence S can be denoted as S_{prefix}. Similarly, the remaining sequence after removing the prefix sequence is called the suffix sequence of the $\langle e \rangle$-projected ML-SDB, and the suffix sequence S can be denoted as S_{suffix}. The suffix sequence consists of all events that follow the prefix sequence within the same projected sequence, preserving their original order. To formally define these sequences, let $S = \langle e_1 e_2 \ldots e_n \rangle$ be a sequence of events. The notation $S[i]$ represents the event at index i within sequence S, where $S[1]$ denotes the first event and $S[n]$ denotes the last event. When a prefix sequence S_{prefix} of length k is given, the corresponding suffix sequence S_{suffix} is defined as $S[k+1:]$, which consists of all events from index $k+1$ to n, preserving their original order. If $k = n$, meaning the prefix sequence includes all events, then $S[k+1:]$ is an empty sequence (i.e., $\langle \rangle$). The detailed process of mining sequential patterns by calling ML-SPM_sub(ML-SDB$|_{\langle e \rangle}$, $\langle e \rangle$) is detailed in lines 1-17 of the **Function:** ML-SPM_sub(ML-SDB$|_{S_{\text{prefix}}}$) in Algorithm 1. The function ML-SPM_sub(ML-SDB$|_{S_{\text{prefix}}}$, S_{prefix}) takes the S_{prefix}-projected ML-SDB ML-SDB$|_{S_{\text{prefix}}}$ and the prefix sequence S_{prefix} as input parameters. The ML-SPM algorithm recursively repeats the process of finding new sequential patterns and creating new projected ML-SDB.

B LLM-based Automated SHIRBT Modeling System

We evaluated multiple naming rule candidates through user assessments to generate optimized SHIRBT names that are easily understandable by humans and developed an LLM-based automated SHIRBT modeling system as mentioned in Section 3.2. This appendix provides additional details on the naming rule candidates evaluation and prompt engineering process in system development, expanding on the information introduced in Section 3.2. We provided the user evaluation process for selecting the optimal naming rule in Section B.1, the prompt engineering approach for training the LLM-based automated SHIRBT modeling system in Section B.2, and Section B.3 reports the system's detailed performance in task clustering and labeling.

B.1 Generating Optimized Task Name via User Evaluation

We interviewed three researchers with experience in app development and smartphone usage to evaluate three SHIRBT naming rule candidates: (1) general behavior-based tasks, (2) service-specific behavior-based tasks, and (3) app-specific behavior-based tasks, as shown in Figure 12. General behavior-based tasks are not limited to

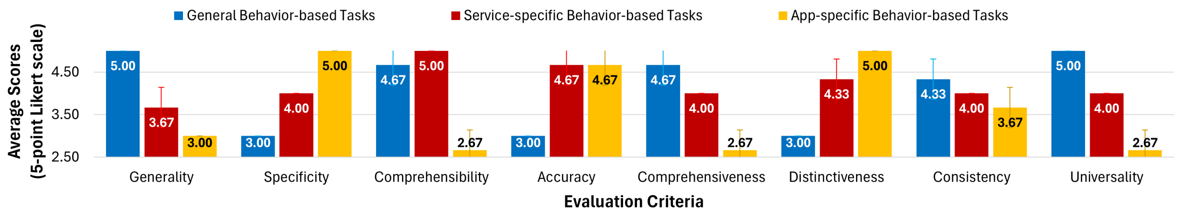


Fig. 12. Evaluation of task naming rule candidates

specific services or apps, offering high versatility. For example, chatting on Messenger or SNS can be simply modeled as a "Chatting" task. This type scored highest in Generality, Comprehensiveness, and Universality, but due to the lack of contextual information, it was rated lower in Specificity, Accuracy, and Distinctiveness. Service-specific behavior-based tasks reflect the context of a particular service, allowing for more accurate meaning delivery. For instance, listening to music on a streaming service can be expressed as a "Music streaming listening" task. This approach scored higher in Specificity, Comprehensibility, Accuracy, and Distinctiveness, but since it requires familiarity with the service, it received lower scores in Generality, Comprehensiveness, Consistency, and Universality as shown in Figure 12. App-specific behavior-based tasks include app names in SHIRBT names, requiring a specific user experience. For example, "Apple Music listening" or "Spotify listening" are easily understood by many due to their global popularity. However, apps used mostly in specific regions, like Korea's Melon, may not be as widely recognized. This type scored low in Generality, Comprehensiveness, Consistency, and Universality, but achieved the highest scores in Specificity, Accuracy, and Distinctiveness.

The average total scores were 33.67±0.47 for service-specific behavior-based tasks, 32.67±0.47 for general behavior-based tasks, and 29.33±0.94 for app-specific behavior-based tasks. Service-specific behavior-based tasks achieved the highest overall score by balancing across all evaluation criteria. General behavior-based tasks were too broad, leading to ambiguity, while app-specific tasks lacked universality and scored the lowest. Therefore, we generated SHIRBT names based on service-specific behavior-based tasks.

B.2 Building LLM-based Automated SHIRBT Modeling System via Prompt Engineering

As explained in Section 3.2, we developed an automated system that models sequential patterns into SHIRBTs in compliance with task labeling evaluation criteria, naming rules and labeling criteria through prompt engineering for LLM. To ensure that the LLM accurately models SHIRBTs while adhering to task labeling evaluation criteria, naming rules and labeling criteria through, we designed the prompt shown in **LLM-based Automated SHIRBT Modeling System**. *1. Task Naming Rule* outlines the rules for generating SHIRBT names in a standardized and easily understandable by human. *2. Task Categorization & Labeling Criteria* specifies the categorization and labeling criteria for SHIRBTs based on the different order, repetition, representativeness, and functional similarity of events composing the sequential pattern. Additionally, to enhance the accuracy of the LLM-based automated SHIRBT modeling system, we applied *3. Few-shot Prompting and Chain-of-Thought Reasoning in Applying Naming Rules & Labeling Criteria* in our system.

LLM-based Automated SHIRBT Modeling System

This machine classifies sequential patterns composed of package name and class name events based on click streams into smartphone human interaction behavior tasks and assigns names to make them easier for humans to understand. In this process, the sequential patterns are composed of fully qualified class names, which are composed of an app package name (e.g., com.kakao.talk) and a class name (e.g., activity.main.MainActivity), such as com.kakao.talk.activity.main.MainActivity. The criteria and rules for naming, categorizing and labeling smartphone human interaction behavior tasks based on sequential patterns are as follows.

1. Task Naming Rule

Naming Rule 1: Considering the eight evaluation criteria above, a smartphone human interaction behavior task name should maintain consistency and follow one of these formats.

(1) If the verb describing human behavior is intransitive, the task name should follow "*A noun representing the app service's characteristic, not a specific app name*" + "*A gerund describing the human action*" + *task.*

(2) If the verb describing human behavior is transitive, the task name should follow "*A noun representing the app service's characteristic, not a specific app name*" + "*A noun describing the object of the action*" + "*A gerund describing the human action*" + "*task*".

Naming Rule 2: If the sequential pattern is ⟨A⟶B⟩, the task name is assigned based on the representative action that the user can typically perform in activity A.

2. Task Categorization & Labeling Criteria

When you have various sequential patterns composed of two or more events, and their sequences or repetitions differ, the task labeling criteria are as follows.

- *Criteria 1*: If the events composing two sequential patterns are the same but occur in a different order, the resulting smartphone human interaction behavior task should also differ accordingly.
- *Criteria 2*: If the same events make up a sequential pattern and occur two or more times, multiple actions may appear within one sequential pattern. In that case, if one representative task encompasses all actions, unify them into one. Also, if the same event pair (e.g., ⟨A⟶B⟩) is repeated (e.g., ⟨A⟶B⟶A⟶B⟩), do not repeat the task name but represent it once.
- *Criteria 3*: If the same events are repeated two or more times within a sequential pattern and multiple distinct tasks appear—yet no single task can comprehensively cover the others—do not unify them but instead express them as a combined form.
- *Criteria 4*: When there are two or more sequential patterns, if the app package name and class name differ but they share similar app service characteristics and the functional meanings of the class names are similar, they can be labeled under the same smartphone human interaction behavior task.

3. Few-shot Prompting and Chain-of-Thought Reasoning in Applying Naming Rules & Labeling Criteria

Given Sequential Pattern	Task Name	Criteria	CoT Reasoning
⟨KT.MA⟶KT.CRHA⟩	Messenger chat list browsing task	1	Moving from the main activity to the chat list implies browsing the chat list.
⟨KT.CRHA⟶KT.MA⟩	Messenger chatting task	1	Returning from a chat room implies the user was chatting.
⟨KT.MA⟶KT.CRHA⟶KT.MA⟩ ⟨KT.CRHA⟶KT.MA⟶KT.CRHA⟩ ⟨KT.MA⟶KT.CRHA⟶KT.MA⟶KT.CRHA⟩	Messenger chatting task	2	Include chat list browsing and chatting, but chatting is primary. Repeated main-chat transitions make chatting the representative behavior. To avoid redundancy, these patterns are unified as Messenger chatting task.
⟨GM.MAG⟶GM.CAG⟩	Email reading task	1	Transitioning from mail activity to compose suggests reading an email before replying.
⟨GM.CAG⟶GM.MAG⟩	Email writing task	1	Switching from compose activity back to mail suggests writing an email.
⟨GM.MAG⟶GM.CAG⟶GM.MAG⟩ ⟨GM.CAG⟶GM.MAG⟶GM.CAG⟩	Email reading &writing task	3	Include email reading and writing in different orders. As they are distinct actions, they cannot be merged. Thus, the task is Email reading & writing task, regardless of sequence.
⟨CHRM.CTA⟶CHRM.SA⟶AC.FL⟩ ⟨NAS.MA⟶NAS.SWSLA⟶NAS.IABA⟩	Web browsing task	4	Involve web searching and browsing in Chrome & Naver. Since searching serves as a means for browsing, they merge into Web browsing task. Similar behaviors fall into the same task.

Abbreviations:
KT.MA = `com.kakao.talk.activity.main.MainActivity`,
KT.CRHA = `com.kakao.talk.activity.chatroom.ChatRoomHolderActivity`,
CHRM.CTA = `com.android.chrome.org.chromium.chrome.browser.ChromeTabbedActivity`,
CHRM.SA = `com.android.chrome.org.chromium.chrome.browser.searchwidget.SearchActivity`,
AC.FL = `com.android.chrome.android.widget.FrameLayout`,
GM.MAG = `com.google.android.gm.ui.MailActivityGmail`,
GM.CAG = `com.google.android.gm.ui.ComposeActivityGmail`,
NAS.MA = `com.nhn.android.search.proto.MainActivity`,
NAS.SWSLA = `com.nhn.android.search.browser.control.searchwindow.suggest.SearchWindowSuggestListActivity`,
NAS.IABA = `com.nhn.android.search.browser.InAppBrowserActivity`

B.3 Evaluation of LLM-Based SHIRBT Modeling Performance Measurement

Table 15 shows the test dataset (i.e., Given Sequential Patterns) and the system's corresponding output (i.e., LLM-generated Task Names), as well as its performance measured by Exact Match Accuracy (EMA) and Semantic Match Accuracy (SMA).

Table 15. LLM's task clustering and labeling performance for diverse apps measured by exact match accuracy (EMA) and semantic match accuracy (SMA)

Apps	Given Sequential Patterns	Ground Truth Task Names	LLM-generated Task Names	EMA	SMA
Daum mail	⟨DM.MA→DM.WA⟩	Email reading task	Email reading task, Mail reading task	95.0	96.5
	⟨DM.WA→DM.MA⟩	Email writing task	Email writing task, Mail writing task	95.5	96.0
	⟨DM.MA→DM.WA→DM.MA⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	98.0	99.5
	⟨DM.WA→DM.MA→DM.WA⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	94.0	99.5
Microsoft Outlook	⟨OL.CA→OL.CAWF⟩	Email reading task	Email reading task, Mail reading task	95.0	95.5
	⟨OL.CAWF→OL.CA⟩	Email writing task	Email writing task, Mail writing task	95.5	96.5
	⟨OL.CA→OL.CAWF→OL.CA⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	99.0	99.0
	⟨OL.CAWF→OL.CA→OL.CAWF⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	95.5	99.0
Yahoo email	⟨YH.MPPA→YH.MCA⟩	Email reading task	Email reading task, Mail reading task	95.5	96.0
	⟨YH.MCA→YH.MPPA⟩	Email writing task	Email writing task, Mail writing task	96.0	97.0
	⟨YH.MPPA→YH.MCA→YH.MPPA⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	99.5	99.5
	⟨YH.MCA→YH.MPPA→YH.MCA⟩	Email reading & writing task	Email reading & writing task, Mail reading & writing task	96.0	99.5
Daum	⟨DA.MA→DA.SA→DA.BA⟩	Web browsing task	Web browsing task	100.0	100.0
Microsoft Edge	⟨MS.MSA→MS.AITSA→MS.BA⟩	Web browsing task	Web browsing task	100.0	100.0
Baidu	⟨BD.MA→BD.LSA→BD.WV⟩	Web browsing task	Web browsing task	100.0	100.0
Naver Line	⟨NL.MA→NL.CHA⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	100.0	100.0
	⟨NL.CHA→NL.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨NL.MA→NL.CHA→NL.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨NL.CHA→NL.MA→NL.CHA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
Whats app	⟨WA.HA→WA.C⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	100.0	100.0
	⟨WA.C→WA.HA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨WA.HA→WA.C→WA.HA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨WA.C→WA.HA→WA.C⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
WeChat	⟨WC.LUI→WC.CUI⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	99.5	99.5
	⟨WC.CUI→WC.LUI⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨WC.LUI→WC.CUI→WC.LUI⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨WC.CUI→WC.LUI→WC.CUI⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
Signal	⟨SG.MA→SG.CA⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	98.5	99.5
	⟨SG.CA→SG.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	99.0	99.5
	⟨SG.MA→SG.CA→SG.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	99.0	99.5
	⟨SG.CA→SG.MA→SG.CA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	99.0	99.5
Telegram	⟨TG.LA→TG.CA⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	99.5	99.5
	⟨TG.CA→TG.LA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨TG.LA→TG.CA→TG.LA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨TG.CA→TG.LA→TG.CA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
Facebook MSG	⟨FB.MA→FB.TVA⟩	Messenger chat list browsing task	Messenger chat list browsing task, Messaging chat list browsing task	99.5	99.5
	⟨FB.TVA→FB.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨FB.MA→FB.TVA→FB.MA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0
	⟨FB.TVA→FB.MA→FB.TVA⟩	Messenger chatting task	Messenger chatting task, Messaging chatting task	100.0	100.0

Abbreviations:
EMA = Exact Match Accuracy,
DM.MA = `net.daum.android.mail.list.MailListActivity`,
DM.WA = `net.daum.android.mail.write.WriteActivity`,
OL.CA = `com.acompli.acompli.CentralActivity`,
OL.CAWF = `com.microsoft.office.outlook.compose.ComposeActivityWithFragment`,
YH.MPPA = `com.yahoo.mail.ui.activities.MailPlusPlusActivity`,
YH.MCA = `com.yahoo.mail.flux.ui.MailComposeActivity`,
GM.MAG = `com.google.android.gm.ui.MailActivityGmail`,
GM.CAG = `com.google.android.gm.ui.ComposeActivityGmail`,
BD.MA = `com.baidu.searchbox.MainActivity`,
BD.LSA = `com.baidu.browser.search.LightSearchActivity`,
BD.WV = `com.baidu.webkit.sdk.WebView`,
MS.MSA = `com.microsoft.sapphire.app.main.MainSapphireActivity`,
MS.AITSA = `com.microsoft.sapphire.app.search.autosuggest.activity.AIToolsSuggestActivity`,
MS.BA = `com.microsoft.bing.com.microsoft.sapphire.app.browser.BrowserActivity`,
SMA = Semantic Match Accuracy,
DA.MA = `net.daum.android.daum.ui.main.MainActivity`,
DA.SA = `net.daum.android.daum.features.entrypage.SearchActivity`,
DA.BA = `net.daum.android.daum.browser.BrowserActivity`,
NL.MA = `jp.naver.line.android.activity.main.MainActivity`,
NL.CHA = `jp.naver.line.android.activity.chathistory.ChatHistoryActivity`,
WA.HA = `com.whatsapp.home.ui.HomeActivity`,
WA.C = `com.whatsapp.Conversation`,
WC.LUI = `com.tencent.mm.ui.LauncherUI`,
WC.CUI = `com.tencent.mm.ui.chatting.ChattingUI`,
SG.MA = `org.thoughtcrime.securesms.MainActivity`,
SG.CA = `org.thoughtcrime.securesms.conversation.v2.ConversationActivity`,
TG.LA = `org.telegram.ui.LaunchActivity`,
TG.CA = `org.telegram.ui.ChatActivity`,
FB.MA = `com.facebook.messenger.neue.MainActivity`,
FB.TVA = `com.facebook.messenger.threadview.ThreadViewActivity`

C SHIRBT-MMF: Feature Extraction Process

We extracted usage time, usage frequency, and typing behavior metrics from SHIRBT, total app usage, and categorized app usage within DLV 2–5 as shown in Figure 1. Since app units were not considered in DLV 1 in Table 1, which pertains to smartphone usage, specific typing behavior within a particular app could not be captured; therefore, only usage time and usage frequency were extracted. In the ML-SPM process, we used only the *WindowStateChanged* event (i.e., within-app UI state elements such as activity, view, layout, dialog) to extract SHIRBT as shown in Section 3.1. In the feature extraction process, we additionally incorporated the *NotificationStateChanged* event and *ViewTextChanged* event to extract quantitative metrics related to reaction time to notifications and typing behavior as explained in Section 3.3. To distinguish each event instance type, we represented *WindowStateChanged*, *NotificationStateChanged*, and *ViewTextChanged* as e^{wc}, e^{nt}, and e^{tc}, respectively. In DLV 2–3, we defined the start events of categorized and overall foreground app usage as e^{ca} and e^{oa}, respectively. In DLV 1, we defined the events of the screen on/off sessions as the screen on (i.e., start event) e^{so}, screen unlock (i.e., end event) e^{su}, and screen off (i.e., end event) e^{sf}. To explain the detailed feature extraction process, we defined the timestamp of an event instance e as t. For example, the timestamp of e^{wc} can be denoted as t^{wc}.

C.1 Usage Time Metrics Extraction

As shown in Table 9, we extracted Task Completion Time (TCT), View Transition Time (VTT), and Notification Response Time after App launching (NRTA) from *WindowStateChanged* events (i.e., e^{wc}) composing the SHIRBT, App Usage Duration Time (AUD) and NRTA from overall/categorized foreground app usage start events (i.e., e^{oa}, e^{ca}), and Smartphone Usage Duration Time (SUD) and Screen Unlocking Time (UKT) from smartphone usage events (i.e., e^{so}, e^{sf}, e^{su}). The formulas for calculating duration time (TCT, AUD, SUD, UKT), VTT, and NRTA from these events are as follows.

$$\text{Duration Time} = t^{\text{end}} - t^{\text{start}} \tag{3}$$

$$\text{VTT} = \sum_{j=1}^{n} (t^{\text{wc}}_{j+1} - t^{\text{wc}}_{j}), \quad \text{NRTA} = t^{\text{start}} - t^{\text{nt}} \tag{5}$$

Where t^{start} and t^{end} refer to the start and end timestamp of the events, respectively, and t^{nt} refers to the timestamp of e^{nt} immediately preceding e^{start}.

C.2 Usage Frequency Metrics Extraction

We extracted Task Frequency (TF) and View Transition Frequency (VTF) from e^{wc}, App Usage Frequency (AUF) from e^{oa} and e^{ca}, and Smartphone Usage Frequency (SUF) and Screen Unlocking Frequency (UKF) from e^{so}, e^{sf}, and e^{su}, as shown in Table 9. The usage frequency (TF, AUF, SUF, UKF) is the number of start events (i.e., e^{wc}_1, e^{ca}_1, e^{oa}_1, e^{so}) and e^{su} that occurred in a day. VTF is the product of the length of the sequential pattern that constitutes the extracted SHIRBT and the total number of sequential patterns in a day.

C.3 Typing Behavior Metrics Extraction

As shown in Table 9, we extracted generic typing such as Number of Typed Characters (NTC), Number of Typed Backspaces (NTB), Number of Typed Keystrokes (NTK), Typing Duration Time (TDT) and calculated typing such as Keystrokes per Character (KSPC), Characters per Second (CPS), Keystrokes per Second (KPS), Seconds per Character (SPC), and Second per Keystroke (SPK) from *ViewTextChanged* event within DLV 2–5. The *ViewTextChanged* event e^{tc}, representing user interactions, consists of three types of key events: ba representing backspace, ch representing all characters entered except backspace, and ke representing all keystrokes, including backspace. We extracted only e^{tc} (i.e., ba, ch, and ke) that are located between e^{oa} and e^{ca} until their next event, as well as ba, ch, and ke events that are positioned between the start event (i.e., e^{wc}_1) and end event (i.e., e^{wc}_n) of the sequential pattern. Among the extracted e^{tc} events, the i-th event can be represented as e_i^{tc}, and its corresponding timestamp can be denoted as t_i^{tc}.

The generic and calculated typing metrics are calculated as follows.

$$\text{Generic Typing} = \begin{bmatrix} e^{wc} \\ e^{ca} \\ e^{oa} \end{bmatrix} \cdot \left[\sum_{j=1}^{n} ba_j, \sum_{j=1}^{n} ch_j, \sum_{j=1}^{n} ke_j, t_n^{tc} - t_1^{tc} \right] \tag{6}$$

$$\text{KSPC} = \begin{bmatrix} e^{wc} \\ e^{ca} \\ e^{oa} \end{bmatrix} \cdot \left[\frac{\sum_{j=1}^{n} ke_j}{\sum_{j=1}^{n} ch_j} \right] \tag{7}$$

$$\text{CPS, KPS} = \begin{bmatrix} e^{wc} \\ e^{ca} \\ e^{oa} \end{bmatrix} \cdot \left[\frac{\sum_{j=1}^{n} ch_j}{t_n^{tc} - t_1^{tc}}, \frac{\sum_{j=1}^{n} ke_j}{t_n^{tc} - t_1^{tc}} \right] \tag{8}$$

$$\text{SPC, SPK} = \begin{bmatrix} e^{wc} \\ e^{ca} \\ e^{oa} \end{bmatrix} \cdot \left[\frac{t_n^{tc} - t_1^{tc}}{\sum_{j=1}^{n} ch_j}, \frac{t_n^{tc} - t_1^{tc}}{\sum_{j=1}^{n} ke_j} \right] \tag{9}$$

Where each n represent the total number of each e^{tc} (i.e., ba_j, ch_j, and ke_j) events corresponding to e^{wc}, e^{ca}, and e^{oa} that enclose the entire e^{tc}, respectively. t_1^{tc} and t_n^{tc} represent the timestamp of start and end e^{tc}, respectively.

C.4 Statistics Calculation

Since usage frequency serves as a daily feature sample by itself, we calculated statistics (e.g., average, median, min, max, sum) from other metrics to derive 4–5 daily feature samples each. From duration time, NRTA, and generic typing, we calculated five statistics, while from VTT and calculated typing, we calculated four statistics excluding the sum. The formulas below describe how to calculate statistics from multiple instances of the derived metric m_i, where i represents the index of each extracted instance within a day.

$$\text{Average} = \frac{1}{k} \sum_{i=1}^{k} m_i, \quad \text{Sum} = \sum_{i=1}^{k} m_i \tag{11}$$

$$\text{Min} = \min\{m_i \mid 1 \le i \le k\}, \quad \text{Max} = \max\{m_i \mid 1 \le i \le k\} \tag{13}$$

$$\text{Median} = \frac{1}{2} \left(m_{\left(\frac{k}{2}\right)} + m_{\left(\frac{k}{2}+1\right)} \right) \text{ if } k \text{ is even, else } m_{\left(\frac{k+1}{2}\right)} \tag{14}$$

Where k is the total number of m_i extracted per day for each SHIRBT, app category, overall category, or screen on/off session.