Noname manuscript No.
(will be inserted by the editor)

Energy-Efficient Prediction of Smartphone Unlocking

Chu Luo · Aku Visuri · Simon Klakegg · Niels van Berkel · Zhanna Sarsenbayeva · Antti Möttönen · Jorge Goncalves · Theodoros Anagnostopoulos · Denzil Ferreira · Huber Flores · Eduardo Velloso · Vassilis Kostakos ·

Received: date / Accepted: date

Abstract We investigate the predictability of the next unlock event on smartphones, using machine learning and smartphone contextual data. In a two-week field study with 27 participants, we demonstrate that it is possible to predict when the next unlock event will occur. Additionally, we show how our approach can improve accuracy and energy efficiency by solely relying on software-related contextual data. Based on our findings, smartphone applications and operating systems can improve their energy efficiency by utilising short-term predictions to minimise unnecessary executions, or launch computation-intensive tasks, such as OS updates, in the locked state. For instance, by inferring the next unlock event, smartphones can pre-emptively collect sensor data or prepare timely content to improve the user experience during the subsequent phone usage session.

Keywords Smartphones · Machine learning · Sensors · Context-awareness

1 Introduction

Predicting when a phone will be unlocked in an accurate and energy-efficient manner is crucial to both resource management and user experience. Literature makes a distinction between glancing at the phone's lock screen [3] versus actually unlocking the device [5]. We are interested in predicting the latter, *i.e.* when users actually choose to unlock their device for further use. Although a plethora of

C. Luo, N. van Berkel, Z. Sarsenbayeva, J.Goncalves, E. Velloso, V. Kostakos
The University of Melbourne, Australia
C. Luo E-mail: CHUL3@student.unimelb.edu.au
V. Kostakos E-mail: vassilis.kostakos@unimelb.edu.au
A. Visuri, S. Klakegg, A. Möttönen, D. Ferreira
University of Oulu, Finland
T. Anagnostopoulos
University of West Attica, Greece
H. Flores
University of Helsinki, Finland

work contributes to understanding phone usage patterns, little work has considered techniques to predict smartphone unlock events. Such predictions can benefit smartphones in a number of ways, such as:

1. *trigger timely data acquisition*: prediction of upcoming unlocking events allows mobile applications, especially lockscreen applications, to prepare timely content which users can see right after unlocking. This can directly improve the user experience on the device. A typical use case is lockscreen-based crowdsourcing, which presents questions with pictures or icons for users to answer via various unlock gestures. Given a prediction of the next upcoming unlocking event, smartphones can, in advance, connect to the Internet to download timely questions and their media files. Without this prediction, the user experience may suffer when users see obsolete questions or have to wait for the newest content from the ad-hoc downloading.
2. *disable unnecessary executions*: given a prediction that the phone is likely to remain locked for a certain time, the phone can suspend the execution of applications that continuously prepare content. For example, mobile crowdsourcing applications, such as Truong et al.'s, collect sensor data and download online content every few minutes[43]. The continuous execution of such applications consumes a significant amount of power, which can prove detrimental to the user experience [33] .
3. *schedule computation-intensive tasks*: when predicting that the phone is likely to remain locked, the operating system can launch computation-intensive background tasks, such as data synchronisation or updates, which would have a negative impact on the user experience during phone use. A typical example is the OS update which often involves hundreds of megabytes of data to download, and a long reboot. During an OS update, smartphone users must wait for the completion without any possibility for interaction. Since untimely schedules of such computation-intensive tasks greatly degrade user experience, Apple's iOS 10 engineers have publicly highlighted that scheduling OS updates is a challenging problem [29].

Crucially, the energy cost to achieve such predictions must be sufficiently low, otherwise making these predictions will defeat their very purpose. Therefore energy efficiency is an essential requirement in the selection of smartphone context features and the prediction process.

We hypothesise that by analysing smartphone context we can develop a learning model to predict the next unlock event. We intend to make this prediction regardless of whether the user actively unlocks their phone, or the user is notified to unlock it by a notification. In a 2-week field study, we used the AWARE framework [17] to collect phone usage and context from 27 users during their daily activities. Our analysis is multi-faceted, and a key tradeoff that we investigate is between prediction accuracy and time windows. Overall, our work makes these contributions:

1. We present an energy-efficient approach to predict the next unlocking event on a smartphone, by considering smartphone unlock time prediction as either a regression problem or a binary classification.
2. We identify which features are the strongest indicators of upcoming unlock events, and show that software-based context data be more useful predictors than hardware sensors.

3. We describe and discuss the trade-off between the accuracy of positive and negative predictions in the selection of time windows for this classification.
4. We evaluate two real-world deployment scenarios: using personalised models and cold start.

2 Related Work

Predicting when applications will be launched helps to reduce launch time and improves efficiency, particularly for frequent users [45,39]. Shin et al. dynamically presented application icons on the launch screen by predicting application usage through contextual data with accurate predictions in nearly 90% of cases [39]. Lee et al. reduced the size of contextual data needed for such predictions by integrating a feature selection algorithm [31].

These techniques focus on either expanding the functionality of the lock screen or predicting interactions following the unlock event. No study has investigated the accuracy of predicting when the smartphone will be unlocked.Predicting the time when users will unlock their phones is beneficial for applications related to phone unlocking, such as smart notifications [38], music recommendation, and auto-execution services [31].

2.1 Smartphone Usage Patterns

Several factors impact phone usage patterns, including time, location, and mental state. Time is an important measure in previous studies of smartphone usage patterns. In a longitudinal study, Yan et al. showed that most phone interactions are very short, 50% of interactions last less than 30 seconds, and 90% last less than 4 minutes [46]. Approximately 40% of all application uses consist of short bursts of up to 15 seconds, particularly when the user is at home and alone—what Ferreira et al. call *micro-usage* [16]. Falaki et al. found that phone interactions happen very frequently (10-200 times per day), are short (10-250 seconds), and involve multiple applications (10-90) [13]. McGregor et al. identified three usage patterns in a study with 15 iPhone users: micro-breaks, digital knitting, and mobile reading [8]. Van Berkel et al. further analysed smartphone usage sessions and recognised that multiple micro-breaks can combine into longer combined sessions [5].

In terms of correlation between phone usage and user location, Verkasalo [44] highlights that more than half of the time spent on smartphones is at home; over 25% on the move; and the remainder at work. In a case study of a Chinese city, Yuan et al. found that mobile phone usage correlates with human travel behaviour [47]. One important finding is that when users make more phone calls, their average movement radius increases. Furthermore, based on a smartphone usage dataset from 77 participants over a 9-month period, Do et al. report that application usage is strongly correlated with locations [12]. For example, voice calls are frequently made at work. In contrast, clock application use occurs very seldom at work.

In addition, researchers found that phone usage is impacted by mental states. For example, several studies state that when people feel bored, mobile phones commonly serve as means to kill time [8,34,36]. Other studies found links between

phone usage and compulsive anxiety [27], depressive symptons, higher interpersonal anxiety, and lower self-esteem [22].

Although these previous studies illuminate various aspects of smartphone usage patterns, there is little insight into what contextual factors lead to the user actually unlocking their phone.

2.2 Context-Aware Services on Smartphones

With embedded sensors and increasingly capable processors, modern smartphones act as more than a communication tool. New applications for e-health, environmental monitoring and transportation benefit from the collection of big sensor data on smartphones [30]. Particularly, providing context-aware services to smartphone users is of increasing interest to the HCI and UbiComp community. Studies have repeatedly confirmed the feasibility of inferring user intent using contextual data on smartphones. For example, predicting whether users are available to answer an incoming call[35], how responsive they are to instant messaging [2,37] and when to best interrupt users with notifications [18,38].

However, most of these context-aware services are battery-intensive as they rely on continuous sensing [33,32], leading to the need for daily charging [14,15]. The literature points to alternative ways of using sensor data, such as relying on low-energy sensors wherever possible [4], and avoiding energy-intensive sensors like GPS [48]. To reduce the power cost from context-aware services, Lu et al. proposed the Jigsaw continuous sensing engine [33]. Jigsaw adaptively controls accelerometer, microphone and GPS sensors according to phone sensing environments. Similarly, Chon et al. presented a smartphone sensing manager SmartDC that monitors, learns, and predicts user mobility and location to conduct adaptive energy-efficient duty cycling [11].

In spite of these efforts, prior work has not attempted to achieve energy efficiency by minimising irrelevant executions while the user keeps the phone in the locked state for a long time. If a smartphone application can predict when it will be unlocked next, it is easier for other applications to prepare relevant content and schedule tasks for the next session, such as delaying or dismissing context-aware notifications, downloading timely content, or changing phone settings (e.g., slowing down data synchronisation and enabling power saving mode).

3 Methodology

We conducted an in-the-wild study with 27 Android smartphone users (16M/11F) recruited through mailing lists of at our University. The data collection lasted for two weeks. Most participants were students or had an academic background. Participants were young (M = 24.7, SD = 4.32), and from various academic backgrounds with 14 participants from engineering, 6 from humanities, 2 from business, 1 from natural sciences, and three preferred to not say. Most participants were local (20) to our city, while seven were foreign.

After we explained the data collection mechanism and study process, we installed the software on each participant's phone. We then instructed participants

Table 1: Features and data types extracted from the smartphone sensors and other data sources used in data collection.

Data Source	Description	Feature and Data Type
Activity	Physical activity	e.g. walking, running and being in vehicle (value within a categorical set)
Application	Foreground application	The number of foreground applications used in last session (integer)
Data Traffic	Data upload and download transmission	Since last session, WiFi data upload (integer); WiFi data download (integer); Cellular data upload (integer); Cellular data download (integer);
Demographics	Gender	Gender (value within a categorical set)
Light	Ambient light (lux)	Ambient light lux (integer)
Location	Location on, off and mode (GPS, network)	GPS availability (Boolean); Network location availability (Boolean);
Pedometer	Steps travelled	Step count since last session (integer)
Proximity	Coverage on phone	Proximity (Boolean)
Screen Events	Screen on, off and unlocking	Duration of last session (integer); time since last session (integer)
Time	Timestamp of phone clock	Minute of hour (integer); hour of day (integer); day of week (integer)
WiFi	WiFi on, off and the number of nearby spots	WiFi availability (Boolean); WiFi spots nearby (integer);

to use their smartphones as they normally would in their everyday life. Our software did not require any active input from participants, nor did it notify them in any way. Our software logged smartphone usage events, including unlocking events and contextual data from sensors. The application worked as a plugin of the AWARE framework, an open-source middleware to capture contextual data on mobile devices [17]

3.1 Data Collection and Features

Our plugin constantly collected contextual data in the background. Table 1 describes the features extracted from the sensors and other data sources. We consider a *session* of phone usage as a continuous interaction period from the moment of unlocking the smartphone to the moment of the subsequent locked/power off state. To minimise power consumption, only low-frequency proximity (200ms/sample), light (5s/sample), location (180s/GPS sample, or 300s per network sample when GPS unavailable) and WiFi (60s/scan) sensors are logged. We also used Google's Activity Recognition API (6s/sample) [20] and Android's Pedometer API [21] to obtain users' activities and steps, computed from accelerometer data. Based on such contextual data, we extracted 18 features. In addition, we also considered gender as a feature, as reported by participants.

Because the characteristics of a previous usage session might be informative as to the timing of the next unlocking event, we collect application usage as provided

by the operating system. Finally, data traffic reveals the intensity of networked activities, as measured by two hardware modules (WiFi and Cellular antenna).

3.2 Analysis

Our analysis first considers unlock time prediction as a regression problem: given the user context, the algorithm should predict the timing of when the user will unlock the phone. Without knowing the linearity of the relation, we compare the performance of two regression algorithms: multiple linear regression as a linear approach (it serves as a baseline of regression), and Random Forests as a nonlinear approach (using Weka's default parameters, i.e., Random Forests with 100 trees and $\lfloor \log_2 m \rfloor + 1$ features, where m is the number of initial features). For Random Forests, using $\lfloor \log_2 m \rfloor + 1$ features avoids high complexity of the model and reduce overfitting. Considering that there are only 19 features in our dataset ($\lfloor \log_2 19 \rfloor + 1 = 5$ features per tree), 100 trees in Random Forests are sufficient. We also evaluate the performance loss incurred in using solely low-power software-related features as compared to the full feature set. Finally, we compare the performance of personalised models to cold start models (i.e. models trained on other users' data) [6].

We also attempt to use a classifier approach: given the user context, the classifier should predict whether the user will unlock the phone within the next x minutes. Similar to previous work employing different machine learning algorithms on smartphones [35,38], we compared four widely used classifiers to run 10-fold cross-validation tests: Naive Bayes, J48 Decision Tree, Random Forests and Support Vector Machine (SVM) with the Radial Basis Function (RBF) kernel (with normalised samples). The parameters were left to their default values in Weka for simplicity, i.e., Random Forests with 100 trees and $\lfloor \log_2 m \rfloor + 1$ features (m is the number of initial features); SVM with the RBF kernel ($\gamma = \frac{1}{m+1}, C = 1$).

In summary, our analysis empirically investigates the following questions:

1. How well do machine learning algorithms perform?
2. What are the most useful features in the prediction?
3. How does performance change using low-power software-related data sources?
4. How does a personalised model perform on each user who has provided training data?
5. How does a general model trained by previous users' data perform on a new user who has not provided training data (the new user cold start problem [6])?
6. How much energy does the prediction consume on a real device?

3.3 Measures

For regression, we used three commonly used measures: coefficient of determination (R^2), mean absolute error (MAE) and square root of mean squared error (RMSE). For the importance of each feature in the regression model, we used two feature importance measures provided by Random Forests: increased mean squared error (IncMSE) and increased node purity (IncNodePurity). For each feature, IncMSE is computed by comparing the difference of mean squared error in predictions after permuting this feature. IncNodePurity corresponds to the increase in node

Table 2: Confusion matrix for classification analysis of the two classes.

	Actual Unlock (Class "Yes")	Actual Lock ("No")
Predicted Unlock (Class "Yes")	TP	FP
Predicted Lock (Class "No")	FN	TN

Table 3: Recall and Precision definitions of the two classes.

	Recall	Precision
Unlock (Class "Yes")	$\frac{TP}{TP+FN}$	$\frac{TP}{TP+FP}$
Lock (Class "No")	$\frac{TN}{TN+FP}$	$\frac{TN}{TN+FN}$

purity of having each feature within the trees related to the mean squared error. Hence, a high IncNodePurity of a feature means that this feature significantly helps the algorithm to reduce mean squared error. The details of these measures are illustrated by Genuer et al. [19].

In the classification models, because of class imbalance, we report ROC area, precision, and recall, instead of accuracy, as recommended by Kostakos et al. [28]. All results are reported using a 10-fold cross-validation on the entire dataset. In our data we separately consider the two classes: "no" (the cases where the user did not unlock the phone within x minutes), and "yes" (where the user did unlock the phone within x minutes). Table 2 shows the confusion matrix for classification analysis on the two classes. Based on the confusion matrix, Table 3 shows recall and precision definitions of the two classes.

For the feature importance of classification, we used Mean Decrease Accuracy (MDA) and Mean Decrease Impurity (MDI). MDA is measured by randomly permuting features in the out-of-bag samples, while MDI (also known as Mean Decrease Gini if the Gini index is used as in our analysis) is the average weighted impurity decreases for all nodes containing a specific feature across all trees [7]. Specifically, we used 3 types of MDA to investigate the influence of removing each feature on both classes, as well as the overall prediction. The higher the MDA value is, the more importance a feature has. In practice, the scenario may require high accuracy for only one class.

In contrast, MDI does not measure feature importance for each class. MDI provides an overall assessment of each feature's importance based on the assumption that if a feature has high usefulness, it tends to divide nodes with multiple classes into nodes with single classes. However, this assumption may result in bias. Usually, feature importance analysis should use a combination of different measures to avoid the bias from a single measure [41].

4 Results and Optimisation

We collected 7,472,609 entries of raw user data, from which we sampled 175,684 segments of phone usage. We used Weka [23] to analyse these samples in both the regression and classification problems.

Table 4: Regression results of the entire dataset in coefficient of determination (R^2), mean absolute error (MAE) (in minutes) and square root of mean squared error (RMSE) (in minutes).

Dataset	Regression type	R^2	MAE	RMSE
All sensors	Linear Regression	0.06	122.59	263.96
All sensors	Random Forests	0.98	14.24	48.09
Software sensors only	Linear Regression	0.03	124.87	268.11
Software senors only	Random Forests	0.98	12.50	45.85

Table 5: Regression results of personalised models and the model of cold start in coefficient of determination (R^2), mean absolute error (MAE) (in minutes) and square root of mean squared error (RMSE) (in minutes).

Bootstrap approach	Regression type	R^2	MAE	RMSE
Personalised	Linear Regression	0.08	117.16	181.51
Personalised	Random Forests	0.94	8.89	24.44
Cold Start	Linear Regression	0.04	168.80	251.87
Cold Start	Random Forests	0.96	150.05	258.40

4.1 Regression Approaches

Table 4 shows the regression results using 3 commonly used measures: coefficient of determination (R^2), mean absolute error (MAE) and square root of mean squared error (RMSE). Times are displayed in minutes. Results show that for both algorithms, removing hardware features does not lead to a significant performance loss. Also, the Random Forests algorithm outperforms linear regression in both accuracy and dispersion.

Table 5 shows the regression results for the personalised and cold start models. Although Random Forests significantly outperformed linear regression in personalised models, the accuracy and dispersion are similarly poor for the cold start model.

To quantify the relative importance of each feature when using regression, we used two feature importance measures provided by Random Forests: IncMSE and IncNodePurity. The unit of IncMSE and IncNodePurity is the same as the regression process: min^2. Table 6 shows the importance of all the 19 features using these two measures. The 5 features with the highest IncMSE are: WiFi data download, duration of last session, time since last session, WiFi data upload, number of foreground apps used in last session. Three of these features come from software sensors. The 5 features with the highest IncNodePurity are: duration of last session, time since last session, WiFi data upload, WiFi data download, hour of day. Again, 3 of these features come from software sensors. The difference of top features across the two measures is little, meaning that the results have high robustness.

Table 7 shows the correlation (Spearman's rank correlation coefficients) between each feature and the time before next unlocking event. We only considered numerical features in this analysis (10 out of 19 features). We found that all 10 showed a significant ($\alpha = 0.05$) correlation with the time before the next unlocking event, although relations were mostly weak. The time since last session, a software-related feature, had the strongest relation ($r_s = 0.411$). From the aspect of correlation, the results reflected the different importances of each factor. It is

Table 6: When using Random Forests regression, the importance (in min^2) of all the features is estimated by two measures (IncMSE and IncNodePurity)

Feature	IncMSE	IncNodePurity
Time since last session	65104.8	13.51 $\times 10^8$
Duration of last session	68947.3	23.67 $\times 10^8$
Hour of day	51155.7	10.17 $\times 10^8$
Activity	7382.5	1.12 $\times 10^8$
Minute of hour	1970.9	1.28 $\times 10^8$
Day of week	32953.2	8.31 $\times 10^8$
Proximity	17930.7	2.75 $\times 10^8$
Ambient light	34608.7	3.73 $\times 10^8$
Cellular data download	24750.8	4.99 $\times 10^8$
Cellular data upload	32317.6	7.28 $\times 10^8$
WiFi data download	78213.3	11.04 $\times 10^8$
WiFi data upload	64162.6	11.09 $\times 10^8$
Step count since last session	19901.4	2.13 $\times 10^8$
Number of foreground apps used in last session	55934.8	5.51 $\times 10^8$
GPS availability	13162.1	1.22 $\times 10^8$
WiFi spots nearby	13057.4	2.67 $\times 10^8$
Network location availability	10391.2	2.29 $\times 10^8$
WiFi availability	7995.7	1.23 $\times 10^8$
Gender	18829.9	2.31 $\times 10^8$

worth noting that features with weak relations may also be useful if the classification or regression method can learn from nonlinear data.

4.2 Classification Approaches

When considering the average number of unlocking events our users exhibited per day, the maximum was 163, which is in line with previous studies [13]. Given that phone unlocking events can be rather frequent during the day, and that mobile applications can download sufficiently large files in several minutes via either WiFi or 4G, we decided to also model our problem as a classification rather than regression.

Doing so means that we do not try to answer the question "When will the next unlock happen?", but rather we ask the question "Will the next unlock happen in the next x minutes?" In essence, this is an easier question to answer, but it stil provides utility given that many smartphone operations can complete within a few minutes.

In our analysis, we consider a time period of up to 10 minutes, in 1-minute segments. Therefore, our classifier tries to answer the question: "Will the next unlock happen in the next x minutes?" for x between 1 and 10. Figure 1 using our 175,684 samples to visualise our ground truth: did an unlock actually happen in the next x minutes during the duration of the study? As expected, we observe a class imbalance, especially for short time windows (e.g., 1 min had 163129 "no" instances and 12555 "yes" instances). As the time window increase, there are more

Table 7: Correlation between each feature and the time before next unlocking event, measured by Spearman's rank correlation coefficients.

Feature	p-value	r_s
Time since last session	$< 2.220 \times 10^{-16}$	0.411
Duration of last session	$< 2.220 \times 10^{-16}$	0.024
Hour of day	N/A	N/A
Activity	N/A	N/A
Minute of hour	N/A	N/A
Day of week	N/A	N/A
Proximity	N/A	N/A
Ambient light	$< 2.220 \times 10^{-16}$	-0.182
Cellular data download	1.548×10^{-10}	-0.015
Cellular data upload	0.042	0.005
WiFi data download	$< 2.220 \times 10^{-16}$	0.136
WiFi data upload	$< 2.220 \times 10^{-16}$	0.153
Step count since last session	$< 2.220 \times 10^{-16}$	-0.214
Number of foreground apps used in last session	$< 2.220 \times 10^{-16}$	-0.098
GPS availability	N/A	N/A
WiFi spots nearby	$< 2.220 \times 10^{-16}$	-0.072
Network location availability	N/A	N/A
WiFi availability	N/A	N/A
Gender	N/A	N/A

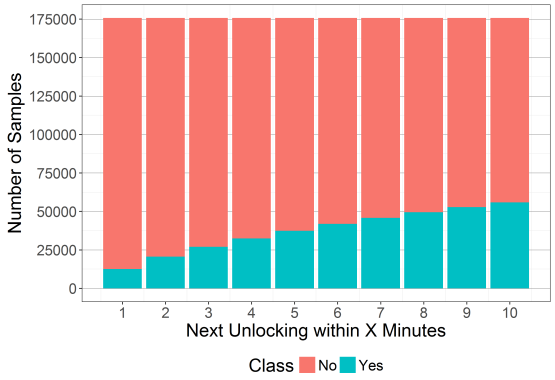

Fig. 1: 175,684 samples of the absence and presence of unlocking events measured by different time windows at different moments during our study.

samples in the class "yes" balancing the two classes (e.g., 10 min has 119831 "no" instances and 55853 "yes" instances).

To better understand the classification performance, we first need to define a baseline classifier. We simply use the statistical distribution of mean times between two sessions (i.e., the idle usage period between two consequitive lock & unlock events). Our baseline classifier outputs positive predictions of the next smartphone unlocking event when the idle usage period is longer than the average value, and vice versa (since the baseline is not an algorithm, it does not give ROC).

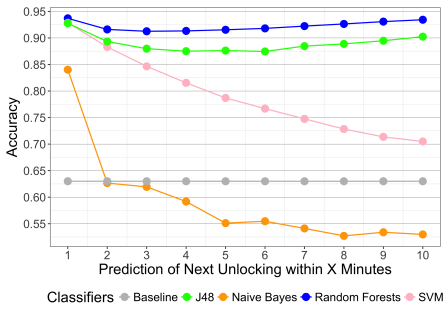

Fig. 2: Comparison between baseline, Naive Bayes, J48 Decision Tree, Random Forests and SVM.

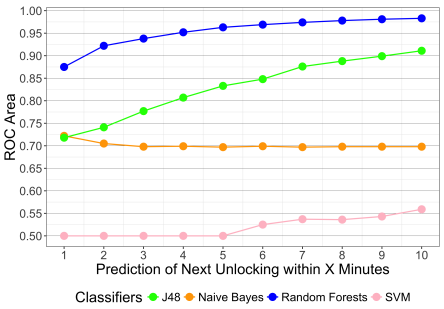

Fig. 3: ROC Areas of Naive Bayes, J48 Decision Tree, Random Forests and SVM.

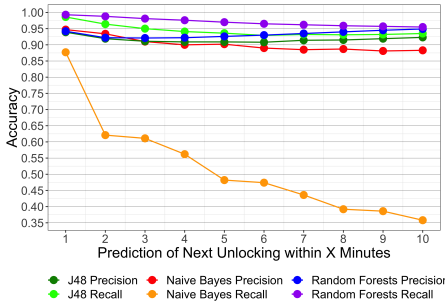

Fig. 4: Precision and recall of Naive Bayes, J48 Decision Tree and Random Forests on class "no"

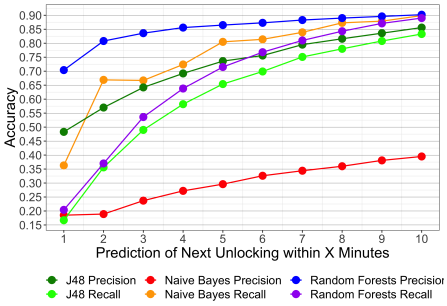

Fig. 5: Precision and recall of Naive Bayes, J48 Decision Tree and Random Forests on class "yes"

Figure 2 depicts the comparison results in overall classification accuracy (i.e., the ratio of correctly classified instances). The baseline classifier's performance is 0.630 regardless of time windows. The accuracy of Random Forests is the highest in all time windows, stably ranging from 0.913 to 0.937.

Figure 3 shows the receiver operating characteristic (ROC) areas of the four classifiers. Here, each classifier is asked to predict whether an unlock event will be observed in the next x minutes (x-axis). Random Forests outperformed other classifiers. Its ROC areas range from 0.875 to 0.983, which show a steady ascending trend as time window increases.

To further investigate the classifiers' performance, we separately consider the two classes: "no" (which represents the cases where the user did not unlock the phone within x minutes), and "yes" (where the user unlocked the phone within x minutes). Figure 4 shows the precision and recall of the three classifiers for the class "no" and Figure 5 for class "yes". In these figures, precision measures the ratio of true positives to all positive predictions on this class. Similarly, recall denotes how many relevant instances are correctly predicted.

For class "no", Random Forests had the highest precision within 0.921 to 0.949. For class "yes" the precision of Random Forests is also the highest among the three classifiers, increasing approximately in a shape of a logarithmic function from 0.704

at 1 min to 0.902 at 10 mins. The recall of Naive Bayes is the highest, significantly outperforming the others from 1 min to 5 mins (growing from 0.363 to 0.899). The difference becomes insignificant in larger time windows.

Since Random Forests had the best and stable performance in our analyses, we only consider this algorithm for further analysis of predicting smartphone unlocking. We actually attempted to apply parameter tuning methods on the RBF kernel for SVM (see Appendix A.1), but due to the large number of possible γ values we could not improve further.

4.2.1 Random Forests Parameter Tuning

Previous work, such as [42], stated that the increase of number of trees in Random Forests can improve the performance, and the default setting of feature number $\lfloor \log_2 m \rfloor + 1$ is optimal (m is the number of initial features). Hence, we attempted to repeat the 10-fold cross-validation on the data of 10 min time window, with different numbers of trees in Random Forests, including increasing and decreasing. As expected, the highest number of trees produced the best performance. The ROC Area grows up from 0.978 at 25 trees to 0.983 at 150 trees; class "no" precision from 0.942 at 25 trees to 0.949 at 150 trees; class "no" recall from 0.952 at 25 trees to 0.955 at 150 trees; class "yes" precision from 0.896 at 25 trees to 0.902 at 150 trees; class "yes" recall from 0.874 at 25 trees to 0.889 at 150 trees.

The results indicate that increasing the number of trees improve the overall performance of Random Forests. The default value of the number of trees in Random Forests is 100, which is already large. Considering limited computing resources on smartphones, it is not necessary to use a larger number, unless the targeted mobile applications have strong need in the high performance of classification.

4.2.2 Optimisation regarding Class Imbalance for Random Forests

From Figure 5, we observed that using a small time window significantly reduced the classification performance of Random Forests for class "yes". This may be because there are fewer data instances within the class "yes" in small time windows, which cause a class imbalance problem [26]. The class imbalance problem represents the challenge in the cases where data instances within one class are significantly more or less than those within other class(es). Algorithms such as Random Forests may perform poorly in predicting the class with an imbalanced number of instances. Similar to previous work, such as [26], handling this problem, we attempted to apply repeated random sub-sampling to improve the performance of Random Forests in small time windows. Repeated random sub-sampling is a popular method for highly imbalanced datasets (e.g., medical records containing a large number of negatives and a several positives) [26].

Figure 6 summarises the process of repeated random sub-sampling for the construction of unlocking prediction models using Random Forests. First, given the training data with N_0 instances of class "no" and N_1 instances of class "yes", it computes NoS, where $NoS = \lfloor \frac{N_0}{N_1} \rfloor$. If $NoS < 3$, it ends the process by assuming that there is no class imbalance problem. Otherwise ($NoS \geq 3$, which means $N_0 \gg N_1$, and at least 3 models can be voters), it randomly divides N_0 instances of class "no" into $N0S$ groups. Then, it trains NoS models where each model is

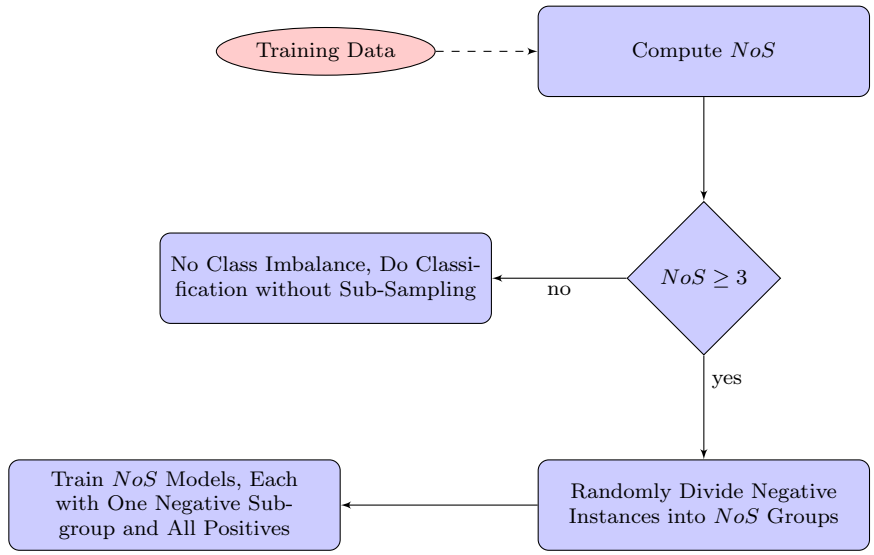

Fig. 6: Training models of Random Forests with sub-sampling against class imbalance. NoS is the ratio of "no" to "yes" instances.

trained by one subgroup of instances from the class "no" and all instances from the class "yes".

Figure 7 summarises the prediction process. The values from all features are sent to all the NoS models. The results from the the NoS models go to a voting system which simply accepts the prediction of the majority. In the case where there are equal numbers of positive and negative predictions from the models, it gives the negative prediction (i.e., class "no"), since the negative instances are the majority in training data.

With the repeated random sub-sampling method, we repeated the 10-fold cross-validation on the data of all time windows using Random Forests. The data sets of time windows from 1 min to 7 mins were considered as imbalanced ($NoS \geq 3$).

Figure 8 compares the process with and without repeated random sub-sampling using the precision and recall measure for the class "no". And Figure 9 shows the results for the class "yes". Overall, repeated random sub-sampling significantly improved the performance for the class "yes" which was challenged by the class imbalance problem. However, repeated random sub-sampling reduced the performance for the class "no" which had the majority of instances. For various mobile applications, the two classes may be of different importance. Developers should use repeated random sub-sampling if their applications need higher performance of unlocking prediction for the class "yes" using small time windows.

4.3 Feature Evaluation in Classification

To quantify the importance of each feature, we compared them using two criteria. First, we used the "select attributes" function of Weka to generate the average ranking of features in 10-fold cross validation. The evaluator used was the classifier

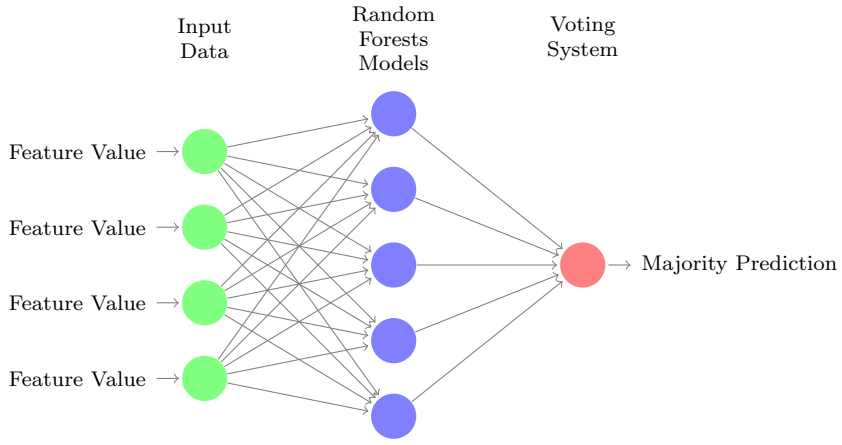

Fig. 7: Using models of Random Forests with sub-sampling to classify samples with class imbalance

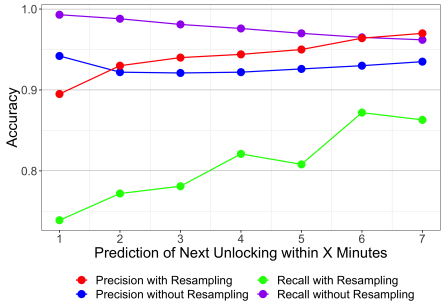

Fig. 8: Precision and recall of Random Forests with and without sub-sampling on class "no"

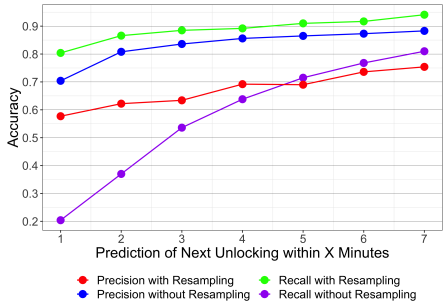

Fig. 9: Precision and recall of Random Forests with and without sub-sampling on class "yes"

attributes subset which measures the accuracy decrease caused by the removal of features from a classifier. Since Random Forests performed the best in the previous analyses, we only consider this classifier. The search method was *Greedy Stepwise* which explores forward or backward in a greedy search through space within feature subsets. Second, we also included two feature importance measures generated by the training process of Random Forests in 10-fold cross validation: Mean Decrease Accuracy (MDA) and Mean Decrease Impurity (MDI).

Since Random Forests gained the highest ROC area in the time window 10 mins, we used this timeframe to maximise the effect of using different feature subsets (also, using this timeframe can minimise the effect of class imbalance which is illustrated in the discussion section).

Table 8 includes the importance of all the 19 features computed using our two methods. According to Weka's "select attributes", the 5 features with the highest average rank are: time since the end of last session, duration of last session, hour of day, activity, and minute of hour. Among them, only activity data is collected from hardware sensors. The 5 features with the highest MDA on the class "no"

Table 8: For the binary classification problem, the importance of all the features calculated by select attributes (average rank) and the two kinds of measures (MDA and MDI) of Random Forests.

Feature	Rank	No-MDA	Yes-MDA	All-MDA	MDI (Gini)
Time since last session	1	0.16681	0.23758	0.18931	11982.5
Duration of last session	2	0.11121	0.11011	0.11086	6669.2
Hour of day	4.3	0.11097	0.12337	0.11491	4931.3
Activity	4.4	0.05428	0.03985	0.04969	2092.5
Minute of hour	5.5	0.03897	0.07236	0.04958	5105.0
Day of week	5.9	0.05578	0.06556	0.05889	2999.3
Proximity	7.5	0.03234	0.02655	0.03050	882.9
Ambient light	8.1	0.08299	0.09787	0.08772	3719.0
Cellular data download	8.8	0.08347	0.08378	0.08357	4173.8
Cellular data upload	10.1	0.08838	0.08568	0.08753	4136.5
WiFi data download	10.3	0.13248	0.12501	0.13010	5094.3
WiFi data upload	12.1	0.13656	0.12556	0.13306	5067.1
Step count since last session	12.3	0.06695	0.07625	0.06991	2776.5
Number of foreground apps used in last session	14.1	0.10863	0.11551	0.11082	3274.9
GPS availability	15.3	0.03025	0.03679	0.03233	773.4
WiFi spots nearby	16.2	0.03636	0.04596	0.03941	2025.4
Network location availability	16.5	0.02753	0.03404	0.02960	752.3
WiFi availability	16.6	0.01905	0.02253	0.02015	524.7
Gender	19	0.06928	0.09512	0.07749	1179.9

are: time since the end of last session, WiFi data upload, WiFi data download, duration of last session, hour of day. Only 2 of them (WiFi data upload, WiFi data download) are hardware-related features. Similarly, the 5 features with the highest MDA on the class "yes" are: time since the end of last session, WiFi data upload, WiFi data download, hour of day, number of foreground apps used in last session. Still, 3 of them are software-related features.

4.4 Classification using only Software-Related Context

To reduce power consumption in prediction, we calculate the prediction accuracy without considering context derived from hardware sensors. Thus, we only selected the following "software-related" features: application usage, screen events, time and gender.

Figure 10 shows the ROC areas of Random Forests using all context and software-related context. Using software-related context only, Random Forests actually achieve a slightly higher ROC area curve (from 0.918 at 1 min to 0.990 at 10 mins) than using all available context (from 0.875 at 1 min to 0.983 at 10 mins). As the time window grows, the performance gap gradually decreases. Figure 11 and 12 show the precision and recall of Random Forests using all context and software-related context for both classes "no" and "yes". Precision and recall actually improves when using software-related context.

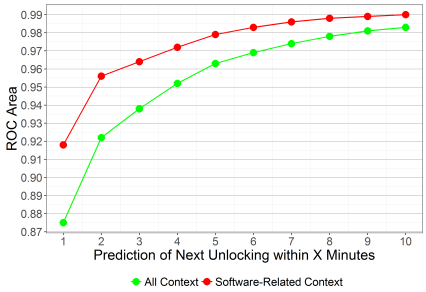

Fig. 10: ROC areas of Random Forests using all context and software-related context.

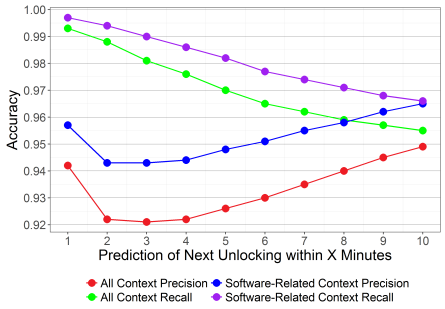

Fig. 11: Class "no": precision and recall of Random Forests using software-related context vs. using all context

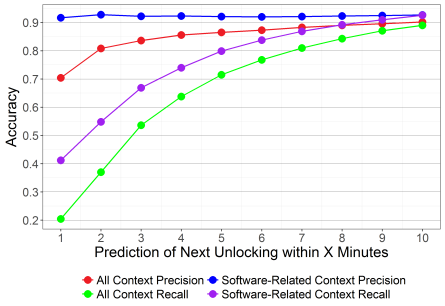

Fig. 12: class "yes": precision and recall of Random Forests using software-related context vs. using all context

4.5 Personalised Model in Classification

To validate our approach under more realistic conditions, we investigated personalised models built by data from each participant. Thus, we personalised the model for each participant using Random Forests and software-related context. We conducted 10-fold cross validation on each participant's own data. Then we computed the average ROC areas across all participants.

Figure 13 depicts the ROC areas of the personalised models. The ROC area of the personalised model grows rapidly as the time window increases from 1 min to 5 mins. For longer time windows, the ROC areas plateau.

Figure 14 shows the precision and recall of the personalised model for the class "no". The precision curve of the personalised model is stably high, between 0.958 and 0.970, in all time windows. In contrast, the recall curve, although on a higher level, follows a descending pattern: reaching the peak on 0.993 in time window at 1 min, then steadily dropping at a slow speed as the time window grows (down to 0.981 at 10 mins).

Figure 15 compares the precision and recall of the personalised model for the class "yes". The precision curve grows steadily from 0.854 to 0.959 in time window from 1 min to 10 mins. Comparatively, the recall curve grow rapidly from the level below 0.5: from 0.421 at 1 min to 0.695 at 3 mins. Then it grows gradually in larger window sizes (up to 0.904 at 10 mins).

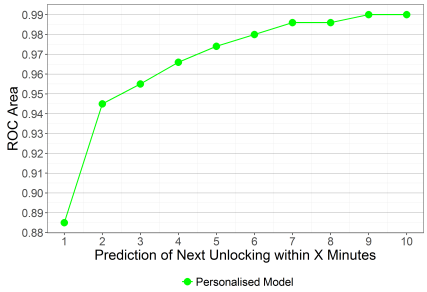

Fig. 13: Using software-related context only, ROC area of the personalised model trained by Random Forests.

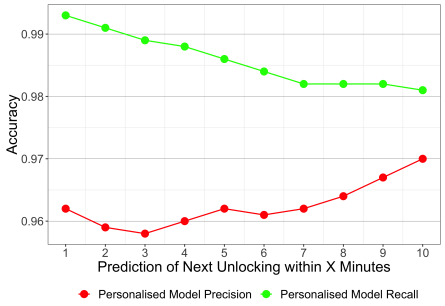

Fig. 14: Class "no": precision and recall of the personalised model trained by Random Forests

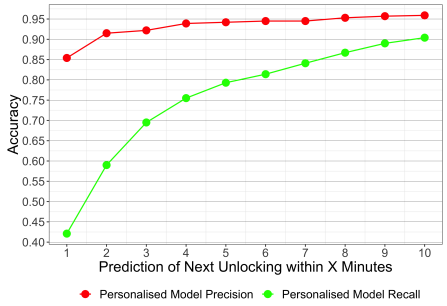

Fig. 15: class "yes": precision and recall of the personalised model trained by Random Forests

4.6 The New User Cold Start Problem

A common scenarios for classification system is the cold-start problem, where we need to make inferences or predictions about a new user who has not provided any information. In this case, our classifier needs to use data collected from other users to make a recommendations for new users. This challenge is called "the new user cold start problem" [6]. In the same manner, our system predicting the next smartphone unlocking event also has to handle this problem. A new user may want to use this system immediately after installation, meaning that there will be no training data from this user for the system to learn and build a personalised model. The system must generate predictions using the model trained by data collected from previous users.

Consequently, we investigated how a cold start of smartphone unlocking prediction performs on a new user. Similar to previous work [6], we conducted a leave-one-out cross validation which iterates through each user by considering the data from other users as the training data, and then validating the model using the data of this user as the testing data. The evaluation conditions are the same as the analysis of personalised models: using Random Forests and software-related context.

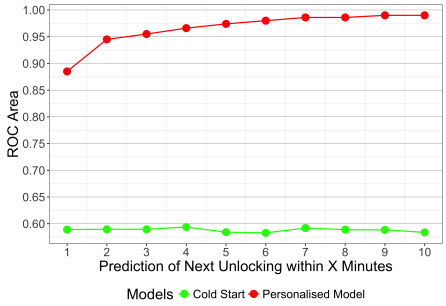

Fig. 16: ROC Areas for our classifier in cold start vs. personalised model.

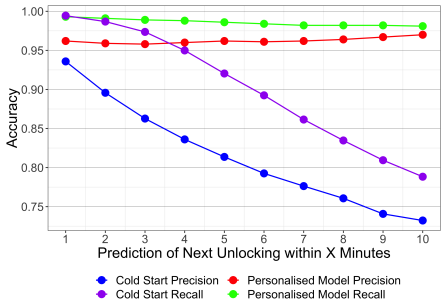

Fig. 17: Class "no": precision and recall of the model in cold start, compared to the personalised model

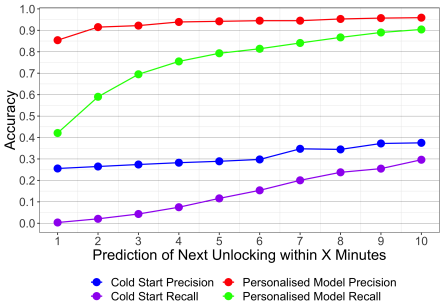

Fig. 18: class "yes": precision and recall of the model in cold start, compared to the personalised model

Figure 16 compares the ROC areas of the model in cold start and the personalised model. Across all time windows, the ROC areas of the model in cold start remains low, ranging from 0.583 to 0.594. In contrast, the ROC areas of the personalised model are significantly higher, growing up steadily as the time window increases.

Figure 17 shows the precision and recall of the model in cold start and the personalised model for the class no. As the time window increases, both the precision and recall curve of the model in cold start drop steadily: precision from 0.936 at 1 min to 0.732 at 10 mins; recall from 0.994 at 1 min to 0.788 at 10 mins. Comparatively, the curves of the personalised model are stably high.

Figure 18 depicts the precision and recall of the model in cold start and the personalised model for the class yes. As the time window increases, both the precision and recall curve of the model in cold start follow a slowly ascending pattern: precision from 0.256 at 1 min to 0.375 at 10 mins; recall from 0.004 at 1 min to 0.296 at 10 mins. The curves of the personalised model also follow an ascending pattern with significantly higher values in different time windows.

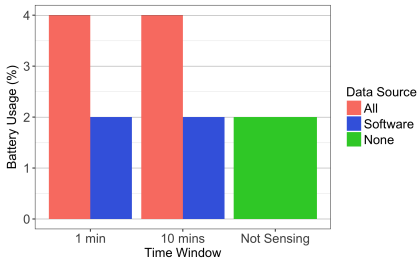

Fig. 19: Energy consumption of our approach on Huawei GR5 2017 in different conditions.

4.7 Energy Consumption

As energy efficiency is a critical requirement of our approach, we quantified the energy consumption on a real smartphone in different conditions. We selected a Huawei GR5 2017 with Android 7.0 as the experiment device, with 3340mAh full battery capacity.

We installed AWARE and our implementation of the unlocking prediction system which is based on the Random Forests classifier in Weka. We compared two sets of contextual data sources (all data sources vs. software-related sources) and two time windows (1 min vs. 10 mins). Also, to provide a baseline, we measured the energy consumption in standby state by disabling the installed applications on the device. The test for each condition ran for 24 hours.

Figure 19 shows the results of the energy tests in different conditions. When using all the data sources, the device used 4% (133.6mAh) of its battery capacity when using a prediction with time window either 1 min or 10 mins. In contrast, the devices used only 2% (66.8mAh) battery when using software-related data sources only for time windows either 1 min or 10 mins. This energy consumption is virtually the same as that of standby state, meaning that using software-related data sources only has negligible impact on smartphone battery.

5 Discussion

Our work has shown that it is possible to predict whether users will unlock their phone in the near future. Our analysis shows that contextual data can be used to make this prediction with varying degrees of confidence. We adopted two independent methods to make this prediction: using regression, and using classification. Classification allows us to ask whether the user is likely to unlock their phone in the near future (up to 10 minutes). This approach is suitable when a short-term prediction is required. Regression enables us to predict the timing when we expect the next unlock to happen. This approach can be useful for longer-term planning. For example, if the classification suggests that the user will not unlock their phone in the next 10 minutes, regression can be used to estimate when the next unlock may take place.

Additionally, our work has investigated the use of software-only sensors, as a means of reducing the power consumptions necessary for prediction. The analysis shows that in fact using software-only sensors improves predictive power. Finally,

our analysis shows that personalised models perform quite well and on par with general models, but in cold-start settings the performance significantly drops.

5.1 Regression Methods

We investigated the effectiveness of regression methods attempting to predict the exact time of the next unlocking event. Our findings show that Random Forests can perform significantly better than multiple linear regression. This indicates that the relation between smartphone usage patterns and phone unlock events are highly nonlinear. We also found that the difference between using all features and using software-related features only was insignificant in this case. Further, a software-related feature, time since last session, had the strongest correlation to the time before the next unlocking event. Using personalised models and software-related features, Random Forests with the regression model can achieve an MAE of 8.89 mins and an RMSE of 24.44 mins. This accuracy is sufficient to schedule computation-intensive tasks, such as OS updates taking half an hour, when the classifier predicts a very long period of idle phone usage.

The strength of regression methods is flexibility, since algorithms can estimate the time when users are likely to unlock their phone next. One exciting possibility for this kind of prediction is the hypothesis that our classifier can be used to predict when users are going to sleep or will wake up. For example, previous work at UbiComp has shown that smartphone usage can be used to detect sleep behaviour [1] with MAE 45 minutes.

That work used screen on-off patterns to build personalized models for retrospectively determining sleep. We believe that our regression model can be used to identify long periods when users are not expected to unlock their phone, therefore identifying times when the users may be sleeping. By its nature, our prediction also identifies the moment in time when users are expected to wake up (and start using their phone), as shown in previous work.

Furthermore, mobile applications can use both classifiers and regression methods to improve user experience, particularly for frequent users [45,39]. When all time windows do not fit the requirement of a specific task (e.g., prediction if there will be 30 mins of idle usage), the smartphone can rely on regression. However, we found that the results generated by a cold start of our regression models tend to be very inaccurate on average. Hence, we suggest avoiding regression methods to launch a cold start prediction of unlocking. For a regression-based cold start of unlocking prediction, future research can investigate the possible strategies to improve the performance.

5.2 Classification Methods

Our study with 27 participants demonstrates that it is possible and practical for smartphones to predict the next unlock event using a machine learning model and contextual data. Given the smartphone unlock time prediction as a binary classification problem, results show that a personalised model trained by Random Forests using software-related contextual data achieves ROC areas from 0.885 at 1 min to 0.99 at 10 mins. For every instance where the user will not unlock the

phone in the next x minutes (labelled as class "no" in our classification), the prediction accuracy of this model ranges from 0.993 at 1 min to 0.904 at 10 mins. Energy-saving mechanisms such as pausing the downloading of timely content and hardware sensor usage can be triggered by our model's predictions that fall in the class "no". With such mechanisms triggered, the amount of saved energy is the same as the prediction accuracy. When the user is going to unlock the phone in next x minutes (labelled as class "yes"), the performance varies greatly from 0.421 at 1 min (without sub-sampling) to 0.981 at 10 mins. Following a prediction in the class "yes", context-aware services on smartphones may launch in the background to generate relevant content for the next usage session after the expected unlock event.

By empirically validating the predictability of unlock events, our work generalises the modeling of smartphone usage patterns to also include unlocking itself, in addition to existing approaches that predict which app users will launch next [25]. Similar to previous approaches using Random Forests on smartphone such as call-availability prediction [35], boredom detection [36], and gesture recognition [40], our approach is feasible as a stand-alone application for off-the-shelf smartphones.

5.3 Software-Related Context

Based on the results of feature evaluation on the binary classification problem using Weka's "select attributes", and the two importance measures of Random Forests, we found that the strongest indicators of unlock events are software-related features: idle time (time since the end of last session), the duration of last session, and the hour of day. The same holds true for regression: using IncMSE and IncNodePurity we also observed that software-related features lead to better higher accuracy and lower dispersion. For regression, those feature were; time since the end of last session, the duration of last session, and the hour of day. Additionally, we found a strong correlation ($r_s = 0.411$) between idle time (time since the end of last session) and the time before next unlocking event. This correlation is the strongest compared to that of any other feature which is a continuous variable.

This finding is in line with results from van Berkel et al. [5], who compared multiple features to predict whether a user would continue with a previous objective or start a new objective unrelated to the previous task when unlocking their phone. Following from these results, we argue that mobile phone usage behaviour follows a pattern that is related to both the previous user interaction and the time of day. These attributes can be collected by relying solely on context information obtained through the device's software. This means that our approach can run on the personal device of the user without need for hardware sensors or communication with the outside world.

Furthermore, it is interesting to note that Random Forests using software-related context obtains slightly higher prediction accuracy than using all the context. This phenomenon indicates that some features obtained from hardware are correlated with software-related features, or are irrelevant to the unlock events. Redundant features can decrease the classification performance of Random Forests. Also, feature evaluation shows that several hardware-related features such as WiFi availability and GPS availability have significantly low importance, yet are energy intensive.

5.3.1 Energy Efficiency in Unlocking Prediction

Since one of our objectives is to reduce energy consumption in the locked state, an important condition of a successful implementation is that the model itself is energy efficient. In the usage of unlocking prediction, the algorithm works with a trained model on the smartphone, consuming no phone power to train a new model. Specifically, based on our experimental results , we found that using software-related context (i.e., the data collection for screen events and clock time) has negligible impact on smartphone battery. This find is in line with prior work stating that quick computations by CPU and RAM consume only little power in the locked state [9], [17]. As stated in [17], operating systems has optimal mechanisms for system-level broadcasts such as screen and application events. Fetching these events does not involve extra overhead.

Overall, our findings suggest that using only software-related features is a feasible strategy to achieve better accuracy and energy efficiency in performing smartphone unlock prediction, compared to using both software-related and hardware-related features.

5.4 Deployment of Unlocking Prediction: Personalised Model and Cold Start

In practice, systems based on supervised learning have two ways of deployment for a new user: personalising a model using data collected from the new user, or launching a cold start with a model trained by data collected from previous users' data.

Personalised models are commonly used by most context-aware techniques on smartphone. These models gradually learn and adapt to their user. Using personalised models can maximise the accuracy of predictions, as demonstrated in our results 16. When using short time windows, personalised models tend to generate wrong prediction for the class "yes" instances. Repeated random sub-sampling can be used to overcome this problem. Using long time windows, personalised models can generate accurate predictions for both classes.

An important strength of personalised models is the protection of privacy. Since a smartphone collects and processes data from its own user, personalised models do not involve any connection to a network and/or other devices. After training a model, the smartphone can delete the raw data to minimise risks. However, personalised models require that some data collection occurs before they can be used. Although unlocking prediction is an opportunistic sensing system (i.e., users do not explicitly input information to train a model), it may take long time (e.g., 2 weeks in our study) to train a model with sufficient performance. Also, many factors will affect the training cost (time and power usage), such as hardware, data size, algorithm parameters and implementation of software. Since our approach only requires Random Forests algorithm, it can work on any platform including Android and iOS.

In contrast, a cold start can directly install the system with a trained model on the smartphone of the new user, so that the smartphone can have predictions immediately after the installation. This model is trained by data collected from previous users. Although there is no training effort for the new user in a cold start, our findings indicate that this approach has poor performance. Our evaluation

results indicate that the models of cold start cannot accurately predict instances of class "yes" regardless of time window sizes. This finding reflects the significant difference of phone usage behaviours across different users. Also, cold start models cannot accurately predict instances of class "no" in long time windows. This means that a cold start may be useful in very few cases.

Also, a cold start implies that previous users have to upload their raw data to a central server that trains the model. It is challenging to recruit such users without sufficient compensation. Even with considerable compensation, this data collection may involve privacy risks that may be harmful to these users and may decrease their willingness of data uploading. In practice, this can be achieved through volunteers that sign up for beta testing.

5.5 Limitations and Future Work

In this study we do not distinguish unlock events caused by notifications or phone calls which users unlock the phone in a passive way (i.e., phone usage without a planned user intention) . This means that future research can investigate smartphone unlock events on a lower level comprising:

- active unlock events initiated by the phone user;
- passive unlock events where the user is notified via sound, vibration, LED light or the bright screen by the phone.

Then the classifier can make more detailed predictions to support certain applications. However, the classification performance on the lower level might decrease for brief time windows because the classes are more imbalanced (active/passive unlock events are subclasses of unlock events.) , as we have discussed in this paper. Based on the predictions of active/passive unlock events, smartphone apps can further benefit users. On the other hand, splitting more classes within unlock events increases the cost of when the model is trained on hardware.

Constrained by our computing resources for data analysis, another limitation is that we did not include diverse demographic features in our model. We had 27 participants (23 with academic background) which cannot capture rich demographic diversity among billions of smartphone users. Many demographic features including age, occupation, education and personality may be useful for the classifier to achieve higher accuracy. These features are also critical to implement an accuracy model for new users in a cold start scenario. Future research can explore possible solutions to improve the model for a cold start. Moreover, changes in demographics may greatly impact phone usage. For example, a user may radically change their phone usage behaviours after moving to another country, so that a model trained in the previous country may have lower performance. Similarly, users' mental states (e.g., depression and anxiety) may be helpful to increase the accuracy the prediction. Developers can connect medical apps as a data source to extend our system. To investigate the effectiveness of additional features, future research can extend our study to a larger group of participants. However, the computational cost (e.g., time and hardware) in data analysis will also be larger. Researchers should have sufficient computing resources such as servers or workstations with large RAM.

Additionally, in the future, researchers can also attempt to generalise our findings to other mobile devices such as tablets and smartwatches. The usage scenarios

of our approach on smartphones are similar to those of tablets and smartwatches. If our approach is suitable for these devices, accurate unlock prediction will also be a great benefit for tablet and smartwatch applications.

6 Conclusion

In this paper we propose multiple approaches for using context data to predict when a smartphone will be unlocked. Based on the results of our field study with 27 participants, we found that it is possible to predict the next phone unlock event, using Random Forests as a classifier/regressor and smartphone contextual data including hardware sensor data and phone usage patterns. Furthermore, our feature evaluation indicates that the strongest indicators of phone unlocking are software-related features including idle time (time since the end of last session), the duration of last session, and the hour of day. We also found that using software-related features only can slightly improve the accuracy and enable energy efficiency of continuous sensing in unlocking prediction.

In the comparison between personalised models and cold start, our results show that personalised models can generate better predictions (ROC Area from 0.885 to 0.99 using different time windows), whereas cold start models cannot achieve the same performance. We also show that there exists a trade-off in the time window selection. If the classifier uses a longer time window, the model will have higher accuracy in positive predictions and lower accuracy in negative predictions. To achieve high accuracy in positive predictions by Random Forests using short time windows, we found that the repeated random sub-sampling method is very effective.

In contrast, personalised models using Random Forests in the regression mode can generate the exact time before next unlocking event (MAE = 8.89 mins, RMSE = 24.44 mins), without considering time windows. Our findings enable smartphones applications to collect sensor data or prepare timely content to improve the context-awareness for the next phone usage session. Also, by inferring the next period of the idle state, smartphones applications and operating systems can reduce unnecessary operations to improve energy efficiency, and schedule computation-intensive tasks, such as OS updates, in the locked state to avoid the disturbance of phone usage.

A Appendix

A.1 Parameter Tuning

To improve the performance of SVM, we first tried the linear kernel. However, the performance of SVM with linear kernel degraded ($ROC\ Area = 0.523$ at 10 min) compared to the RBF kernel ($ROC\ Area = 0.571$ at 10 min). Hence, we attempted to apply parameter tuning methods on the RBF kernel. As illustrated in previous work investigating parameter tuning for SVM with the Radial Basis Function kernel [24, 10] , we repeated the 10-fold cross-validation on the data of 10 min time window with replacing the default setting $\gamma = \frac{1}{m+1}$ with a wide variety of γ values.

Figure 20 shows the performance of SVM with the RBF kernel having different γ values. Among all γ assignments, $\gamma = 100$ achieved the best performance: $ROC\ Area = 0.813$, class

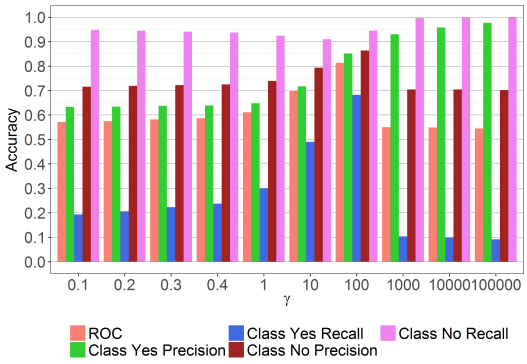

Fig. 20: SVM with the Radial Basis Function kernel

"no" precision 0.864, class "no" recall 0.945, class "yes" precision 0.852, class "yes" recall 0.682.

The results indicate that, given a suitable γ, SVM with the RBF kernel can also achieve high performance in the classification to predict unlocking events. However, due to the large number of possible γ values and our limited computing resources, we could not refine our finding, since parameter tuning is highly time-consuming and computation-intensive (in our case, running each γ value took about 10 days for a normal PC). With $\gamma = 100$, although SVM with the RBF kernel achieved considerably good results, the performance of Random Forests was still better. Hence, we focused on Random Forests in further analysis. Future work may conduct deeper investigation about the employment of SVM with the RBF kernel in phone unlocking prediction.

References

1. Abdullah, S., Matthews, M., Murnane, E.L., Gay, G., Choudhury, T.: Towards circadian computing: "early to bed and early to rise" makes some of us unhealthy and sleep deprived. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp '14, pp. 673–684. ACM, New York, NY, USA (2014). DOI 10.1145/2632048.2632100. URL `http://doi.acm.org/10.1145/2632048.2632100`
2. Avrahami, D., Hudson, S.E.: Responsiveness in instant messaging: predictive models supporting inter-personal communication. In: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 731–740. ACM (2006)
3. Banovic, N., Brant, C., Mankoff, J., Dey, A.: Proactivetasks: the short of mobile device use sessions. In: Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, pp. 243–252. ACM (2014)
4. Ben Abdesslem, F., Phillips, A., Henderson, T.: Less is more: energy-efficient mobile sensing with senseless. In: Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds, pp. 61–62. ACM (2009)
5. van Berkel, N., Luo, C., Anagnostopoulos, T., Ferreira, D., Goncalves, J., Hosio, S., Kostakos, V.: A systematic assessment of smartphone usage gaps. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, pp. 4711–4721. ACM (2016)
6. Bobadilla, J., Ortega, F., Hernando, A., Bernal, J.: A collaborative filtering approach to mitigate the new user cold start problem. Knowledge-Based Systems **26**, 225–238 (2012)
7. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
8. Brown, B., McGregor, M., McMillan, D.: 100 days of iphone use: understanding the details of mobile device use. In: Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, pp. 223–232. ACM (2014)
9. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone (2010)

10. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 27 (2011)
11. Chon, Y., Talipov, E., Shin, H., Cha, H.: Mobility prediction-based smartphone energy optimization for everyday location monitoring. In: Proceedings of the 9th ACM conference on embedded networked sensor systems, pp. 82–95. ACM (2011)
12. Do, T.M.T., Blom, J., Gatica-Perez, D.: Smartphone usage in the wild: a large-scale analysis of applications and context. In: Proceedings of the 13th international conference on multimodal interfaces, pp. 353–360. ACM (2011)
13. Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R., Estrin, D.: Diversity in smartphone usage. In: Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 179–194. ACM (2010)
14. Ferreira, D., Dey, A., Kostakos, V.: Understanding human-smartphone concerns: a study of battery life. Pervasive computing pp. 19–33 (2011)
15. Ferreira, D., Ferreira, E., Goncalves, J., Kostakos, V., Dey, A.K.: Revisiting human-battery interaction with an interactive battery interface. In: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing, pp. 563–572. ACM (2013)
16. Ferreira, D., Goncalves, J., Kostakos, V., Barkhuus, L., Dey, A.K.: Contextual experience sampling of mobile application micro-usage. In: Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, pp. 91–100. ACM (2014)
17. Ferreira, D., Kostakos, V., Dey, A.K.: Aware: mobile context instrumentation framework. Frontiers in ICT **2**, 6 (2015)
18. Fischer, J.E., Greenhalgh, C., Benford, S.: Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In: Proceedings of the 13th international conference on human computer interaction with mobile devices and services, pp. 181–190. ACM (2011)
19. Genuer, R., Poggi, J.M., Tuleau-Malot, C.: Variable selection using random forests. Pattern Recognition Letters **31**(14), 2225–2236 (2010)
20. Google: Activity recognition api. `https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi` (2017)
21. Google: Sensor. `http://developer.android.com/reference/android/hardware/Sensor.html` (2017)
22. Ha, J.H., Chin, B., Park, D.H., Ryu, S.H., Yu, J.: Characteristics of excessive cellular phone use in korean adolescents. CyberPsychology & Behavior **11**(6), 783–784 (2008)
23. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD explorations newsletter **11**(1), 10–18 (2009)
24. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98 pp. 137–142 (1998)
25. Jones, S.L., Ferreira, D., Hosio, S., Goncalves, J., Kostakos, V.: Revisitation analysis of smartphone app use. In: International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp, pp. 1197–1208 (2015). DOI 10.1145/2750858.2807542. URL `http://people.eng.unimelb.edu.au/vkostakos/files/papers/ubicomp15.pdf`
26. Khalilia, M., Chakraborty, S., Popescu, M.: Predicting disease risks from highly imbalanced data using random forest. BMC medical informatics and decision making **11**(1), 51 (2011)
27. Khoshgoftaar, T.M., Golawala, M., Van Hulse, J.: An empirical study of learning from imbalanced data using random forest. In: Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on, vol. 2, pp. 310–317. IEEE (2007)
28. Kostakos, V., Musolesi, M.: Avoiding pitfalls when using machine learning in hci studies. interactions **24**(4), 34–37 (2017)
29. Krstic, I.: Behind the scenes of ios security. Black Hat (2016)
30. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., Campbell, A.T.: A survey of mobile phone sensing. IEEE Communications magazine **48**(9) (2010)
31. Lee, M., Bak, C., Lee, J.W.: A prediction and auto-execution system of smartphone application services based on user context-awareness. Journal of Systems Architecture **60**(8), 702–710 (2014)
32. Liu, Y., Xu, C., Cheung, S.C.: Where has my battery gone? finding sensor related energy black holes in smartphone applications. In: Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on, pp. 2–10. IEEE (2013)

33. Lu, H., Yang, J., Liu, Z., Lane, N.D., Choudhury, T., Campbell, A.T.: The jigsaw continuous sensing engine for mobile phone applications. In: Proceedings of the 8th ACM conference on embedded networked sensor systems, pp. 71–84. ACM (2010)
34. Oulasvirta, A., Rattenbury, T., Ma, L., Raita, E.: Habits make smartphone use more pervasive. Personal and Ubiquitous Computing **16**, 105–114 (2012)
35. Pielot, M.: Large-scale evaluation of call-availability prediction. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 933–937. ACM (2014)
36. Pielot, M., Dingler, T., Pedro, J.S., Oliver, N.: When attention is not scarce-detecting boredom from mobile phone usage. In: Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing, pp. 825–836. ACM (2015)
37. Pielot, M., de Oliveira, R., Kwak, H., Oliver, N.: Didn't you see my message?: predicting attentiveness to mobile instant messages. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, pp. 3319–3328. ACM (2014)
38. Poppinga, B., Heuten, W., Boll, S.: Sensor-based identification of opportune moments for triggering notifications. IEEE Pervasive Computing **13**(1), 22–29 (2014)
39. Shin, C., Hong, J.H., Dey, A.K.: Understanding and prediction of mobile application usage for smart phones. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 173–182. ACM (2012)
40. Song, J., Sörös, G., Pece, F., Fanello, S.R., Izadi, S., Keskin, C., Hilliges, O.: In-air gestures around unmodified mobile devices. In: Proceedings of the 27th annual ACM symposium on User interface software and technology, pp. 319–329. ACM (2014)
41. Strobl, C., Boulesteix, A.L., Zeileis, A., Hothorn, T.: Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC bioinformatics **8**(1), 25 (2007)
42. Svetnik, V., Liaw, A., Tong, C., Culberson, J.C., Sheridan, R.P., Feuston, B.P.: Random forest: a classification and regression tool for compound classification and qsar modeling. Journal of chemical information and computer sciences **43**(6), 1947–1958 (2003)
43. Truong, K.N., Shihipar, T., Wigdor, D.J.: Slide to x: unlocking the potential of smartphone unlocking. In: Proceedings of the 32nd annual ACM conference on Human factors in computing systems, pp. 3635–3644. ACM (2014)
44. Verkasalo, H.: Contextual patterns in mobile service usage. Personal and Ubiquitous Computing **13**(5), 331–342 (2009)
45. Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., Campbell, A., Choudhury, T.: Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In: Proceedings of the 2013 International Symposium on Wearable Computers, pp. 69–76. ACM (2013)
46. Yan, T., Chu, D., Ganesan, D., Kansal, A., Liu, J.: Fast app launching for mobile devices using predictive user context. In: Proceedings of the 10th international conference on Mobile systems, applications, and services, pp. 113–126. ACM (2012)
47. Yuan, Y., Raubal, M., Liu, Y.: Correlating mobile phone usage and travel behavior–a case study of harbin, china. Computers, Environment and Urban Systems **36**(2), 118–130 (2012)
48. Zhuang, Z., Kim, K.H., Singh, J.P.: Improving energy efficiency of location sensing on smartphones. In: Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 315–330. ACM (2010)