Chapter 3
A Literature Review on Context-Aware Machine Learning and Mobile Data Analytics

3.1 Contextual Information

The term context has a wide range of meanings and can be applied to a variety of situations. In this section, we first go through some of the existing context definitions in the domain of mobile and pervasive computing, and then we go over why contexts are important in a particular application.

3.1.1 Definitions of Contexts

Context has been employed in a variety of fields, including pervasive and ubiquitous computing, human-computer interaction, computer-supported collaborative work, and ambient intelligence [1]. Early efforts on context-awareness in the area of ubiquitous and pervasive computing referred to context as essentially the location of people and objects [2]. Context has recently been expanded to encompass a broader set of factors, such as an entity's physical and social features, as well as user behaviors [1]. Following a review of the pervasive and ubiquitous computing community's definitions and categories of context, this part aims to describe the concept of the context within the area. Because the concepts of context in the domain of pervasive and ubiquitous computing are similarly broad and this discussion is meant to be informative rather than comprehensive.

From various viewpoints, several research has sought to define and describe the context. Schilit et al. [2], for example, consider the user's location information, the surrounding persons and objects, and the changes to those objects as contexts. Contexts are also defined by Brown et al. [3] as the user's locational information, temporal information, the surrounding individuals around the user, temperature, and so on. In the same way, the user's locational information, ambient information, temporal information, and identity are all considered contexts By Ryan et al. [4].

I. H. Sarker et al., *Context-Aware Machine Learning and Mobile Data Analytics*,
https://doi.org/10.1007/978-3-030-88530-4_3

Other context definitions have merely provided synonyms for contexts, such as context as the environment or social condition. A lot of studies have considered the context as the user's environmental information. For example, in [5], Brown et al regarding the environmental information that the user's computer is aware of as context, whereas Franklin et al [6] regards the user's social setting as context. Other researchers, on the other hand, believe it is the environment that is related to the applications. Ward et al. [7], for example, consider the state of the applications' surrounding information as contexts. Context is defined by Hull et al. [8] as the features of the user's current position, which includes the complete surroundings. In Rodden et al. [9], the settings of apps are likewise considered as context.

Schilit et al. [10] argued that the best parts of context are (i) where you are, (ii) who you are with, and (iii) what resources are nearby. In their definition, information about the changing environment is taken into consideration as context. They encompass the computational environment as well as the physical environment, in addition to the user environment, e.g., user location, adjacent individuals, and the current social position of the user. For example, the computing environment can include connection, available processors, user input and display, network capacity, and computing costs, while the physical environment can include noise, temperature, and lighting levels.

Dey et al. [11] give a survey of different views of context, which are mostly imprecise and indirect, often defining context by synonym or example. Finally, he provides the following definition of context, which is now widely accepted. According to Dey et al. [11] "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves".

3.1.2 Understanding the Relevancy of Contexts

Realizing the importance of contexts is merely the first step in properly utilizing them in mining contextual behavioral rules of individual mobile phone users. We need a clear understanding of what circumstances influence users to make decisions in different situations to make efficient use of contexts in mobile phone users' behavioral rules. The contexts associated with the user are the most relevant as we aim to discover the user behavioral rules using their mobile phone data. Table 3.1 depicts an example of user situations influencing decision-making when dealing with phone call interruptions. The relevance of the contexts, on the other hand, is application particular, i.e., it may differ from one application to the next in the real world.

Consider a personalized smart mobile app management system that can predict an individual's future app usages (Skype, Whatsapp, Facebook, Gmail, Microsoft Outlook) based on contextual data. When the user is in her office on weekdays

Table 3.1 Various types of user contexts influencing making decisions while handling the phone call interruptions

Context category	Context examples
Temporal context	User's activity occuring date (YYYY-MM-DD), time (hh:mm:ss), period (e.g., 1 h, 10:00 a.m.–12:00 p.m.), weekday (e.g., Monday), weekend (e.g., Saturday), etc.
Spatial context	User's coarse level location such as office, work, home, market, restaurant, vehicle, playground etc.
Social context	User's social activity or situation such as professional meeting, lecture, seminar, lunch break, dinner, etc., and/or, social relationship between individuals such as mother, friend, colleague, boss, significant one, unknown, etc.

between 9:00 a.m. and 10:00 a.m., she normally uses Microsoft Outlook for mailing purposes. The user's contexts, such as temporal (Weekdays between 09:00 a.m. and 10:00 a.m.) and place (office), may be relevant to intelligently assist her in finding this particular mobile application among a large number of installed apps on her mobile phone.

Consider another example: a smartphone call interruption management system, which may require more contexts. Mobile phones are commonly considered to being "always on, always connected" devices in the real world, yet mobile users are not always attentive and receptive to incoming contact [12]. Let's say a user has a regular meeting at her office on Monday between 9:00 and 11:00 a.m. She usually rejects incoming phone calls during that period since she does not want to be interrupted during the meeting. If the phone call is from her boss or mother, she wants to answer it since it seems to be important to her. According to this example, user phone call response behaviors are related not only to contexts, location (e.g., workplace), and temporal (e.g., Monday, between 9:00 a.m. and 11:00 a.m.), but also to additional contexts, social situations (e.g., meeting), and social relationships between individuals (e.g., boss or mother). As a result, the relevance of user circumstances differs from app to app in the real world.

With a better understanding of contexts, mobile app developers will be able to choose which contexts to be included in their apps, allowing them to create context-aware apps that deliver personalized services and intelligently aid users in their daily activities as well as smartphone based IoT services [123, 124]. According to the aforementioned real-world examples, individual mobile phone users' behavioral rules should not be dependent on a fixed number of contexts. To meet these needs, we provide a set of behavioral rules for individual mobile phone users based on multi-dimensional contexts available in the mobile phone dataset, which may be employed in relevant applications for the mobile phone user.

3.2 Context Discretization

The discretization of continuous contextual data is one of the key research areas covered in this book. The discretization method converts continuous numerical attributes into discrete or nominal attributes with a finite number of intervals, resulting in a non-overlapping partition of a continuous domain.

3.2.1 Discretization of Time-Series Data

Temporal context, represented as time-series data, is the most important aspect that influences user behavior in a mobile Internet portal [13], according to the scope of continuous context considered in this book. A time series is defined as "a sequence of data points ordered in time, often measured at successive time points" [14]. In this section, we focus on the discretization of time-series data as context discretization.

Unlike digital systems, human perception of time is not precise. Routine behaviors always have a time interval, even if it is only a little one, such as 5 min. Time must be segmented into meaningful categories that act as a proxy for distinguishing user's various activities to evaluate time as a condition in a high confidence rule. As a result, discretization of time-series data is required, which may then be used as the foundation for a mobile phone-based context-aware rule learning system. Its major purpose is to convert continuous time-series attributes into discrete or categorical values, such as time segments, hence converting quantitative data into qualitative data. According to [15], time-based behavior modeling is an open problem. Hence, we summarize the existing time-series segmentation approaches into two broad categories; (i) static segmentation, and (ii) dynamic segmentation, which is used in various mobile applications. In the following, we discuss these methods used in various application domains.

3.2.2 Static Segmentation

A static segmentation is simple to comprehend and can be useful for comparing population behavior among cell phone users. Most researchers recently consider only the temporal coverage (24-hours-a-day) and statically segment time into arbitrary categories (e.g., morning) or times (e.g., 1 h) to produce segments, as shown in Table 3.2. This form of static time segmentation focuses primarily on time intervals. According to [16], there are two forms of time intervals: equal and unequal time intervals.

A number of researchers have used equal interval-based segmentation in their applications. For instance, Song et al. [17] present a log-based analysis on users' search activity in order to increase search relevance by splitting the 24-h day

Table 3.2 Various types of static time segments used in different applications

Time interval type	Number of segments	Used time interval and segment details	References
Equal	3	Morning [7:00–12:00], afternoon [13:00–18:00] and evening [19:00–24:00]	Song et al. [17]
Equal	3	[0:00–7:59], [8:00–15:59] and [16:00–23:59]	Rawassizadeh et al. [18]
Equal	4	Morning [6:00–12:00], afternoon [12:00–18:00], evening [18:00–24:00] and night [0:00–6:00]	Mukherji et al. [19]
Equal	4	Morning [6:00–12:00], afternoon [12:00–18:00], evening [18:00–24:00] and night [0:00–6:00]	Bayir et al. [20]
Equal	4	Morning, afternoon, evening and night	Paireekreng et al. [21]
Equal	4	Morning [6:00–11:59], day [12:00–17:59], evening [18:00–23:59], overnight [0:00–5:59]	Jayarajah et al. [22]
Equal	4	Night [0:00 a.m.-6:00 a.m.], morning [6:00 a.m.-12:00 p.m.], afternoon [12:00 p.m.-6:00 p.m.], and evening [6:00 p.m.-0:00 a.m.]	Do et al. [23]
Unequal	3	Morning (beginning at 6:00 a.m. and ending at noon), afternoon (ending at 6:00 p.m.), night (all remaining hours)	Xu et al. [24]
Unequal	4	Morning [6:00–12:00], afternoon [12:00–16:00], evening [16:00–20:00] and night [20:00–24:00 and 0:00–6:00]	Mehrotra et al. [25]
Unequal	5	Morning [7:00–11:00], noon [11:00–14:00], afternoon [14:00–18:00] and so on	Zhu et al. [26]
Unequal	5	Morning, forenoon, afternoon, evening, and night	Oulasvirta et al. [27]
Unequal	5	Morning [7:00–11:00], noon [11:00–14:00], afternoon [14:00–18:00], evening [18:00–21:00], and night [21:00-Next day 7:00]	Yu et al. [28]
Unequal	>5	Early morning, morning, late morning, midnight and so on	Naboulsi et al. [29]
Unequal	>5	Early morning, morning, late morning, midnight and so on	Dashdorj et al. [30]
Unequal	>5	Early morning, morning, late morning, midnight and so on	Shin et al. [31]
Unequal	8	S1[0:00 a.m.–7:00 a.m.], S2[7:00 a.m.–9:00 a.m.], S3[9:00 a.m.–11:00 a.m.], S4[11:00 a.m.–2:00 p.m.], S5[2:00 p.m.–5:00 p.m.], S6[5:00 p.m.–7:00 p.m.], S7[7:00 p.m.–9:00 p.m.] and S8[9:00 p.m.–12:00 a.m.]	Farrahi et al. [33]

into three equivalent time segments, e.g., morning [7:00–12:00], afternoon [13:00–18:00], and evening [19:00–2400]. Using three temporal segments [0:00–7:59], [8:00–15:59], and [16:00–23:59], Rawassizadeh et al. [18] propose a scalable method for regular behavioral pattern mining from multiple sensor data. Morning [6:00–12:00], afternoon [12:00–18:00], evening [18:00–2400], and night [0:00–6:00] are the four period segments considered by Mukherji et al. [19]. Using the same four time segments, Bayir et al. [20] suggest a web-based customized mobility service for mobile applications. Paireekreng et al. [21] introduced a personalization mobile game recommendation framework using time-of-day divided into four cycles—morning, midday, evening, and night. Jayarajah et al. [22] use morning [6:00–11:59], day [12:00–17:59], evening [18:00–23:59], and overnight [0:00–5:59] to understand the difference in variety seeking over various time windows. In their application model, Do et al. [23] night [0:00 a.m.–6:00 a.m.], morning [6:00 a.m.–12:00 p.m.], afternoon [12:00 p.m.–6:00 p.m.], and evening [6:00 p.m.–0:00 a.m.] to explain how user behavior changes with respect to time of day.

Several researchers have used unequal interval-based segmentation in their applications. For instance, Xu et al. [24] have provided a prediction system for smartphone app usages that incorporates three important everyday factors that affect user app use behavior (context, group behavior, and user preferences). Morning (starting at 6:00 a.m. and finishing at noon), afternoon (ending at 6:00 p.m.), and night (all remaining hours) are the time segments they use. Mehrotra et al. propose a novel interruptibility management solution in [25] that learns users' preferences for receiving mobile alerts based on automated rule extraction by mining their contact with mobile phones. Morning [6:00–12:00], afternoon [12:00–16:00], evening [16:00–20:00], and night [20:00–24:00 and 0:00–6:00] are the four-time slots they use for segmentation. In their recommendation scheme, Zhu et al. [26] use five static time segments in a day that are predefined as morning [7:00–11:00], noon [11:00–1400], afternoon [14:00–18:00], and so on. Oulasvirta et al. [27] use five-time slots (morning, forenoon, afternoon, evening, and night) as temporal context to explain each user's thoughts, ideas, beliefs, and emotions. Yu et al. investigate how to mine topic models to manipulate user context logs for customized context-aware suggestion in [28]. Morning [7:00–11:00], noon [11:00–14:00], afternoon [14:00–18:00], evening [18:00–21:00], and night [21:00-Next day 7:00] are the period segments used throughout their framework.

A number of authors [29–31] add to the above segmentations by introducing early morning, late morning, midnight, and so on. Shin et al. propose a new context model for app prediction in [32], which gathers a wide variety of contextual information in a smartphone and makes customized app predictions using a naive Bayes model. They divide time for weekdays and weekends into early morning, morning, afternoon, evening, and night in their model. Farrahi et al. [33] divide each day into 8 coarse-grain time slots as follows: [0:00 a.m.–7:00 a.m.], [7:00 a.m.–9:00 a.m.], [9:00 a.m.–11:00 a.m.], [11:00 a.m.–2:00 p.m.], [2:00 p.m.–5:00 p.m.], [5:00 p.m.–7:00 p.m.], [7:00 p.m.–9:00 p.m.] and [9:00 p.m.–12:00 a.m.]. These time slots were chosen to represent popular activities in everyday life, such as lunch, dinner, or work hours in the morning and afternoon. These types of segmentation are often used in a

variety of applications such as mining mobile user habits [34, 35], managing mobile intelligent interruption management system [36], mining frequent co-occurrence patterns on the mobile phones [37], making app prefetch practical on mobile phones [38].

Several authors use time segments for different events scheduled in their calendar in addition to the above time segments to predict individual cell phone user activity. Cell phones are also considered one of the main means of accessing calendars (e.g., Google Calendar) to coordinate schedules such as meetings since they are still associated and carrying with the users [39]. The calendar [40] allows the user to identify unique tasks or events with length, temporal domain, and other attributes. For example, if the calendar shows a meeting between 13:00 and 14:00, they presume the user is inaccessible and is in a location with at least one other person [41]. The time interval [13:00–14:00] is then used to forecast her cell phone use. Calendar entries, according to Khalil et al. [42], is a good indicator of whether an individual is available or unavailable for a phone call. Salovaara et al. [43] conducted a study and found that 31% of incoming phone calls were due to unavailability, i.e., users were unable to answer the phone calls due to meetings, classes, appointments, driving, or sleeping.

Several authors have designed a context-aware interruption management framework that produces as an output if an incoming call should be enabled to ring by taking into account the user's above unavailability solution for a specific time segment (e.g., between 13:00 and 14:00) using an individual's calendar details. Dekel et al. [44] build an application to reduce cell phone disturbances, for example. The developers of [45] and [36] use calendar information to create a context-aware interruption management system. To enhance mobile phone understanding, Seo et al. [46] use the user's schedule to determine policy rules in their context-aware phone configuration management framework. The interruption handling rules in these methods are focused on static temporal segments based on their scheduled appointments in their individual's calendar details, for example, the user is unable to answer the incoming call while s/he is in a calendar case (e.g., a meeting between 13:00 and 14:00). However, in some situations, such an unavailability approach offers poor accuracy.

Khalil et al. [47] surveyed 72 phone users and discovered that the above unavailability solution for mobile communication has low accuracy (62%) for loosely organized home activities like lunch, watching TV, and doing homework, but high accuracy (93%) for structured events like classes, meetings, and appointments. However, such special terms are insufficient to cover real-world use cases; a larger range of meeting categories keywords is needed to capture users' actual actions [44]. Even if the user is involved in an ongoing task or social situation, the phone call is always not disruptive, and the call is welcomed because it offers a required mental break from the current task [48]. According to [49], 24% of mobile phone users feel compelled to pick up a phone call while in a meeting. According to a user survey conducted by Rosenthal et al. [50], 35% of participants want to receive phone calls at work, while the rest do not. Sarker et al. [41] have demonstrated that the presence of a calendar event for a specific time segment is insufficient to assume

individual actions for their different calendar events. According to [51], the calendar does not offer a reliably accurate depiction of the real world because events do not occur or occur outside the calendar's allocated time window. As a result, calendar-based temporal segments are ineffective at capturing the actions of individual mobile phone users [41].

Although different time intervals and corresponding segmentation are used for different purposes (see Table 3.2), these methods all take into account a fixed number of segments for all users. However, users' behavioral evidence that varies from user to user over time in the real world is not taken into account when doing such segmentation. As a result, static segment generation may not be appropriate for producing high-confidence temporal rules for individual smartphone users. For example, in one case, a N_1 number of segments may yield meaningful results, whereas in another case, a N_2 number of segments may yield better results, where $N_1 \neq N_2$. As a result, rather than statically generating rules, dynamic segmentation of time may be able to represent individuals' behavioral evidence over time and play a role in producing high confidence rules based on their utilization records.

3.2.3 Dynamic Segmentation

A segmentation technique that produces a variable number of segments, as discussed above, will be more useful for modeling users' behavior. To achieve the target, a dynamic segmentation technique rather than a static segmentation technique may be used. The number of segments in a dynamic segmentation is not set and predefined; it can change based on behavioral features, patterns, or preferences. There are many dynamic segmentation strategies for modeling users' behavioral patterns in temporal contexts that generate a variable number of segments. To produce the segments, several authors simply take into account a single parameter, such as interval length or base time. Depending on the time frame, the number of time segments varies. The number of segments will be T_{max}/BP [52] if T_{max} reflects the entire 24-h time span and BP is a base period. The number of time segments decreases as the base period increases and vice versa. If the base time is 5 min, the number of segments will be determined by dividing 24-$hours$-a-day by 5. In this case, a base period of 5 min is assumed to be the finest granularity for distinguishing an individual's day-to-day activities. The number of segments decreases as the base time is increased to 15 min, with 15 min being considered to be the finest granularity. As a result, the number of segments varies depending on the starting time frame.

For example, Ozer et al. [53] suggest using sequential pattern mining techniques to predict the location and time of cell phone users. In their process, they use a 15-min time interval for segmentation and then switch to 60-min intervals in their experiments. Do et al. present a system for predicting where users will go and which app they will use next using rich contextual knowledge from smartphone sensors in [54]. They use 30 min as the parameter value in their system. Farrahi et al. use temporal data to discover everyday habits from large-scale cell phone data

in [55]. They also divide each day of the week into 30-min segments using 30 min as the parameter value. Karatzoglou et al. use 2-h as the parameter value in their mobile app recommendation system in [56]. In their analysis to classify human daily activity patterns using cell phone data, Phithakkitnukoon et al. [57] use 3 h for time segmentation. For various purposes, different authors use different interval values to create a variable number of segments. When the value of such an interval is high, it results in a small number of segments, and vice versa. However, the optimal value of this parameter, which we are interested in, is required for an effective segmentation that captures individual mobile user behavior.

Individual calendar schedules and corresponding time boundaries may also be used to evaluate variable duration time segments to model users' actions in a temporal sense, which may differ depending on users' preferences [41]. For example, one user may have an event between 1 and 2 p.m., while another may have an event between 1:30 pm and 2:30 pm. As a result, the time segmentation varies depending on the events they have planned in their calendars. Multiple thresholds, sliding windows, and data shape-based methods, as shown in Table 3.3, are also used in many applications. Halvey et al. [58] proposed a multi-thresholds-based approach for segmenting time-series log data to predict mobile device navigation patterns. However, since no previous awareness of user behaviors exists, choosing these thresholds to define the lower and upper boundary of a segment is extremely difficult.

Several authors use machine learning methods such as clustering, genetic algorithms, and others in addition to these approaches. To discover rules from time series, Das et al. [59] suggest a cluster-based technique. The issue is that the number of clusters must be known ahead of time, which is difficult to predict for an individual. Besides these, GA based [60, 61], sliding window-based [62, 63], shape-based [16, 64] segmentation have been proposed for different purposes.

The user's total number of activity occurrences at each time point is used to segment the data. These are not, however, behavior-oriented segmentations since they do not account for the various actions of individuals that we are interested in. Using cell phone data, a variety of authors examine various usage habits over time. Phithakkitnukoon et al. [65], for example, create a behavior-based adaptive call prediction system based on mobile phone data. Jang et al. have shown in [66] that different users' app usage activity differs over time in a day while using mobile data. Henze et al. use mobile phone data in [67] to determine the best time to deploy applications. Xu et al. [68] use cell phone data to determine the best period for active applications. Based on user activity, Bohmer et al. [69] describe the peak time of typical app usages. These methods consider scanning over each hour time slot of the day (for example, [1:00 p.m.–2:00 p.m.]) to capture user habits and locate a specific predefined section for their purposes. Such methods, on the other hand, ignore the complex optimal segmentation based on an individual's actions. We have summarized a variety of works that use dynamic segmentation techniques for various purposes in Table 3.3.

Table 3.3 Various types of dynamic time segments used in different applications

Base technique	Description	References
Single parameter	A predefined value of time interval, e.g., 15 min is used to generate segments	Ozer et al. [53]
	A different value of time interval, e.g., 30 min is used for segmentation	Do et al. [54], Farrahi et al. [55]
	A relatively large value of the parameter, e.g., 2-h is used to generate time segments	Karatzoglou et al. [56]
	Another large value of time interval, e.g., 3-h is used for segmentation to make the number of segments small	Phithakkitnukoon et al. [57]
Calendar	Various calendar schedules and corresponding time boundaries are used to model users' behavior in temporal context	Khail et al. [47], Dekel et al. [44], Zulkernain et al. [36], Seo et al. [46], Sarker et al. [41]
Multi-thresholds	To identify the lower and upper boundary of a segment for segmenting time-series log data	Halvey et al. [58]
Data shape	A data shape based time-series data analysis	Zhang et al. [16], Shokoohi et al. [64]
Sliding window	A sliding window is used to analyze time-series data	Hartono et al. [62], Keogh et al. [63]
Clustering	A predefined number of clusters is used to discover rules from time-series data	Das et al. [59]
Genetic algorithm	A genetic algorithm is used to analyze time-series data	Lu et al. [60], Kandasamy et al. [61]

Clustering, as shown in Table 3.3, is an effective machine learning technique for forming broad time segments that take into account such user activity patterns. Clustering algorithms are typically built on certain assumptions and are biased against certain types of problems. In this sense, saying "best" in the context of clustering algorithms is a challenging task; it depends on the particular application [70]. The K-means algorithm is the most well-known squared error-based clustering algorithm [71] among a variety of clustering algorithms in the area of machine learning and data science. However, this algorithm requires the initial partitions and a fixed number of clusters K to be defined. With different starting points, the convergence centroids often change. Because of the estimation of mean values, outliers may often affect this algorithm. More significantly, this algorithm's characteristics aren't directly applicable to our context-aware rule learning. This algorithm, for example, assigns objects to the nearest cluster using the Euclidean distance function as a measure of similarity. However, Euclidean distance is ineffective for determining individual behavioral similarity and, as a result, learning behavioral rules. In the

presence of outliers, another K-medoids method [72], is more robust than the K-means algorithm since a medoid is less affected by outliers than a mean. As it reduces the outlier problem, K-means and the problem of time-series modeling have other characteristics in common.

Since the size and number of time segments are determined by the user's actions, which vary from one user to the next, bottom-up hierarchical data processing may aid in the formation of behavioral clusters. There are two types of hierarchical algorithms currently available: agglomerative methods and divisive methods. The system clustering process, on the other hand, is not widely used in practice [70]. Single linkage [73] and full linkage [74] are the simplest and most common agglomerative clustering methods. The single linkage agglomerative clustering algorithm is similar to another tool, nearest neighbor [70]. All of these hierarchical algorithms rely on a proximity matrix, which is computed by calculating the distance between two new clusters. The clusters are then successively merged according to the matrix value until the desired cluster structure is obtained. Because of the differences in user behavior, it is impossible to predict the degree to which merging is optimal according to a proximity matrix. Thus, using such clustering techniques, segments could be produced based on time-series data on user behavior patterns. Similarly, approaches based on genetic algorithms, such as those shown in Table 3.3, generate dynamic segments.

In summary, time-series modeling, using both the static and dynamic segmentation methods discussed above, can produce a variety of time segments that can be used for a variety of purposes. The above time-series modeling approaches, on the other hand, do not always map to trends of individual users based on their preferences, which are based on users' diverse habits through time-of-week and may differ from user to user. To effectively use temporal context as the basis for discovering rules capturing smartphone user behavior, a machine learning-based time-series modeling technique that takes into account such patterns may be important.

3.3 Rule Discovery

Another major focus of this study is using smartphone data to discover useful behavioral rules of individual cell phone users based on multi-dimensional contexts, such as temporal, spatial, or social contexts. In the field of machine learning, the most popular techniques for discovering such rules of individual cell phone users are association rule learning [75] and classification rule learning [76]. We will provide a brief overview of both association and classification strategies for discovering rules based on multi-dimensional contexts in the following sections.

3.3.1 Association Rule Mining

Association rule mining [77] is the discovery of associations or patterns or rules or relationships among a set of available items in a given dataset. Due to the descriptive and easily understandable existence of the discovered association rules, association rule mining has become a popular data mining technique [77]. Initial research into mining association rules was largely motivated by the analysis of retail market basket data to understand the purchasing behavior of the customers. One example is that "if a customer buys a computer or laptop (an item), s/he is likely to also buy anti-virus software (another item) at the same time". A common way of measuring the usefulness of association rules is to use its parameter, the 'support' and 'confidence' which is introduced in [77]. Support of a rule $Sup(A \Rightarrow C)$ is the percentage (%) of records in the dataset which carries all the items or contexts in a rule, and the confidence $Conf(A \Rightarrow C)$ is the percentage (%) of the records that carry all the items or contexts in the rule among those records that carry the items in the antecedent (A) of the rule.

Association rule mining algorithm discovers association rules that satisfy the predefined minimum support and confidence constraints from a given dataset [75]. The association rule mining problem is usually decomposed into two subproblems; (i) the first one is to identify several item sets whose occurrences exceed the predefined minimum support threshold in the dataset, those item sets are called frequent itemsets. We can define 'item set' as a non-empty set of items (each context value is considered as an item in the mobile phone dataset). The cardinality of an item set ranges from one to any positive number, e.g., is greater than zero. Each transaction record in the dataset contains an item set of size n, i.e., if a transaction record contains three different items (I_1, I_2, I_3), then the size of the item set is 3. An item set that can be found frequently in a dataset is typically called a frequent itemset, which identified the minimum support threshold. For instance, if a threshold is set to identify the frequent or infrequent item sets, then the item sets that are observed below this minimum support threshold are called infrequent itemsets. On the other hand, the item sets that are observed with a higher value of this minimum support threshold are called frequent itemsets. Both frequent and infrequent itemsets are subsets of a superset and (ii) the second problem is to generate association rules from those frequent itemsets with another constraint of minimal confidence. Association rules are discovered from only the frequent itemsets that are discovered using the minimum support threshold discussed above. Thus, the discovery of a frequent itemset affects the number of discovered association rules. To determine whether an item set is frequent and infrequent, a minimum support threshold must be preset by the user. Otherwise, it is typically not possible to discover neither the frequent itemsets from a dataset nor their corresponding association rules.

Although association rule mining was introduced to extract associations from market basket data [77], association rules are employed today in many other domains such as data analysis, recommender systems, intrusion detection, and web usages mining etc. In the area of mining mobile phone data, recently, a number

of researchers [25, 26, 37] also use association rules for various purposes. Many association rule mining algorithms have been proposed in the data mining literature, such as logic based [78], frequent pattern based [75, 79, 80], tree-based [81] etc. In the following, we mainly focus on some classic and popular association rule mining algorithms, such as AIS [77], Apriori [75], Apriori-based such as Apriori-TID and Apriori-Hybrid [75], FP-Tree [81], and RARM [82] algorithms.

3.3.1.1 AIS Algorithm

The AIS algorithm, proposed by Agrawal et al. [77], is the first algorithm designed for association rule mining. The main focus of this algorithm is to improve the quality of the datasets with the necessary functionalities for findings of associations or relations within this data and to process decision support queries using the discovered associations. In this algorithm, the consequent of the discovered association rules contains only one item, however, the antecedent may contain several items or contexts. An example of such association rule is like $A_1 \cap A_2 \Rightarrow C$, where A_1, A_2 are the items in antecedent and C (one item) represents the consequences of that rule.

In AIS algorithm [77], the frequent itemsets (each context value is considered as an item in the mobile phone dataset) were generated by scanning the datasets several times. A frequent item set satisfies the minimal support. This algorithm works based on several iterations or passes over the dataset. While processing, during the first pass over the dataset, the support count of each context value (item) was accumulated. The context that is infrequent gets eliminated from the list of items. An item is considered infrequent if it has a support count less than its predefined minimum support value according to the preference of the individual user. In such a way, candidate 1-item sets are generated from the dataset. After that, candidate 2-item sets are generated. To do this, this algorithm extends the generated frequent 1-item sets with the remaining other contexts available in the dataset during the second pass over the dataset. After that, similar to the first pass, the infrequent item set is eliminated in the second pass. For this, the algorithm again counts the support value of the generated candidate 2-item sets and checked with the same minimum support threshold that is preferred. The item sets whose support count do not satisfy this predefined threshold are also considered as infrequent item set. Similarly, based on the remaining other contexts in a record of the dataset, the $(n + 1)$ candidate itemsets are generated by extending the frequent n-item sets. The generation of all these candidate item sets and the corresponding frequent itemsets (identified by checking with the minimum support preference) identifying process iterate until any one of them becomes empty. Finally, this algorithm generates association rules based on the frequent itemsets that are identified in different iteration over the dataset.

To make AIS algorithm [77] more efficient, an estimation method was introduced to prune those generated item sets (combination of contexts) that cannot become frequent according to its support value. It not only prunes the unnecessary item sets

candidates but also helps to avoid the corresponding unnecessary effort of counting those item sets. Besides such candidate generation, the memory management in the AIS algorithm is another issue, as all the candidate itemsets and frequent itemsets are assumed to be stored in the main memory. This is important when memory is not enough to store the huge amount of generated candidates. To resolve this issue memory management is also proposed for AIS; (i) to delete the generated candidate itemsets that have never been extended, (ii) to delete the generated candidate itemsets containing the maximal number of items and their siblings, and store this the parent item sets in the disk as a seed for the next pass, which is described with examples in [77].

The main drawback of the AIS algorithm is too many candidate item sets generation and consequently produce a huge number of redundant associations or rules. As a result, it not only produces several useless rules but also requires more space or memory related to such unnecessary generation and wastes much effort that turned out to be useless. At the same time, this algorithm needs too many passes over the whole dataset, which makes the AIS algorithm inefficient for mining mobile phone data to build context-aware real-life applications for mobile phone users.

3.3.1.2 Apriori Algorithm

Apriori proposed by Agrawal in [75] is the most popular algorithm in the area of mining association rules. According to [83], it is a great improvement in the history of association rule mining. The AIS algorithm [77] described above is just a straightforward association generation approach that requires many passes over the dataset, generating many candidate item sets while most of them turn out to be useless. Comparing with the AIS algorithm, Apriori is more efficient during the candidate generation process for two reasons. The first one is Apriori employs a new method deferring from the AIS algorithm, for generating the candidate itemsets (a set of context values), and the second one is it also introduces a new pruning technique for eliminating the infrequent candidates.

To generate all the candidate itemsets (a set of contexts) from a given dataset, there are two processes in Apriori algorithm [75]. Firstly, this algorithm generates the candidate itemsets using the available items in the dataset. After generating these candidates, the support value of the corresponding generated item sets is counted by scanning the dataset. While processing, during the first scanning over the dataset, the support count of each context value (item) is calculated to identify the frequent item set. A frequent itemset satisfies the minimal support preferred by an individual user. This algorithm performs a pruning operation and prunes the infrequent item sets to reduce the burden of further processing. An item is considered infrequent if it has a support count less than its predefined minimum support value set by the preference of an individual user. On the other hand, the item sets that satisfy this threshold are considered as frequent item sets, which are checked in each iteration of the algorithm and generates only those candidate item sets that include the same specified number of items, such as 1-context set, 2-context set, etc. In such a way,

the candidate n-item sets are generated after the $(n-1)$th passes. To do this, this algorithm performs the joining operation with only the frequent $(n-1)$-item sets by counting their corresponding support value over the dataset. To generate these candidate item sets, the Apriori algorithm follows its property, which is defined as "every sub $(n-1)$-item sets of the frequent n-item sets must be frequent" [75]. As such, if any sub $(n-1)$-item sets are not in the list of frequent $(n-1)$-item sets, then the n-item sets candidate is pruned. According to this condition, all the candidate n-item sets are pruned by checking their sub $(n-1)$-item sets during the process as there is no possibility to be frequent according to the Apriori property [75].

In summary, Apriori algorithm [75] avoids the effort wastage of counting the candidate itemsets (a set of contexts) that are already known to be infrequent (not satisfy the support threshold), during the process of identifying frequent itemsets. This algorithm not only generates the candidate itemsets by joining among the frequent itemsets (satisfy the support threshold preferred by an individual) level-wisely but also prunes the candidates according to the Apriori property that is mentioned above. As a result, the number of remaining candidate item sets becomes much smaller, which are ready for further processing. It dramatically reduces the computation, I/O cost and memory requirement comparing with the AIS algorithm [77]. However, the Apriori still has two major drawbacks, of which one has been inherited from the AIS approach. The inherited drawback is that it still has to scan the entire dataset multiple times as it builds the list of frequent itemsets, which eventually produces a huge number of redundant rules. The second drawback is that the candidate generation process is time and memory-consuming and complex that is not effective for mining mobile phone data to build context-aware real-life applications for mobile phone users.

3.3.1.3 Apriori-Based Algorithms

Based on the Apriori algorithm [75], several new association rule mining algorithms were designed with some modifications of this algorithm. For example, Apriori-TID and Apriori-Hybrid [75] are the modifications of the Apriori algorithm.

These algorithms are based on the Apriori algorithm and try to improve the efficiency in terms of execution time by making some modifications. These algorithms try to reduce the number of passes over the dataset and to reduce the size of the dataset to be scanned in every pass for generating the candidate itemsets. Also, these algorithms try to prune the generated candidates by using different techniques.

Apriori-TID [75] extends the original Apriori algorithm by removing the need for multiple scanning of the datasets. This algorithm sets a counter during the first pass through the dataset. This counter is then used later to determine the frequent itemsets. As a result, the original dataset is not needed to counter this. On the other hand, Apriori-Hybrid [75] is based on the idea that is not necessary to use a similar process for each pass over the dataset for generating candidates. This approach combines the advantage of using the Apriori algorithm in the early passes and later this algorithm uses the Apriori-TID algorithm. There is a problem with

this approach is the cost of switching between Apriori and Apriori-TID algorithms. Another algorithm Predictive Apriori proposed by Scheffer [84] generates rules by predicting predictive accuracy combining form support and confidence. So sometimes it produced the rules with large support but low confidence and got unexpected results. These algorithms could give better results in terms of computational time in some cases. However, the problem of redundancy still exists in these modified algorithms.

3.3.1.4 FP-Tree Algorithm

Frequent Pattern Tree (known as FP-Tree) is another association rule mining algorithm, which was introduced by Han et al in [81], to mine frequent patterns like Apriori. FP-Tree is another milestone in the development of mining association rules in terms of execution time. This algorithm needs no candidate generation process like AIS and Apriori algorithm. It can generate frequent itemsets with only two passes over the dataset. The frequent patterns generation process includes two sub-processes: (i) it first constructs the frequent pattern tree, and (ii) then generates the frequent patterns from the tree. By avoiding the candidate generation process and taking less scanning over the dataset, FP-Tree becomes an order of magnitude faster algorithm than the AIS [77] and Apriori [75] algorithm for generating frequent patterns from a given dataset [83].

Three reasons make the FP-Tree algorithm more efficient [83]. First, this algorithm generates a frequent pattern tree, which is a compressed representation of a given dataset. While constructing the tree, only frequent items measured by counting the support value are used. A frequent itemset satisfies the minimal support preferred by an individual user. The other irrelevant information is pruned. This algorithm also does ordering the items (contexts) according to their support values [81]. Secondly, this algorithm only scans the dataset twice. The patterns that satisfy the user-specified minimum threshold are generated by constructing the conditional FP-Tree. To do this, this algorithm uses the concept of the suffix of the patterns. For instance, the conditional FP-Tree contains only the patterns with the specified suffix of the patterns, which reduces the computation cost dramatically. Thirdly, the frequent pattern tree uses a divide and conquer method. It considerably reduces the size of the subsequent conditional tree. This algorithm generates the longer frequent patterns by extending the shorter patterns, i.e., adding a suffix to the shorter frequent patterns [83].

Although the FP-tree does not generate candidates like Apriori, it produces similar outputs for the same dataset. As a result, the problem of producing redundant association rules still exists. Thus, it will not be effective for mining mobile phone data to build context-aware real-life applications for mobile phone users because of its redundant association generation [83].

3.3.1.5 RARM Algorithm

Rapid Association Rule Mining (RARM) is an approach proposed by Das et al. [82] that also uses the tree structure to represent the dataset. This approach also does not utilize a candidate generation process. RARM algorithm uses the SOTrieIT (Support Ordered Trie Itemset) structure for generating the candidate itemsets such as 1-item sets and 2-item sets. It can quickly generate such item sets without generating the candidates and scanning the dataset as well. Similar to the FP-Tree [81] that is discussed above, every node of the SOTrieIT contains one item and the corresponding support value. This algorithm uses the SOTrieIT tree for generating the candidate item-sets [82].

The main focus of this algorithm is faster processing than the existing algorithms. According to [82] RARM is up to 100 times faster than Apriori [75]. However, the problem of producing redundant association rules still exists. Thus, it will not be effective for mining mobile phone data to build context-aware real-life applications for mobile phone users because of its redundant association generation [83].

3.3.1.6 Association Rule Mining Summary

The association rule learning algorithm finds association rules from a dataset that satisfy predefined minimum support and confidence constraints [75]. In the literature on data mining, several association rule learning algorithms have been suggested, such as logic-based [78], frequent pattern based [75, 79, 80], tree-based [81] etc. As it has its parameter support and confidence, the association rule learning technique is well established in terms of rule efficiency, e.g., accuracy and flexibility [85]. A number of researchers [25, 26, 37] have used association rule learning technique (e.g., Apriori)[75] to mine rules capturing mobile phone users' behavior. However, when it comes to discovering users' behavioral rules, association rule learning has some limitations. The disadvantages of association rules for discovering the behavioral rules of individual mobile phone users when taking into account multi-dimensional contexts are summarized below.

- *Lacking in Understanding the Impact of Contexts:* Different contexts in mobile phone data, such as temporal, spatial, or social background, can have varying effects or influence on individual mobile phone users' behavioral rules. Incoming phone calls from a significant person, such as a mother, are often answered by a person, even if she is in a meeting since her family comes first. In this case, the importance of individuals' social relationships ($social\ relationship \rightarrow mother$) in making behavioral decisions is greater than other related contexts such as time, weekday or holiday, place, accompanied with, and so on. However, when discovering rules based on multi-dimensional contexts, the standard association rule learning methodology implicitly assumes that all of the contexts in the datasets have the same nature and/or effect.

- *Redundancy:* If a user preference is defined as a minimum support value and minimum confidence value, an association rule learning technique, such as Apriori, discovers all the contextual associations in a given dataset. As a consequence, the association rule learning methodology generates a large number of redundant rules because it does not consider the usefulness of the associations when creating them. For instance, it produces up to 83% redundant rules for a given dataset that makes the rule-set unnecessarily large [86]. Therefore, it is very difficult to determine the most interesting ones among the huge amount of rules generated. As a result, it makes the rule-based decision-making process ineffective and more complex, which is not effective to build a context-aware intelligent system [87].
- *Computational Complexity and High Training Time:* The association rule learning method necessitates a significant amount of preparation time to generate rules. For example, when the association rule learning algorithm is used to discover user behavioral rules, the authors find a long-running period spanning many hours in an experimental study in the cell phone domain [37]. The key explanation for the long training period is that traditional association methods compute all possible correlations between contexts and are unable to filter out the useful rules for decision-making. As a consequence, generating patterns that aren't required adds to the computational complexity and training time.

In summary, traditional association rule learning techniques may not be appropriate to generate users' behavioral rules in multi-dimensional environments, to create intelligent context-aware systems, when taking into account the effect of contexts, the redundancy issue while producing rules, and computational complexity.

3.3.2 Classification Rules

Another method for extracting user behavioral rules from datasets is classification. Classification is another tool for discovering rules in the field of data mining, where A represents contextual information and C represents the corresponding activity class. In general, classification is classified as a learning method for mapping (classifying) a data instance into the dataset's predefined class labels. According to [88], data classification is a two-step process; (i) is the learning stage, in which a classification model is built from a given dataset; the training set is the data from which a classification feature or model is learned, and (ii) second one is a classification step, in which the model is used to evaluate or predict class labels for previously unknown data; the testing set is the data set used to test the learned model's or function's classification ability.

Classifier efficiency is normally determined by accuracy, which is the percentage of correct predictions over the total number of predictions made for a given test dataset, according to [88]. Many other metrics, such as sensitivity, error rate, specificity, precision, recall, and f-measure, are also used to understand the various

aspects of the generated model. In the data mining literature, several classification algorithms with rule generation capabilities have been proposed. In the following, we mainly focus on some basic and popular approaches such as ZeroR [89], OneR [90], Decision Tree [76], PART [91], and RIDOR [89].

3.3.2.1 Zero-R

Among the classification techniques used in the data mining field, Zero-R is the most basic [89]. This algorithm only considers the objective and disregards all predictors. The majority group is predicted by the Zero-R classifier (class). For example, a Zero-R model for a sample phone call dataset may be "$behavior \rightarrow reject$". Although Zero-R has no predictability capacity, it can be used to establish a baseline output for other classification methods [89].

3.3.2.2 One-R

Holte et al. [90] proposed One-R, which is a simple and inexpensive classification method. One-R, short for "One Rule," is a straightforward but precise classification algorithm for generating the predicting rule. In this method, a one-level decision tree is built from the training records, and the rules are extracted from that tree, which is connected to frequently occurring classes in the dataset. Humans can easily interpret and comprehend the rules produced by the One-R approach. This algorithm creates a frequency table for each predictor against the target to generate a rule for it. It then produces one rule for each predictor in the data and chooses the rule with the smallest total error as its "one rule". If a user is in a meeting, for example, the phone call action is rejected. One-R has been shown to generate rules that are only marginally less reliable than state-of-the-art classification algorithms while still being easy to understand by humans [90].

3.3.2.3 RIDOR

A direct classification tool is the Ripple Down Rule learner (RIDOR) [89]. The default rule and the additional rules of that default rule are created by this algorithm. The algorithm produces a default rule by evaluating the dataset first, according to this principle. Following that, it generates a set of additional rules. To do this, the algorithm measures the error rate and selects the rules with the lowest error rate. These created additional rules are used in addition to the default rule to predict the unseen classes for a given condition.

3.3.2.4 Decision Tree

Decision trees [92] is a well-known and widely discussed technique for classification and prediction. A decision tree is a graph that illustrates the possible outcome of a decision using a branching process. Each branch of a decision tree represents a choice among a set of options related to the context of attribute values, and each leaf node represents a classification or decision for that choice. The decision tree algorithm aims to derive classification rules from instance learning. We'll go through some simple decision tree algorithms in this section. Various algorithms exist in the implementation of speed, scalability, performance intelligibility, classification accuracy, and other factors, each with its own set of advantages.

J. R. Quinlan suggested ID3 as the central algorithm for creating decision trees [92]. The ID3 algorithm builds a decision tree by using a top-down approach in which each attribute or context is tested at each node using a greedy search through the specified training dataset. It determines the entropy and information gain, which is a statistical property used to determine which attribute to measure at each node in the tree [92]. The degree to which a given attribute distinguishes training examples according to their target classification is measured by information gain. For both missing values and continuous-valued attributes, the ID3 algorithm does not produce sufficient results. The values for a continuous attribute should be mapped to some discrete representation to improve ID3 efficiency.

Quinlan proposes a modified algorithm, the C4.5 algorithm, based on the ID3 algorithm [76]. C4.5 uses the principle of knowledge benefit to construct decision trees from a training dataset in a similar manner to ID3. For splitting the dataset into subsets, the C4.5 decision tree algorithm uses gain ratio as the test attribute selection criterion, and each time selects the attribute with the highest knowledge gain ratio as the test attribute for a given set. C4.5 is a statistical classifier that can deal with both numeric and missing value attributes, is robust in the presence of noise, and can build trees with large branches and scales.

The CART algorithm tends to simplify and increase the performance of the decision tree [93]. CART will deal with both order and disorder data in addition to multi-state numerical data. It chooses the test attribute based on the Gini coefficient and creates a binary tree with a straightforward structure. Following that, the SLIQ [94] and SPRINT [95] algorithms are proposed solely to improve scalability and parallelism.

3.3.2.5 Hybrid Classification

In [91], PART was proposed as a hybrid classification algorithm. This algorithm generates rules using a rule induction method in addition to the decision tree. Instead of using these two algorithms in two steps, this algorithm combines them all into one. Even though this algorithm creates a decision tree, it does not create a complete decision tree. A partial decision tree is built using a divide and conquer method in

PART. A rule induction procedure is used to produce the candidate rules. After that, this algorithm employs a pruning process to filter the intended rules. DTNB, proposed by Sheng et al. [96], is another hybrid classification technique for creating classifications, similar to the PART algorithm. It employs a decision table as well as a Naive Bayes classifier. The generated rules can be used to predict previously unknown classes in a given situation.

3.3.2.6 Classification Rule Mining Summary

One of the most common rule-based classification algorithms is decision tree, which has several advantages, including being easier to interpret, being able to handle high-dimensional data, being simple and quick, being accurate, and being able to generate human-understandable classification rules [97, 98]. A number of authors [36, 99–101], in particular, have used the decision tree classification method to find rules capturing cell phone users' actions for different purposes. However, to model users' actions, classification rules have some limitations. The disadvantages of rule-based classification strategies for discovering the behavioral rules of individual cell phone use are summarized below.

- *Low-Reliability:* In general, reliability refers to the consistency of being dependable or continuously performing well. If the pattern or rule describes a relationship that happens in a high percentage of relevant situations, it is said to be accurate. A classification rule will be reliable if it provides high prediction accuracy, and an association rule will be reliable if it has high confidence correlated with the accuracy, according to Geng et al. [102]. However, in many situations, the classification rules discovered by traditional rule-based classification methods, such as decision trees, have poor reliability [25, 103]. A classification rule does not guarantee high accuracy in forecasts, according to Freitas et al. [85]. The explanation for this is that it may have an over-fitting problem and inductive bias, both of which reduce the accuracy of a machine learning-based model's prediction.
- *Lacking in Flexibility:* Traditional rule-based classification techniques, such as decision trees, do not allow users to set their preferences, and as a result, they make rigid decisions for each test case [76]. However, when considering real-world scenarios, static decisions in modeling user actions can not be meaningful. The explanation for this is that people's preferences aren't always consistent; they can differ from one user to another [104]. For example, one person will wish for the phone call agent to reject incoming calls if she has not answered them more than 80% of the time in the past. This preference could be 95% of the time for another person, depending on her preferences.
- *Lacking in Generalization:* Typically, generality refers to the extent to which a pattern or rule is systematic, i.e., the percentage of all applicable records in the dataset that match the pattern. According to Geng et al. [102], a pattern is more useful and interesting if it characterizes more details in the related

dataset. When creating classification rules, traditional classification strategies take data-driven generalization into account. Aside from that, users' behavior-based generalization might be of concern. Users' actions, for example, can be consistent in a variety of situations, with just a few exceptions [105]. As a result, behavior-oriented generalization may provide more accurate results for modeling users' use patterns. The generalization process not only simplifies the resulting machine learning-based model but also reduces overfitting and increases prediction accuracy.

Neither association rule mining (e.g., Apriori) [75] nor classification rule mining (e.g., Decision tree) [76] are ideal for discovering behavioral rules of cell phone users based on multi-dimensional contexts. As a result, in this book we suggest a behavioral decision tree-based approach that generates not only useful general rules for capturing individual actions at a given level of confidence with a small number of contexts but also rules that articulate unique exceptions to the general rules when more context-dimensions are considered.

3.4 Incremental Learning and Updating

Mobile phone log data is not static; it is gradually applied day by day based on an individual's current (on-going) mobile phone habits. Since people's behavior changes over time, the more recent trends are more likely to be important and relevant for predicting people's potential behavior in specific situations than older ones [106, 107]. As a result, updating and complex management of discovered rules based on individuals' recent behavioral trends (e.g., recency) becomes a challenge, as changes can not only invalidate certain current rules but also make other rules relevant.

Several incremental rule mining techniques have been proposed for mining rules in a complex database in the field of data mining. To get a fully updated set of rules, these techniques use existing rules and the incremental portion of the dataset. For example, Cheung et al. [108] proposed the FUP algorithm, which is the first incremental updating technique for preserving association rules as new data is added into the database. The FUP algorithm is used to discover new frequent itemsets in a complex database and is based on the Apriori [75] algorithm.

By deleting earlier itemsets that are either considered to be still frequent or deemed infrequent only by testing the incremental section, FUP tries to extract the value from the previously discovered rules to produce a relatively small candidate set to be tested against the original data set. Cheung, et al. suggested a new algorithm FUP2, which is an expansion of the FUP algorithm, in [109]. When new transactions are added to a database, the FUP updates the association rules, while the FUP2 extracts the rules from the final data collection, taking into account both the removed and newly added parts. If the data set is only modified by insertions, the FUP2 algorithm would behave similarly to FUP.

Xu et al. [110] suggest another incremental association rule mining algorithm. They suggest an IFP-tree technique, which is an extension of the FP-tree technique [81]. When a data set is incremented, it uses the current FP-tree to construct the IFP-tree. It updates the support of the related nodes of the tree for each transaction of the incremental part. If, on the other hand, a new node is formed, it is held as a new branch of the root node. The frequent itemsets are discovered after the tree has been built. It does so by contrasting both old and new trees. When a new node in the updated tree is discovered, frequent itemsets are created. Since each new node in the tree forms its branch, only the new branches can result in the generation of a new frequent object. However, as the number of dimensions and transactions grows, its efficiency suffers. Thomas et al. propose an algorithm based on the idea of Negative Border that preserves both frequent and border itemsets [111]. The algorithm updates to support counts of all frequent itemsets and border itemsets as data is added to or removed from the original database. However, to minimize scanning times of an initial database, a large number of border itemsets must be stored in memory.

A few algorithms are proposed based on the three-way decision, which is an extension of the widely used binary-decision model with an optional third alternative [112, 113]. The concepts of approval, rejection and no engagement suggested by Yao [114] are used to build a three-way decision theory. All itemsets are divided into three regions using these methods, namely the positive, the boundary, and the negative region. Positive itemsets are already popular, and they consider them without reservation. Itemsets in the boundary region are uncommon, but they could become more common shortly after data increment. Even after data increment, itemsets in the negative region will become less common, and they will be abandoned. As a result, all that is required to keep the frequently used itemsets up to date is to review those in the boundary zone. The runtime is saved because the negative region containing the majority portion is never computed.

Amornchewin et al. suggest a probability-based incremental association rule discovery algorithm in [115]. To avoid reprocessing entire dynamic databases, this algorithm uses the Bernoulli trials principle and uses previously mined information. When only new data is inserted into a dynamic database, the algorithm can efficiently maintain association rules. Thusaranon et al. [116] propose a new probability-based incremental association rule discovery algorithm that is a development of Amornchewin and Kreesuradej's [115] algorithm. They expand the algorithm's ability to maintain association rules of a complex database in the case of record insertion and deletion at the same time in this algorithm.

The above incremental mining techniques primarily consider the overall mining process's faster processing, e.g., performance. Instead of processing the combined dataset that includes the initial dataset and the incremental portion, these techniques minimize scanning on the provided datasets by mining the incremental part separately. As a result, the overall mining method with conventional updating techniques affects the amount of time it takes to find a full set of revised rules. However, the freshness of rules, such as rules based on recent trends, is important in modeling users' behavior, and this has not been taken into account in these techniques. The

explanation for this is that in the real world, user behavior is not static and can change over time. To effectively model smartphone users' behavior in relevant multi-dimensional contexts, updation in terms of freshness in users' behavior while generating rules is needed.

Several researchers use recent cell phone log data behavioral patterns to forecast future actions rather than patterns extracted from the entire historical logs to generate rules based on an individual's current behavior. They do, however, use a static period of recent historical data, which may be insufficient for determining users' recent behavioral rules. For example, Lee et al. [117] have used recent call list data to study cell phone users' calling patterns and create a call recommendation algorithm for an adaptive speed-call list. To achieve their target, they extract call logs from the previous 3 months. Barzaiq et al. [118] suggest an approach that analyzes cell phone historical data over a 2-year cycle to forecast outgoing calls and observe relatively additional computational load that appears to be unnecessary. Phithakkitnukoon et al. [119], conduct their research on reality mining datasets collected over 9 months and find that only a recent portion of contact history is more important. Phithakkitnukoon et al. [65] present a model for forecasting phone calls for the next 24 h based on the users' previous contact history in a separate paper. They demonstrated that the recent trend of the user's calling pattern is more important than the older one and has a higher correlation to the future pattern than the pattern generated from all historical data in their approach. As a result, to improve prediction accuracy, the most recent 60 days of call records in the call logs are considered to be the potential observed call activities [65]. However, since users' actions are not consistent in the real world and may differ from user to user over time, such a static period consideration may not be appropriate to represent one's current behavior.

Apart from these methods, several authors [120, 121] deal with the issue of handling personal information, such as individual's contact lists in their mobile phone, and more precisely, the task of searching for the desired contact number when making an outgoing call. According to Bergman et al. [120], a large number of contacts in cell phones are never used, even though contact lists grow larger. According to their findings, 47% of the users' contacts had not been used in over 6 months or had never been used at all. Stefanis et al. [121] used a window-based model for handling and searching personal information on mobile phones to predict future actions. They demonstrated in their experiment that the training window for predicting an individual's cell phone use behavior should be long enough to provide enough data. A training window of more than 2 weeks, on the other hand, would be unable to capture the dynamic changes in phone call behavior patterns. Furthermore, a training window of fewer than 7 days will be insufficient to capture behavioral changes through all days of the week, including changes in social circumstances on weekends. In conclusion, standard updating techniques discussed above may not be appropriate for producing a full collection of users' behavioral rules in multi-dimensional contexts, to develop intelligent context-aware systems, to provide relevant services to end smartphone users, when taking into account the freshness in rules representing users' current actions and their dynamic updation.

3.5 Identifying the Scope of Research

Mobile phones have become an inseparable part of our lives. As mentioned in this chapter, not all users' cell phone use behaviors are the same and can vary from one user to another based on contextual details. In a context-aware pervasive computing environment, studies have demonstrated that mining contextual behavior rules of individual cell phone users will intelligently assist them in various context-aware personalized systems such as smart call interruption management system, smart call reminder system, mobile notification management system, context-aware mobile app recommender systems, and various predictive services.

Users do not want to use apps that require individual cell phone users to identify and maintain their behavioral rules manually rather than automatically discovered, according to studies. Users may not have the time, inclination, experience, or interest to manually maintain rules [122]. Mining behavioral rules of individual cell phone users based on relevant multi-dimensional contexts, according to individual preferences, is a key prerequisite for developing such smart applications.

The state-of-the-art in the field of mobile data analytics was addressed in this chapter. Based on this, we have summarized the scope of research below, which are taken into account throughout this book:

(i) Contextual data pre-processing and feature selection are the primary parts of an effective context-aware system. In Chap. 4, the basic feature selection and extraction methods for efficient processing have been provided. We also present several contextual datasets that can be utilized to build a machine learning based context-aware model for corresponding mobile applications and services in this chapter. As the real-world data may contain noisy and inconsistency instances, the pre-processing steps have also been analyzed to clean and remove noises from raw data in this chapter.

(ii) Context discretization, e.g., an effective time-series modeling considering mobile user behavioral activities is still lacking for building an intelligent context-aware system. In Chap. 5, we present a behavior-oriented time segmentation technique capturing user behavioral patterns to produce temporal behavioral rules. Using time-series cell phone data, this method dynamically considers not only the temporal coverage of the week but also the number of incidences of various behaviors to produce related behavioral time segments over the week.

(iii) Existing studies have focused on mainly association rule mining techniques (e.g., Apriori) or classification rule mining techniques (e.g., Decision tree) for discovering user behavioral rules utilizing mobile phone data. However, there is still a lack of discovering the useful behavioral rules of individual mobile phone users based on multi-dimensional contexts. In Chap. 6, we present a tree based approach to model an individual's mobile phone usage behavior utilizing their mobile phone data. This approach produces not only the general rules that capture an individual's behavior at a particular level of confidence

with a minimal number of contexts but also produces rules that express specific exceptions to the general rules when more context dimensions are taken into account.

(iv) Another key limitation of the existing literature is that the previous studies have not considered the recency-based rules of individual mobile phone users and updating the existing rules based on the recent behavioral patterns of individuals. In other words, there is a lack of understanding about rules that reflect an individual's recent behavioral patterns ignoring the user's past behavioral patterns to get a complete set of updated rules. In Chap. 7, we present an approach for recency-based updating the rules and their dynamic management. This updated rules-set not only contains all the useful rules of an individual mobile phone user for the whole log period but also expresses recent behavioral patterns that will help model mobile phone usage behavior of individuals to provide personalized services for the end mobile phone users in a context-aware pervasive computing environment.

(v) A rule-based expert system modeling is typically considered one of the key AI techniques that can be used to make intelligent decisions and more powerful applications. In Chap. 8, we discuss mobile expert system as a knowledge or rule-based modeling, where a set of context-aware rules are extracted from mobile data discussed in earlier chapters. Usually, the purpose of the expert system is to take information from a human expert and turn this into a number of hardcoded rules for the input data to be implemented. In this chapter, we focus on the generated rules based on machine learning techniques, rather than the hardcoded rules, as we take into account the dynamism in the context-aware rules.

(vi) Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Although, the rule-based machine learning methods performed well, deep learning can be used, when a large amount of data is available. In Chap. 9, we discuss the importance of deep learning and a context-aware deep learning model for mobile phone users.

(vii) Finally, in Chap. 10, we highlight the most important and vital issues, ranging from contextual data collection to decision-making, that has been thoroughly explored in this book. In terms of new researchers' perspectives, future advances in industries, and smart solutions in the context-aware technology industry, prospective research works, and challenges in the field of context-aware computing have been addressed.

Overall, this book presents a variety of techniques and research scope in the field of context-aware machine learning and data analytics, which can be used for building a variety of real-world applications. The prominent application fields are personalized assistance services, recommendation systems, human-centric computing, adaptive and intelligent systems, IoT services, smart cities, mobile privacy and security systems, and many others, where applications dynamism based on contextual data is needed.

3.6 Conclusion

The effect of multi-dimensional contexts on smartphone data, both in terms of context-aware rule learning and the dataset itself, was addressed in this book. In this chapter, we mainly aimed to address the current state of mobile data analytics and associated rule learning techniques. We have also explored about how multi-dimensional contexts including temporal, spatial, and social contexts can influence such technology. We have summarized applicable research for each popular technique to help others in the context-aware rule learning community. In terms of time-series modeling, rule discovery based on multi-dimensional contexts, and updating rules over time according to individual preferences, we have addressed various issues regarding context-aware rule learning. In terms of current research, there has been a lot of emphasis on conventional context-aware systems and techniques, with less work on machine learning rule-based context-aware systems for successful decision making in a specific domain.

Overall, we reviewed previous research and addressed a discussion of challenges and potential directions for learning context-aware rules from smartphone data in this chapter. The domain-specific context-aware rules can be used to create a variety of context-aware models that intelligently assist end-users in their daily activities. At the end of this chapter, we have summarized the scope of research, which are taken into account throughout this book. We also assume that this study can be used as a reference guide in the relevant application areas including mobile applications, smart systems and security, etc. for both academia and industry.

References

1. Dourish, P. (2004). What we talk about when we talk about context. *Personal and Ubiquitous Computing, 8*(1), 19–30.
2. Schilit, B. N., & Theimer, M. M. (1994). Disseminating active map information to mobile hosts. *IEEE network, 8*(5), 22–32.
3. Brown, P. J., Bovey, J. D., & Chen, X. (1997). Context-aware applications: From the laboratory to the marketplace. *IEEE personal communications, 4*(5), 58–64.
4. Ryan, N. S., Pascoe, J., & Morse, D. R. (1998). Enhanced reality fieldwork: The context-aware archaeological assistant. In *Computer applications in archaeology*. Tempus Reparatum.
5. Brown, P. J. (1995). The stick-e document: A framework for creating context-aware applications. *Electronic Publishing-Chichester-, 8*, 259–272.
6. Franklin, D., & Flaschbart, J. (1998, March). All gadget and no representation makes Jack a dull environment. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments* (pp. 155–160).
7. Ward, A., Jones, A., & Hopper, A. (1997). A new location technique for the active office. *IEEE Personal communications, 4*(5), 42–47.
8. Hull, R., Neaves, P., & Bedford-Roberts, J. (1997, October). Towards situated computing. In *Digest of papers. First international symposium on wearable computers* (pp. 146–153). IEEE.
9. Rodden, T., Cheverst, K., Davies, K., & Dix, A. (1998, May). Exploiting context in HCI design for mobile systems. In *Workshop on human computer interaction with mobile devices* (Vol. 12).

10. Schilit, B., Adams, N., & Want, R. (1994, December). Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications* (pp. 85–90). IEEE.
11. Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing, 5*(1), 4–7.
12. Chang, Y. J., & Tang, J. C. (2015, August). Investigating mobile users' ringer mode usage and attentiveness and responsiveness to communication. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 6–15).
13. Halvey, M., Keane, M. T., & Smyth, B. (2006, April). Time based patterns in mobile-internet surfing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 31–34).
14. Gandhi, S., Oates, T., Boedihardjo, A., Chen, C., Lin, J., Senin, P. & Wang, X. (2015, October). A generative model for time series discretization based on multiple normal distributions. In *Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management* (pp. 19–25).
15. Farrahi, K., & Gatica-Perez, D. (2014). A probabilistic approach to mining mobile phone data sequences. *Personal and Ubiquitous Computing, 18*(1), 223–238.
16. Zhang, G., Liu, X., & Yang, Y. (2014). Time-series pattern based effective noise generation for privacy protection on cloud. *IEEE Transactions on Computers, 64*(5), 1456–1469.
17. Song, Y., Ma, H., Wang, H., & Wang, K. (2013, May). Exploring and exploiting user search behavior on mobile and tablet devices to improve search relevance. In *Proceedings of the 22nd International Conference on World Wide Web* (pp. 1201–1212).
18. Rawassizadeh, R., Momeni, E., Dobbins, C., Gharibshah, J., & Pazzani, M. (2016). Scalable daily human behavioral pattern mining from multivariate temporal data. *IEEE Transactions on Knowledge and Data Engineering, 28*(11), 3098–3112.
19. Mukherji, A., Srinivasan, V., & Welbourne, E. (2014, September). 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* (pp. 1005–1014).
20. Bayir, M. A., Demirbas, M., & Cosar, A. (2011). A web-based personalized mobility service for smartphone applications. *The Computer Journal, 54*(5), 800–814.
21. Paireekreng, W., Rapeepisarn, K., & Wong, K. W. (2009). Time-based personalised mobile game downloading. In *Transactions on edutainment II* (pp. 59–69). Berlin, Heidelberg: Springer.
22. Jayarajah, K., Kauffman, R., & Misra, A. (2014, September). Exploring variety seeking behavior in mobile users. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication* (pp. 385–390).
23. Do, T. M. T., & Gatica-Perez, D. (2010, December). By their apps you shall understand them: Mining large-scale patterns of mobile phone usage. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia* (pp. 1–10).
24. Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z.,& Choudhury, T. (2013, September). 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).
25. Mehrotra, A., Hendley, R., & Musolesi, M. (2016, September). PrefMiner: Mining user's preferences for intelligent mobile notification management. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 1223–1234).
26. Zhu, H., Chen, E., Xiong, H., Yu, K., Cao, H., & Tian, J. (2014). Mining mobile user preferences for personalized context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology, 5*(4), 1–27.
27. Oulasvirta, A., Rattenbury, T., Ma, L., & Raita, E. (2012). Habits make smartphone use more pervasive. *Personal and Ubiquitous Computing, 16*(1), 105–114.

28. Yu, K., Zhang, B., Zhu, H., Cao, H., & Tian, J. (2012, May). Towards personalized context-aware recommendation by mining context logs through topic models. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 431–443). Berlin, Heidelberg: Springer.
29. Naboulsi, D., Stanica, R., & Fiore, M. (2014, April). Classifying call profiles in large-scale mobile traffic datasets. In *IEEE INFOCOM 2014-IEEE conference on computer communications* (pp. 1806–1814). IEEE.
30. Dashdorj, Z., Serafini, L., Antonelli, F., & Larcher, R. (2013, December). Semantic enrichment of mobile phone data records. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia* (pp. 1–10).
31. Shin, D., Lee, J. W., Yeon, J., & Lee, S. G. (2009, July). Context-aware recommendation by aggregating user context. In *2009 IEEE conference on commerce and enterprise computing* (pp. 423–430). IEEE.
32. Shin, C., Hong, J. H., & Dey, A. K. (2012, September). Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 173–182).
33. Farrahi, K., & Gatica-Perez, D. (2010). Probabilistic mining of socio-geographic routines from mobile phone data. *IEEE Journal of Selected Topics in Signal Processing, 4*(4), 746–755.
34. Ma, H., Cao, H., Yang, Q., Chen, E., & Tian, J. (2012, April). A habit mining approach for discovering similar mobile users. In *Proceedings of the 21st International Conference on World Wide Web* (pp. 231–240).
35. Cao, H., Bao, T., Yang, Q., Chen, E., & Tian, J. (2010, October). An effective approach for mining mobile user habits. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 1677–1680).
36. Zulkernain, S., Madiraju, P., Ahamed, S. I., & Stamm, K. (2010). A mobile intelligent interruption.
37. Srinivasan, V., Moghaddam, S., Mukherji, A., Rachuri, K. K., Xu, C., & Tapia, E. M. (2014, September). Mobileminer: Mining your frequent patterns on your phone. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 389–400).
38. Parate, A., Böhmer, M., Chu, D., Ganesan, D., & Marlin, B. M. (2013, September). Practical prediction and prefetch for faster access to applications on mobile phones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and Ubiquitous Computing* (pp. 275–284).
39. Myllärniemi, V., Korjus, O., Raatikainen, M., Norja, T., & Männistö, T. (2014, November). Meeting scheduling across heterogeneous calendars and organizations utilizing mobile devices and cloud services. In *Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia* (pp. 224–227).
40. Alexiadis, A., & Refanidis, I. (2009, April). Defining a task's temporal domain for intelligent calendar applications. In *IFIP international conference on artificial intelligence applications and innovations* (pp. 399–406). Boston, MA: Springer.
41. Sarker, I. H., Colman, A., Han, J., Kayes, A. S. M., & Watters, P. (2020). CalBehav: A machine learning-based personalized calendar behavioral model using time-series smartphone data. *The Computer Journal, 63*(7), 1109–1123.
42. Khalil, A., & Connelly, K. (2005, July). Context-aware Configuration: A study on improving cell phone awareness. In *International and interdisciplinary conference on modeling and using context* (pp. 197–209). Berlin, Heidelberg: Springer.
43. Salovaara, A., Lindqvist, A., Hasu, T., & Häkkilä, J. (2011, August). The phone rings but the user doesn't answer: Unavailability in mobile communication. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 503–512).

44. Dekel, A., Nacht, D., & Kirkpatrick, S. (2009, September). Minimizing mobile phone disruption via smart profile management. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 1–5).
45. Zulkernain, S., Madiraju, P., & Ahamed, S. I. (2010, June). A context aware interruption management system for mobile devices. In *International conference on mobile wireless middleware, operating systems, and applications* (pp. 221–234). Berlin, Heidelberg: Springer.
46. Seo, S. S., Kwon, A., Kang, J. M., Strassner, J., & Hong, J. W. K. (2011, June). Pyp: Design and implementation of a context-aware configuration manager for smartphones. In *Proceedings of the 1st International Workshop on Smart Mobile Applications (SmartApps 11)*.
47. Khalil, A., & Connelly, K. (2005, September). Improving cell phone awareness by using calendar information. In *IFIP Conference on human-computer interaction* (pp. 588–600). Berlin, Heidelberg: Springer.
48. De Guzman, E. S., Sharmin, M., & Bailey, B. P. (2007, May). Should I call now? Understanding what context is considered when deciding whether to initiate remote communication via mobile devices. In *Proceedings of Graphics Interface 2007* (pp. 143–150).
49. Grandhi, S., & Jones, Q. (2010). Technology-mediated interruption management. *International Journal of Human-Computer Studies, 68*(5), 288–306.
50. Rosenthal, S., Dey, A. K., & Veloso, M. (2011, June). Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *International conference on pervasive computing* (pp. 170–187). Berlin, Heidelberg: Springer.
51. Lovett, T., O'Neill, E., Irwin, J., & Pollington, D. (2010, September). The calendar as a sensor: analysis and improvement using data fusion with social networks and location. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing* (pp. 3–12).
52. Sarker, I. H., Colman, A., Kabir, M. A., & Han, J. (2018). Individualized time-series segmentation for mining mobile phone user behavior. *The Computer Journal, 61*(3), 349–368.
53. Ozer, M., Keles, I., Toroslu, H., Karagoz, P., & Davulcu, H. (2016). Predicting the location and time of mobile phone users by using sequential pattern mining techniques. *The Computer Journal, 59*(6), 908–922.
54. Do, T. M. T., & Gatica-Perez, D. (2014). Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing, 12*, 79–91.
55. Farrahi, K., & Gatica-Perez, D. (2008, October). What did you do today? Discovering daily routines from large-scale mobile data. In *Proceedings of the 16th ACM International Conference on Multimedia* (pp. 849–852).
56. Karatzoglou, A., Baltrunas, L., Church, K., & Böhmer, M. (2012, October). Climbing the app wall: Enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management* (pp. 2527–2530).
57. Phithakkitnukoon, S., Horanont, T., Di Lorenzo, G., Shibasaki, R., & Ratti, C. (2010, August). Activity-aware map: Identifying human daily activity pattern using mobile phone data. In *International workshop on human behavior understanding* (pp. 14–25). Berlin, Heidelberg: Springer.
58. Halvey, M., Keane, M. T., & Smyth, B. (2005, September). Time-based segmentation of log data for user navigation prediction in personalization. In *The 2005 IEEE/WIC/ACM international conference on web intelligence (WI'05)* (pp. 636–640). IEEE.
59. Das, G., Lin, K. I., Mannila, H., Renganathan, G., & Smyth, P. (1998, August). Rule discovery from time series. In *KDD* (Vol. 98, No. 1, pp. 16–22).
60. Lu, E. H. C., Tseng, V. S., & Philip, S. Y. (2010). Mining cluster-based temporal mobile sequential patterns in location-based service environments. *IEEE Transactions on Knowledge and Data Engineering, 23*(6), 914–927.
61. Kandasamy, K., & Kumar, C. S. (2015). Modified PSO based optimal time interval identification for predicting mobile user behaviour (MPSO-OTI2-PMB) in location based services. *Indian Journal of Science and Technology, 8*, 185.

62. Hartono, R. N., Pears, R., Kasabov, N., & Worner, S. P. (2014, July). Extracting temporal knowledge from time series: A case study in ecological data. In *2014 international joint conference on neural networks (IJCNN)* (pp. 4237–4243). IEEE.
63. Keogh, E., Chu, S., Hart, D., & Pazzani, M. (2004). Segmenting time series: A survey and novel approach. In *Data mining in time series databases* (pp. 1–21).
64. Shokoohi-Yekta, M., Chen, Y., Campana, B., Hu, B., Zakaria, J., & Keogh, E. (2015, August). Discovery of meaningful rules in time series. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1085–1094).
65. Phithakkitnukoon, S., Dantu, R., Claxton, R., & Eagle, N. (2011). Behavior-based adaptive call predictor. *ACM Transactions on Autonomous and Adaptive Systems, 6*(3), 1–28.
66. Jang, B. R., Noh, Y., Lee, S. J., & Park, S. B. (2015, February). A combination of temporal and general preferences for app recommendation. In *2015 international conference on big data and smart computing (BigComp)* (pp. 178–185). IEEE.
67. Henze, N., & Boll, S. (2011, August). Release your app on Sunday eve: Finding the best time to deploy apps. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 581–586).
68. Xu, Q., Erman, J., Gerber, A., Mao, Z., Pang, J., & Venkataraman, S. (2011, November). Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference* (pp. 329–344).
69. Böhmer, M., Hecht, B., Schöning, J., Krüger, A., & Bauer, G. (2011, August). Falling asleep with angry birds, facebook and kindle: A large scale study on mobile application usage. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 47–56).
70. Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks, 16*(3), 645–678.
71. MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Vol. 1, No. 14, pp. 281–297).
72. Rokach, L. (2009). A survey of clustering algorithms. In *Data mining and knowledge discovery handbook* (pp. 269–298). Boston, MA: Springer.
73. Sneath, P. H. (1957). The application of computers to taxonomy. *Microbiology, 17*(1), 201–226.
74. Sorenson, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species content. *K Dan Vidensk Selsk Biol Skr, 5*, 1–34.
75. Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB* (Vol. 1215, pp. 487–499).
76. Quinlan, J. R. (2014). *C4. 5: Programs for machine learning*. Elsevier.
77. Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (pp. 207–216).
78. Flach, P. A., & Lachiche, N. (2001). Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning, 42*(1), 61–95.
79. Houtsma, M., & Swami, A. (1995, March). Set-oriented mining for association rules in relational databases. In *Proceedings of the Eleventh International Conference on Data Engineering* (pp. 25–33). IEEE.
80. Ma, B. L. W. H. Y., Liu, B., & Hsu, Y. (1998, August). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*.
81. Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM SIGMOD Record, 29*(2), 1–12.
82. Das, A., Ng, W. K., & Woon, Y. K. (2001, October). Rapid association rule mining. In *Proceedings of the Tenth International Conference on Information and Knowledge Management* (pp. 474–481).

83. Zhao, Q., & Bhowmick, S. S. (2003). *Association rule mining: A survey* (Vol. 135). Singapore: Nanyang Technological University.
84. Scheffer, T. (2005). Finding association rules that trade support optimally against confidence. *Intelligent Data Analysis, 9*(4), 381–395.
85. Freitas, A. A. (2000). Understanding the crucial differences between classification and discovery of association rules: A position paper. *ACM SIGKDD Explorations Newsletter, 2*(1), 65–69.
86. Fournier-Viger, P., & Tseng, V. S. (2012, December). Mining top-k non-redundant association rules. In *International symposium on methodologies for intelligent systems* (pp. 31–40). Berlin, Heidelberg: Springer.
87. Bouker, S., Saidi, R., Yahia, S. B., & Nguifo, E. M. (2012, November). Ranking and selecting association rules based on dominance relationship. In *2012 IEEE 24th international conference on tools with artificial intelligence* (Vol. 1, pp. 658–665). IEEE.
88. Han, J., Kamber, M., & Pei, J. (2011). Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems, 5*(4), 83–124.
89. Witten, I. H., & Frank, E. (2002). Data mining: Practical machine learning tools and techniques with Java implementations. *ACM SIGMOD Record, 31*(1), 76–77.
90. Holte, R. C. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning, 11*(1), 63–90.
91. Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization.
92. Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.
93. Lewis, R. J. (2000, May). An introduction to classification and regression tree (CART) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California* (Vol. 14).
94. Mehta, M., Agrawal, R., & Rissanen, J. (1996, March). SLIQ: A fast scalable classifier for data mining. In *International conference on extending database technology* (pp. 18–32). Berlin, Heidelberg: Springer.
95. Shafer, J., Agrawal, R., & Mehta, M. (1996, September). SPRINT: A scalable parallel classifier for data mining. In *Vldb* (Vol. 96, pp. 544–555).
96. Sheng, S., & Ling, C. X. (2005, October). Hybrid cost-sensitive decision tree. In *European conference on principles of data mining and knowledge discovery* (pp. 274–284). Berlin, Heidelberg: Springer.
97. Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H. & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems, 14*(1), 1–37.
98. Wu, C. C., Chen, Y. L., Liu, Y. H., & Yang, X. Y. (2016). Decision tree induction with a constrained number of leaf nodes. *Applied Intelligence, 45*(3), 673–685.
99. Hong, J., Suh, E. H., Kim, J., & Kim, S. (2009). Context-aware system for proactive personalized service based on context history. *Expert Systems with Applications, 36*(4), 7448–7457.
100. Lee, W. P. (2007). Deploying personalized mobile services in an agent-based environment. *Expert Systems with Applications, 32*(4), 1194–1207.
101. Sarker, I. H. (2019). A machine learning based robust prediction model for real-life mobile phone data. *Internet of Things, 5*, 180–193.
102. Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys, 38*(3), 9–es.
103. Ordonez, C. (2006, November). Comparing association rules and decision trees for disease prediction. In *Proceedings of the International Workshop on Healthcare Information and Knowledge Management* (pp. 17–24).
104. Sarker, I. H. (2019). Research issues in mining user behavioral rules for context-aware intelligent mobile applications. *Iran Journal of Computer Science, 2*(1), 41–51.
105. Sarker, I. H. (2018, March). Behavminer: Mining user behaviors from mobile phone data for personalized services. In *2018 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)* (pp. 452–453). IEEE Computer Society.

106. Phithakkitnukoon, S., & Ratti, C. (2010). A recent-pattern biased dimension-reduction framework for time series data. *Journal of Advances in Information Technology, 1*(4), 168–180.
107. Sarker, I. H., Kabir, M. A., Colman, A., & Han, J. (2017, September). Understanding recency-based behavior model for individual mobile phone users. In *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers* (pp. 916–921).
108. Cheung, D. W., Han, J., Ng, V. T., & Wong, C. Y. (1996, February). Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering* (pp. 106–114). IEEE.
109. Cheung, D. W., Lee, S. D., & Kao, B. (1997). A general incremental technique for maintaining discovered association rules. In *Database systems for advanced applications' 97* (pp. 185–194).
110. Chen, J., & Shi, X. (2002). An incremental updating algorithm for mining association rules. *Computer engineering, 7*.
111. Thomas, S., Bodagala, S., Alsabti, K., & Ranka, S. (1997, August). An efficient algorithm for the incremental updation of association rules in large databases. In *KDD* (pp. 263–266).
112. Zhang, Z., Li, Y., Chen, W., & Min, F. (2014). A three-way decision approach to incremental frequent itemsets mining. *Journal of Information & Computational Science, 11*(10), 3399–3410.
113. Li, Y., Zhang, Z. H., Chen, W. B., & Min, F. (2017). TDUP: An approach to incremental mining of frequent itemsets with three-way-decision pattern updating. *International Journal of Machine Learning and Cybernetics, 8*(2), 441–453.
114. Yao, Y. (2012, August). An outline of a theory of three-way decisions. In *International conference on rough sets and current trends in computing* (pp. 1–17). Berlin, Heidelberg: Springer.
115. Amornchewin, R., & Kreesuradej, W. (2009). Mining dynamic databases using probability-based incremental association rule discovery algorithm. *Journal of UCS, 15*(12), 2409–2428.
116. Thusaranon, P., & Kreesuradej, W. (2015). A probability-based incremental association rule discovery algorithm for record insertion and deletion. *Artificial Life and Robotics, 20*(2), 115–123.
117. Lee, S., Seo, J., & Lee, G. (2010, April). An adaptive speed-call list algorithm and its evaluation with ESM. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2019–2022).
118. Barzaiq, O. O., & Loke, S. W. (2011). Adapting the mobile phone for task efficiency: The case of predicting outgoing calls using frequency and regularity of historical calls. *Personal and Ubiquitous Computing, 15*(8), 857–870.
119. Phithakkitnukoon, S., & Dantu, R. (2008). Adequacy of data for characterizing caller behavior. In *Proceedings of KDD Inter. Workshop on Social Network Mining and Analysis (SNAKDD 2008)*.
120. Bergman, O., Komninos, A., Liarokapis, D., & Clarke, J. (2012). You never call: Demoting unused contacts on mobile phones using DMTR. *Personal and Ubiquitous Computing, 16*(6), 757–766.
121. Stefanis, V., Plessas, A., Komninos, A., & Garofalakis, J. (2014). Frequency and recency context for the management and retrieval of personal information on mobile devices. *Pervasive and Mobile Computing, 15*, 100–112.
122. Sarker, I. H., Colman, A., Kabir, M. A., & Han, J. (2016). Behavior-oriented time segmentation for mining individualized rules of mobile phone users. In *2016 IEEE international conference on data science and advanced analytics (DSAA)* (pp. 488–497). IEEE.
123. El Khaddar, M. A., & Boulmalf, M. (2017). Smartphone: The ultimate IoT and IoE device. *Smartphones from an Applied Research Perspective, 137*.
124. Sarker, I. H., Hoque, M. M., Uddin, M. K., & Alsanoosy, T. (2021). Mobile data science and intelligent apps: Concepts, AI-based modeling and research directions. *Mobile Networks and Applications, 26*(1), 285–303.

125. Pejovic, V., & Musolesi, M. (2014, September). InterruptMe: Designing intelligent prompting mechanisms for pervasive applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 897–908).
126. Peng, M., Zeng, G., Sun, Z., Huang, J., Wang, H., & Tian, G. (2018). Personalized app recommendation based on app permissions. *World Wide Web, 21*(1), 89–104.
127. Zheng, P., & Ni, L. M. (2006). Spotlight: The rise of the smart phone. *IEEE Distributed Systems Online, 7*(3), 3.
128. Google trends (2019). https://trends.google.com/trends/
129. Finin, T., Joshi, A., Kagal, L., Ratsimore, O., Korolev, V., & Chen, H. (2001, September). Information agents for mobile and embedded devices. In *International workshop on cooperative information agents* (pp. 264–286). Berlin, Heidelberg: Springer.
130. de Almeida, D. R., de Souza Baptista, C., da Silva, E. R., Campelo, C. E., de Figueirêdo, H. F., & Lacerda, Y. A. (2006, April). A context-aware system based on service-oriented architecture. In *20th international conference on advanced information networking and applications-volume 1 (AINA'06)* (Vol. 1, pp. 6-pp). IEEE.
131. Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science, 2*(3), 1–21.
132. Shi, Y. (2006, August). Context awareness, the spirit of pervasive computing. In *2006 first international symposium on pervasive computing and applications* (pp. 6–6). IEEE.
133. Anagnostopoulos, C., Tsounis, A., & Hadjiefthymiades, S. (2005, July). Context management in pervasive computing environments. In *ICPS'05. Proceedings. International Conference on Pervasive Services, 2005* (pp. 421–424). IEEE.
134. Zhu, H., Chen, E., Xiong, H., Yu, K., Cao, H., & Tian, J. (2014). Mining mobile user preferences for personalized context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology, 5*(4), 1–27.
135. Sarker, I. H., Colman, A., Kabir, M. A., & Han, J. (2016, September). Phone call log as a context source to modeling individual user behavior. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (pp. 630–634).
136. Eagle, N., & Pentland, A. S. (2006). Reality mining: Sensing complex social systems. *Personal and Ubiquitous Computing, 10*(4), 255–268.
137. Srinivasan, V., Moghaddam, S., Mukherji, A., Rachuri, K. K., Xu, C., & Tapia, E. M. (2014, September). Mobileminer: Mining your frequent patterns on your phone. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 389–400).
138. Mehrotra, A., Hendley, R., & Musolesi, M. (2016, September). PrefMiner: Mining user's preferences for intelligent mobile notification management. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (pp. 1223–1234).
139. Paireekreng, W., Rapeepisarn, K., & Wong, K. W. (2009). Time-based personalised mobile game downloading. In *Transactions on edutainment II* (pp. 59–69). Berlin, Heidelberg: Springer.
140. Cao, H., Bao, T., Yang, Q., Chen, E., & Tian, J. (2010, October). An effective approach for mining mobile user habits. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 1677–1680).
141. Rawassizadeh, R., Tomitsch, M., Wac, K., & Tjoa, A. M. (2013). UbiqLog: A generic mobile phone-based life-log framework. *Personal and Ubiquitous Computing, 17*(4), 621–637.
142. Cao, L. (2017). Data science: A comprehensive overview. *ACM Computing Surveys, 50*(3), 1–42.
143. Haghighi, P. D., Krishnaswamy, S., Zaslavsky, A., Gaber, M. M., Sinha, A., & Gillick, B. (2013). Open mobile miner: A toolkit for building situation-aware data mining applications. *Journal of Organizational Computing and Electronic Commerce, 23*(3), 224–248.