Accepted Manuscript

Mobile crowd location prediction with hybrid features using ensemble learning

Zhongliang Zhao, Mostafa Karimzadeh, Florian Gerber, Torsten Braun

PII: S0167-739X(17)31805-8
DOI: https://doi.org/10.1016/j.future.2018.06.025
Reference: FUTURE 4287

To appear in: *Future Generation Computer Systems*

Received date : 8 August 2017
Revised date : 8 June 2018
Accepted date : 15 June 2018

Please cite this article as: Z. Zhao, M. Karimzadeh, F. Gerber, T. Braun, Mobile crowd location prediction with hybrid features using ensemble learning, *Future Generation Computer Systems* (2018), https://doi.org/10.1016/j.future.2018.06.025

Mobile Crowd Location Prediction with Hybrid Features using Ensemble Learning

Zhongliang Zhao*, Mostafa Karimzadeh, Florian Gerber, Torsten Braun

Institute of Computer Science, University of Bern
Neubrückstrasse 10, 3012 Bern, Switzerland

Abstract

With the explosive growth of location-based service on mobile devices, predicting users' future locations and trajectories is of increasing importance to support proactive information services. In this paper, we model this problem as a supervised learning task and propose to use ensemble learning methods with hybrid features to solve it. We characterize the properties of users' visited locations and movement patterns and then extract feature types (temporal, spatial, and system) to quantify the correlation between locations and features. Finally, we apply ensemble methods to predict users' future locations with extracted features. Moreover, we design an adaptive Markov Chain model to predict users' trajectories between two locations. To evaluate the system performance, we use a real-life dataset from the Nokia Mobile Data Challenge. Experiment results unveil interesting findings: (1) For individual predictors, Bayes Networks outperform all others when data quality is good, while J48 delivers the best results when data quality is bad; (2) Ensemble predictors outperform individual predictors in general under all conditions; and (3) Ensemble predictor performance depends on the user movement patterns.

Keywords: Hybrid feature, Supervised learning, Ensemble learning, Location and trajectory prediction.

*Corresponding author
Email address: zhao@inf.unibe.ch (Zhongliang Zhao)

1. Introduction

Smart-phones are becoming part of people's daily life. Increasing pervasive usage of location-based services and smart-phones around the world contributed to vast and rapid growth of mobility data volume. The large size of heterogeneous mobility data gives rise to new opportunities for discovering characteristics and movement patterns of human mobility behaviors. Mobile data normally consists of historical information of users' visiting sequence, which includes the detailed context of the visited locations and corresponding time-stamps.

Future location prediction is a specific topic in mobile data analysis. The knowledge of mobile user positions fosters applications that need to know this information to operate efficiently. Examples of such services are traffic control, location-based advertising, mobile network management, etc. Many location-based services depend on the current or future locations of users. In addition to location prediction, predicting trajectories between two locations is also of great importance, which helps to optimize travel paths between two locations.

The type of dataset plays an important role in accurate location prediction as the prediction models learn user movement patterns from collected data. The Nokia Mobile Data Challenge (MDC) dataset [1] holds great potential for providing fine-quality information for predicting users' next places. It includes the mobility profiles of nearly 180 users for almost 2 years. From the study of the MDC dataset and the ground truth, we could find out that the visits of certain places follow some regular patterns. Moreover, people behaviors at specific locations also provide useful information for certain predictions.

In this work, we formulate the location prediction problem as a standard supervised machine learning task, where an user-place pair is represented by a set of features and the future places are considered as targets. Our goal is to extract and properly select as many useful features as possible, and build accurate classifiers (both individual and ensemble ones) with those features. We prefer to extract features that have discriminative information among different locations, such that locations can be identified from the observed features. Machine learn-

ing techniques have been widely used to discover behaviors and patterns based on large-scale empirical data. Machine learning algorithms can take advantages of training data to capture characteristics of the unknown probability distribution among different locations. They could automatically learn to recognize complex patterns and make intelligent decisions based on the learned knowledge. In this work, we use WEKA [2], which is a comprehensive open source tool for machine learning and data mining. WEKA provides implementations of multiple machine learning algorithms, and we propose to apply ensemble methods to combine multiple individual predictors to achieve the best performance.

Machine learning can only make accurate classification, if high discriminative features are constructed and useful patterns can be observed from the defined features. However, traditional location prediction methods often separately consider spatial or temporal context information [3] [4]. Although there have been some efforts to integrate spatial and temporal features for location prediction, most of them suffer from over-fitting problems due to the large number of spatial-temporal trajectory patterns. Some existing works model next place prediction as a classification problem [5] [6]. However, issues such as the consideration of other rich contextual data, such as accelerometer, Bluetooth/WiFi connectivity, call/SMS logs, information about running applications have not been investigated systematically. In order to accurately predict the future place of a user, it is fundamental to identify and extract a number of descriptive features for each place visited by the user.

Therefore, this work focuses on extracting discriminative features among different locations, such as temporal, spatial, and smart-phone system features. With these features, we apply ensemble learning techniques to improve the location prediction accuracy. The main contributions of this work are as follows.

- First, we systematically characterize the properties of users' visited places and movement patterns from a real-life dataset and then extract various types of features (temporal, spatial, and smartphone system features) to quantify the correlations between places and features.

- Second, with the extracted features, we propose to apply ensemble learning techniques to improve the crowd location prediction performance by integrating multiple individual predictors. We conducted detailed experiments for users with different movement types and trace qualities to show the superiority of ensemble predictors over individual predictors. Moreover, we also measure the algorithm execution time to show that the superior-performance of ensemble predictors comes at a price of higher computation overheads. This detailed analysis enables us to understand which algorithms could achieve the best performance under what conditions.

- Third, we analyze the performance of different individual and ensemble learning predictors from a mathematical perspective and conduct the time complexity analysis of each algorithm to theoretically understand why there are significant performance differences.

- Fourth, we propose an adaptive Markov Chain-based trajectory prediction approach, which adaptively selects the first-order or the second-order Markov Chain model to predict the future trajectory of mobile users based on dataset conditions.

- Fifth, from the experimental and theoretical analysis, we analyze how the prediction performance is affected by various factors such as mobility trace qualities, extracted features, user movement patterns, predictor models, etc. This knowledge enables us to further design an adaptive prediction system, which dynamically selects predictors based on dataset and smartphone conditions, to guarantee the required system performance.

The structure of this paper is as follows. Section 2 discusses existing efforts on location and trajectory prediction from mobile data. Section 3 describes the dataset that has been used in this work. Section 4 details how we define the features and which features are used in our prediction system. Section 5 explains the individual and ensemble predictors that are used in this study. Section 6 discusses the performance evaluation, and the paper concludes in Section 7.

2. Related Work

With a large number of built-in sensors, smartphones are able to record rich types of quality data without the need of any additional devices. Compared to the check-in data collected from the location-based social networks such as Foursquare [7], which only records the discrete checked-in data at different locations, smartphones have the unique advantage to record data in a continuous way. Therefore, human mobility analysis has become an active research topic thanks to the fast development of continuous location tracking techniques. Song et al. [8] presented a study on predictability of human mobility by analyzing the entropy of location traces. To predict user's mobility, in [9] authors used movement history to map real positions into hexagonal grid. Several prediction methods have been proposed for human mobility in different contexts. Ashbrook et al. [10] introduced to extract significant places and represent location traces as strings and then use Markov models to predict the next place that a user will visit. NextPlace [11] proposed a location prediction solution based on nonlinear time series analysis of the arrival and staying duration of users in relevant places. However, the work is only focusing on GPS coordinates-based prediction. Zhao et al. [12] designed a Dynamic Bayesian Network-based model to predict the future cells of mobile users to optimize telecommunication network operations. He et al. [13] described a time-based Markov predictor for the location prediction of stationary and mobile users. However, their works are limited to specific methods, which can only produce a prediction accuracy of nearly 60%. Moreover, the transition matrix-based approaches have clear drawbacks, since they take only the visit logs as model inputs, but completely ignore the rich context information. In the prediction model proposed in [14], multi-expert combination method is used. However, to reach a satisfactory prediction accuracy the model needs long training time, which is not quite efficient.

In the next place prediction task of Nokia Mobile Data Challenge 2012, the best methods relied only on spatial-temporal information to predict future locations [15], [16], [17], [18]. For instance, Lu et al. [18] focused on using the

transitions between places for each individual user, as well as the time context, to make predictions. They also tried to explore other context information such as call-logs and accelerometer data in the current place. However, they only applied a support vector machine (SVM) for each user to predict their future locations. Tran et al. [19] applied an user-specific decision tree, which was learned from each user's movement history, to predict their future locations. However, their works were limited to the decision tree-based predictor. [20] proposed to learn the time distribution for each place as well as the transition patterns between places by using the kernel density estimation to capture spatial-temporal context features. Zhu et al. [21] introduced a feature engineering mechanism to predict semantic meaning of places. However, their works were also limited to very few individual classifiers. As we can see, most of the existing works focused on applying only individual machine learning algorithms to improve prediction accuracy. However, ensemble learning has been proven to obtain better performance than could be obtained from any of the constituent algorithms alone [22] [23]. Therefore, we focus on applying different ensemble learning methods to optimize location prediction accuracy.

The wide adoption of GPS receivers in smartphones generates huge amounts of personal GPS trajectories. By analyzing those GPS trajectories, we can understand each individual's mobility patterns and obtain valuable insights about his/her daily behavior. These patterns and behaviors can be further utilized to improve the quality of various trajectory-based services, such as route prediction or trip planning [24], and location-based recommendation [25], [26]. In [27], authors applied the optimal stopping theory (OST), which is a traditional mathematic approach, to classify user movement trajectory. OST-based approaches normally require strong assumptions when building models to guarantee optimal solution, which makes it non-practical to solve real mobile crowd location prediction problems. Anagnostopoulos et al. [28] attempted to use machine learning techniques to predict future trajectory of users in road networks. They considered only single predictors and the evaluations were based on a synthetic dataset. Hung et al. [29] proposed a time-related metric to measure the similar-

ity between trajectories. With this similarity metric, they developed a graph-based trajectory prediction algorithm. However, physical roads are different from each other. Some direction changes are sharp, while others are smooth. Thus it is difficult to accurately define similarity between trajectories. In [30], authors proposed a solution considering users' movement patterns among different zones of interest. It is a pure statistical approach, which does not include any future location predictions.

3. MDC Dataset

Our experiment data is from the Nokia Mobile Data Challenge (MDC) [1], a dataset that was collected using Nokia N95 smartphones on a 24/7 basis in Switzerland from October 2009 to March 2011. About 180 volunteers participated in the campaign, where they were asked to carry the smartphones during their daily life with recording software running in the background. Even though volunteers agreed to carry the smartphones during the campaign, their different behaviors lead to different trace qualities. Moreover, users also had different movement patterns, and some users traveled regularly while others did not. Based on these observations, we divided the users into multiple categories, depending on the number of available data points, so called instances, which have been recorded and the movement patterns of the mobile users.

3.1. User Classification

3.1.1. User Trace Quality

Different behaviors of users lead to different trace qualities. Some users carry the smartphones all the time. Therefore, the recorded data is complete and useful for making prediction. However, some others forgot to carry the devices or to charge them in time, such that data recordings are non-continuous and useless for prediction. In the MDC dataset, whenever a user stayed in a place for more than 10 minutes, an entry will be created in the table. The instance includes: `User_ID`, `Place_ID`, `Starting_Time`, `Ending_Time`. `Samp_Dist_Corr`,

which means a user with `User_ID` has arrived at a place (with `Place_ID`) from `Starting_Time` and left the place at `Ending_Time`. Therefore, we define 5 categories of quality, depending on the number of instances recorded in a user's movement traces.

- Very good: $\geq$ 1500instances
- Good: 1200-1500 instances
- OK: 1000-1200 instances
- Bad: 800-1000 instances
- Very bad: $\leq$ 800 instances

3.1.2. User Movement Patterns

In addition to the trace quality, user movement patterns also have significant impact on location prediction. Users had different mobility patterns. Some users moved regularly, they traveled between home and office during working days with a homogeneous movement pattern, and, thus, it is easy to find out patterns. However, some other users traveled randomly and visited many different places for very few times during the data collection period. Their movements are heterogeneous and it is hard to predict their future locations even though the recorded number of data entries is high. Based on this, we defined two types of user movements: homogeneous and heterogeneous. Homogeneous movement means that the user's mobility pattern is quite regular and repeatable, and the user visits some places quite frequently. In contrast, heterogeneous movement means that the movement traces are rather random and non-repeatable. In the experiments we retrieved the visited places of each user, and classify users' movements types based on the number of places a user has visited and the number of the visit. Figure 1 shows an example of homogeneous and heterogeneous movement types, where the user visits very few places frequently homogeneous movement pattern and visited many different places occasionally for heterogeneous movement types.

3.2. Place Category

The raw location data from the MDC dataset were recorded as sequences of GPS coordinates. In our work, we defined places as circular areas that around

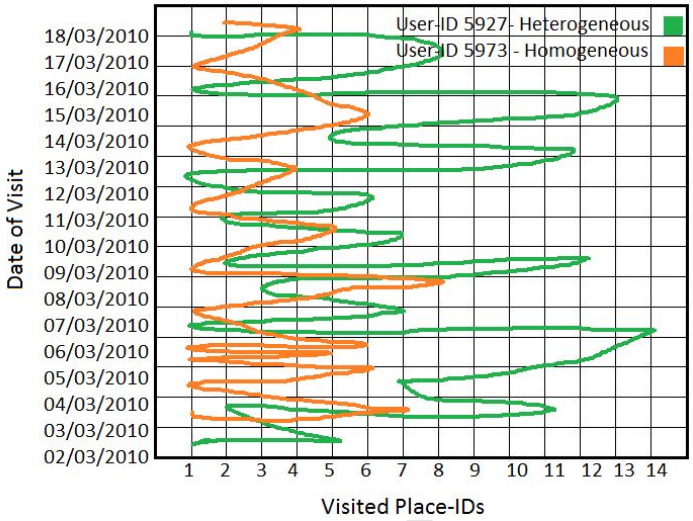

Figure 1: Homogeneous and heterogeneous movements.

GPS coordinate points. As most works on MDC-based location prediction, we defined ten categories of places, which are shown in Table 1.

Table 1: Visited Place Categories

Label	**Place**	**Label**	**Place**
1	Home	6	Outdoor sports
2	Friend home	7	Indoor sports
3	Office	8	Restaurant
4	Transportation	9	Shop
5	Friend office	10	Holiday

3.3. User Trajectory

A mobile user can take different paths to move from one place to another. In Fig.2, the user has two possible trajectories to go from place id 1 to 5, via different other places while being connected to different cells. Thanks to the availability of connected cell ID in the MDC dataset, we are able to extract correlations between users' trajectories with their movement behaviors. We formally define a trajectory t between two places as a sequence t = $\{cell_1, cell_2, ..., cell_n\}$, which contains all the GSM cells that the user connected to while moving from

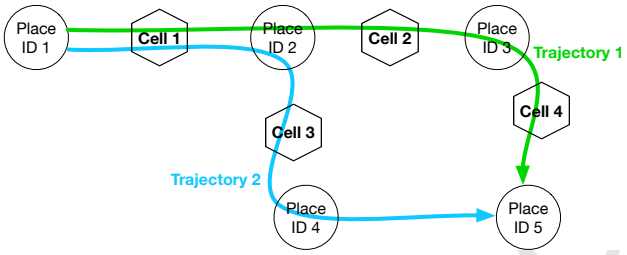

Figure 2: Mobile user trajectories.

one place to another one. Furthermore, we define $T_{i,j} = \{t_1, t_2, ..., t_n\}$ as the set of all trajectories between places i and j.

4. Features

As stated before, a proper feature construction is fundamental to apply supervised machine learning algorithms to make accurate prediction. Therefore, we need to construct features from a tremendous amount of raw data and assign a set of features (feature vector) to each user-place pair. Feature selection is a process of selecting a subset of relevant features (attributes) for their use in prediction model construction. It is the process of choosing a subset of original features such that the feature space is optimally adapted and the appropriate features are selected for classification. The collected MDC raw data is of huge size. Therefore, it is important to select a subset of data by creating feature sets, and identify redundant and irrelevant information. Table 3 shows the association between all the features and places that are used in this work.

4.1. Feature Construction

Most of the MDC-based prediction works use only temporal or spatial features. We combine both and additionally consider the smartphone system-related features, which include context like battery level, charging frequency, detected WiFi network, etc. Below we describe the three categories of features that are used in our system.

4.1.1. Temporal Features

Temporal features include context information relevant to the staying time of a visit. Our visits to certain places tend to have some temporal characteristics that are relevant to the places. For instance, we stay at offices normally between 8:00 to 12:00 and 14:00 to 18:00, and we are at restaurants for lunch between 12:00 to 14:00. Below we detail the extracted temporal features and the feature-place association. We used a time granularity of 1 hour to divide a day of 24 hours. An example of a day time decomposition is shown in Figure 3.

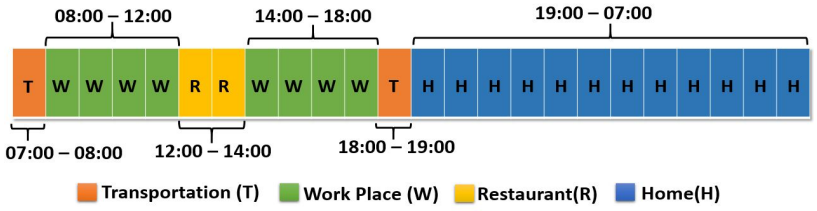

Figure 3: Day time decomposition.

- **Weekday**: to indicate which weekday is the visit.
- **Leaving time**: the ending time of the visit. We defined 6 time intervals, and each time period could be mapped to a specific place. For instance, if the visit is between 07:00 and 08:00, then the place is a transportation hub of a certain probability.
- **Duration**: time duration of the visit at a place.

4.1.2. Spatial Features

Spatial features include context relevant to the geographical information of the visits. We have selected the following feature.

- **Visiting frequency**: how often to re-visit a place.

4.1.3. System Features

Smartphone system features also have discriminative characteristics in different places, and include context information relevant to the smartphone's system

information. We suppose that this information is also helpful when predicting users' future locations. For instance, places like restaurants or homes tend to have more WiFi networks visible than other places, and people tend to have different types of applications running on their phones when they are working in the office or enjoying holidays in a resort.

- **WiFi connection**: the number of visible WiFi networks.
- **Acceleration variation**: movement speed variation, which can be derived from the smartphones' motion sensors. It can be used to detect changes of movement types, for instance a change from slow speed to fast speed probably means the user is at the transportation places.
- **Running application**: the type of running application. This feature is mainly used to detect that whether the users are in indoor or outdoor environments. For instance, map applications are mostly used outdoors, while a connected WiFi network indicates the user is more probably in an indoor environment. These information could further help us to improve the location prediction accuracy.
- **Smartphone profile statement**: profile of the phone, for instance normal or silent mode. Silent mode is more used during office time or concerts, which helps us to predict those places.
- **Charging frequency**: how often the smartphones are charged during the whole period of data collection. People tend to charge their phones in offices and home, which helps us to detect home and office areas.

4.2. Feature Importance

Given the extracted features, the next step is to select those features that influence the prediction output more than others. WEKA has many algorithms to do this automatically, and we choose the *Logistic Regression* algorithm [31]. The *Logistic Regression* algorithm is very efficient for the MDC data set, since it has both nominal and numerical features. Table 2 represents the feature coefficients, which are generated automatically by *Logistic Regression* from WEKA.

Table 2: Feature coefficients

Feature	**Coefficient**
Detected WLAN (1-4)	97.2
Charging frequency (90-100)	85.35
Acceleration variation	32.06
Staying duration (48-120)	30.19
Leaving time (12:30-14:00)	20.5
Frequency of visit (20-60)	29.89
Weekday (Thursday)	21.44
Is_weekend	7.41

It shows that *Detected WLAN* has the best contribution for the prediction result. The *Charging frequency*, *Acceleration variation* and *Duration* of staying at a place are ranked on second level, third level features include the *Visiting frequency* and *Leaving Time* and the *Week day* is the feature with lowest impact on prediction output.

5. Predictors

In this section, we describe the predictors we used to evaluate our prediction system. We focus on the individual predictors as well as on ensemble predictors.

5.1. Individual Predictor

Three categories of individual predictors are mostly used in machine learning: Decision Tree predictors, Bayes predictors, and Neural Networks predictors/Multilayer perceptron.

5.1.1. Decision Tree

A decision tree is a hierarchical structure for classifying objects, composed of nodes that correspond to primitive classification decisions. At the top of the tree is the root node that specifies the first dividing criterion. The root, and every non-leaf node, has multiple child nodes, which can be classified further by checking other criteria. The root node contains all the visits of the training

Table 3: Place-Feature Correlation

Feature / **Place**	Leaving Time	Duration (Minutes)	Weekday	Visit Freq.	# Visible WiFi	Acce. Var. (M/s^2)	Running APP	Phone Profile	Charge Freq.
Home	20:00∼07:00	[480, 2880)	MON to SUN	[300, 450]	[1, 4)	[10, 100)	Indoor	Normal	[250, 300)
Work	08:00∼12:30 13:30∼18:30	[120, 480)	MON to FRI	[200, 300)	[4, 6)	[10, 100)	Indoor	Silent	[90, 250)
Restau.	07:00∼09:00	[40, 120)	MON to SAT	[60, 250)	[6, 12)	–	–	Normal	–
Transp.	07:00∼08:30 18:00∼19:30	[0, 40)	MON to SUN	[20, 100)	[4, 6)	[100,)	–	Normal	–
Outdoor Sports	12:00∼14:00	[0, 60)	SAT to SUN	[15, 70)	–	[50, 100)	Outdoor	Normal	–
Indoor Sports	18:00∼20:00	[0, 60)	SAT to SUN	[15,80)	[1,3)	[50, 100)	–	Normal	–
Shopping Center	–	[40, 120)	FRI to SAT	[30, 130)	[6, 12)	[10, 100)	Outdoor	Normal	–
Holiday Resorts	–	–	–	[5, 30)	–	–	Outdoor	Normal	–
Friend Home	19:00∼22:00	[60, 180)	FRI to SUN	[5, 10)	[1,4)	[10, 100)	–	Normal	[20,90)
Friend Office	–	–	–	–	[4,6)	[10, 100)	–	Normal	–

data, while child nodes contain those visits that match the dividing criteria along the path from root to that node. In our experiments, we used the *J48* and the *Random Forest* algorithms. *J48* is one of the mostly used statistical classifier, and *Random Forest* is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently. Figure 4 shows a J48 tree, in which the first dividing feature is the number of detected WLAN networks, and the features along the path towards the leaf are: duration of a visit in a place, acceleration variation, charging frequency, leaving time from a place, visit frequency of a place, whether the visit is on a weekday or not. The feature ranking is consistent with the feature coefficient shown in Table 2.

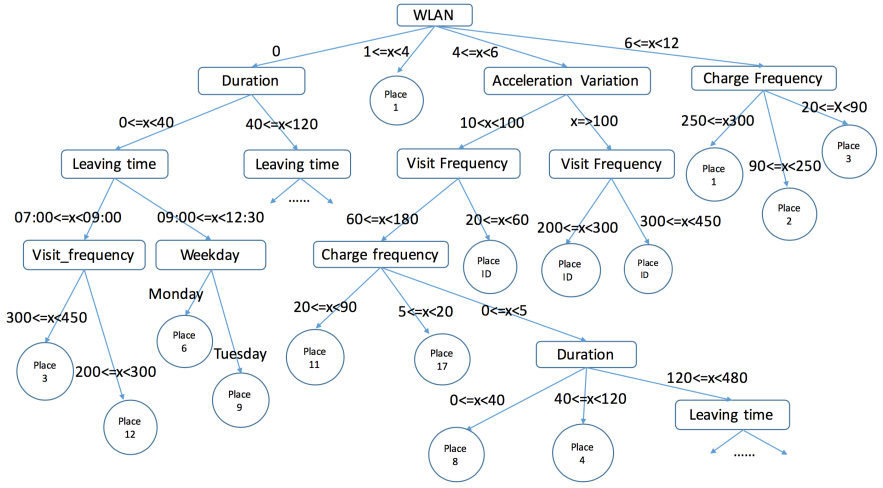

Figure 4: A J48 decision tree.

5.1.2. Bayesian Networks

Bayesian Networks are a class of statistical models to define conditional dependencies between attributes and parent node, represented by a graph. To do so, the *Bayesian Network* uses a *Directed Acyclic Graph* (DAG), to create connections between a set of attributes A=$\{attribute_1, attribute_2, ..., attribute_n\}$ and the parent node. In our case the parent node is visited Place-IDs, because we believed that the current place has a strong connection with the user's next place. Figure 5 shows an example of the *Directed Acyclic Graph* with the extracted features and parent node.

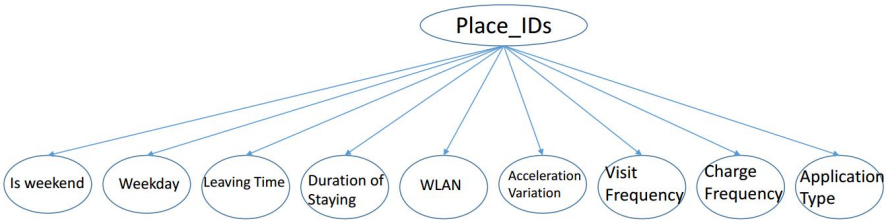

Figure 5: A Directed Acyclic Graph (DAG) of Bayesian networks.

5.1.3. Neural Networks

Artificial Neural Networks are a mathematical model to solve a variety of problems in pattern recognition and classification. ANNs can be viewed as weighted directed graphs in which defined attributes are input layer, classes (Place-IDs) are output layer and directed edges with weights are connections between input and output. In this work, we used the WEKA implementation of ANNs called Multilayer Perceptron (MLP). Figure 6 shows the MLP with extracted features in our case. In this model, connections are organized into layers that have unidirectional connections between them. Weights are determined to allow the network to produce answers as close as possible to the known correct answers. The network usually must learn the connection weight from available training patterns. Performance is improved over time by iteratively updating the weights in the network.

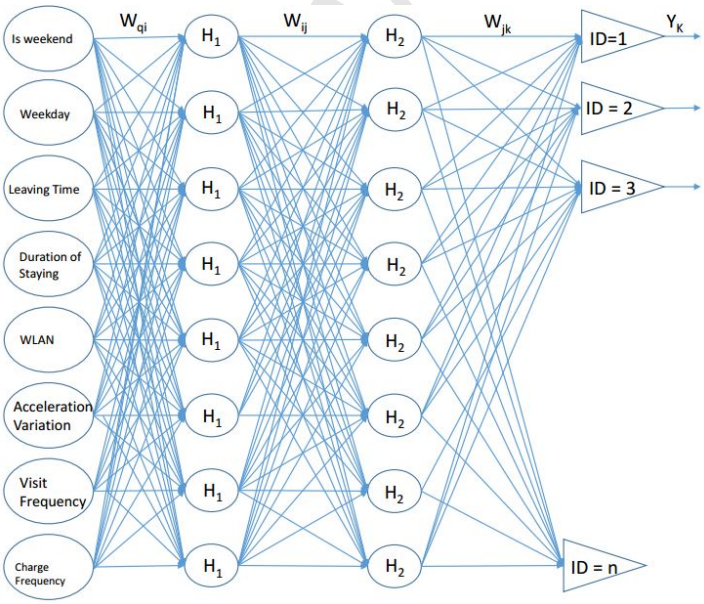

Figure 6: A typical two-layer Multilayer Perceptron Architecture.

5.2. Ensemble Predictors

Ensemble learning is an approach to combine individual predictors to achieve better performance. As different users have different mobility patterns, there is no single predictor that could outperform the others for all users. Therefore, we focus on finding suitable models for different mobility pattern and combine the models to deliver the optimized performance. The task of constructing an ensemble classifier can be broken into two sub-tasks: (1) selecting diverse set of base classifiers with acceptable performance; and (2) appropriate combinations of their predictions with appropriate weights. In this work, three types of ensemble predictors are applied: Boosting, Bagging, and Stacking.

5.2.1. Boosting

Boosting is an ensemble method that begins with a base classifier, which is selected from a first experiment results performed on the training data. A second classifier is then created behind it to focus on the instances in the training data that the first classifier got wrong. The process continues to add classifiers, until an accurate threshold is reached. The *AdaBoost* algorithm was the first practical boosting algorithm that is widely used and studied in numerous applications and research fields [32]. We use it to integrate *J48*, *Random Forest*, *Bayes Networks*, *Naive Bayes* and *MLP*.

5.2.2. Bagging

Bagging is an ensemble method that divides the training data set into several subsets with the same sizes. Then, it creates a classifier for each subset. Afterwards, the final decisions are calculated by getting average values from the results obtained using the individual data sets. In this work, we used *Bagging* to integrate *J48*, *Random Forest*, *Bayes Networks*, *Naive Bayes* and *MLP*.

5.2.3. Stacking

Stacking focuses on a function to combine the outputs of the base learners using a meta-learner, which called *Simple Logistic*. In this work, we integrated*J48*, *Bayes Networks*, and *MLP* with *Stacking*.

6. Performance Evaluation

This section presents the experimentation parameters and detailed performance evaluation of the discussed prediction methods. The evaluation metrics we used are prediction accuracy and prediction execution time, which indicate how accurate the algorithm is and how long it takes to generate the prediction results. From these evaluation results, we further analyze the potential influencing factors on the prediction accuracy performance. We highlight the impacts of temporal and hybrid features, as well as trace quality. Finally, the paper also includes the theoretical analysis about the performance of different algorithms under different conditions.

All experiments were run on a laptop running Windows 8.1 Enterprise with Intel vPro (64-bit-X68 architecture) core i7 CPU 3.2 GHz and 16 GB memory.

6.1. Machine Learning Approaches and Parameters

6.1.1. Location Prediction

In this work we use WEKA [2] to discover the behaviors and mobility patterns of the mobile users by learning from their historical trajectories. WEKA includes several types of machine learning algorithms, such as *Tree-based*, *Bayesian Networks-based* and *Neural Network-based*. Moreover, it also provides ensemble learning methods, such as *Bagging*, *Boosting* and *Stacking*. We study the performance of *J48*, *Random Forest*, *Bayes Networks*, *Naive Bayes* and *Multilayer Perceptron (MLP)* algorithms. In order to improve the accuracy of individual algorithms, we apply *Boosting* and *Bagging* to individual algorithms and apply *Stacking* to integrate multiple individual predictors. We carry out all experiments using temporal+spatial features and hybrid (temporal+spatial+system) features. The experiments are performed using traced data sets of fifteen users, which are randomly selected from different quality categories, and results are averaged over those users. For each user, we divide available trace data into ten subsets using *10-fold cross-validation*, in which one of the 10 subsets is used as the testing set and the other 9 subsets are put together to form a training set. Table 4 shows some of the experiment parameters.

Table 4: Experiments parameters.

Parameter	**Definition**	**Value**
Confidence factor	Reduce the size of the decision tree by removing insignificant nodes	0.25
Number of objects	Minimum number of instances per leaf in the decision tree	2
Hidden layers	Hidden layers of the neural network	45-55
Validation	Number of iterations to run after observing lower prediction accuracy in *Boosting*	2
Maximum depth	Maximum depth of a tree in *J48* and *Random Forest*	1000 *level*
Training time	Duration of training for individual algorithms per iteration in *Boosting*	300 *sec*
h	Number of neurons at each hidden layer *Stacking*	3
o	Number of outputs in MLP *Stacking*	100-160
i	Number of iterations in MLP *Stacking*	5
T	Number of trees to generate in *Random Forest*	20
L	Number of possible iterations for individual algorithms in *Boosting*	5
N	Number of new generated training sets in *Bagging*	10
J	Number of new generated training sets in *Stacking*	10

6.1.2. Trajectory Prediction

For trajectory prediction, we have developed a novel adaptive Markov Chain-based model. As defined in Section 3.3, $T_{i,j}$ is a set of trajectories $T_{i,j} = \{t_1, t_2, t_3, ..., t_n\}$, where each trajectory t_n is a set of m connected cells such that $t_n \in T_{i,j} : \{cell_1, cell_2, cell_3, ..., cell_m\}$. For each subset t_n the first cell is located in *Place-ID* $= i$ and the last cell is located in *Place-ID* $= j$. In addition, connected cells on trajectories between two places i and j do not appear on other trajectories starting from place i towards other places. As shown in Fig. 7, the model compares the detected periodicity (P) with a predefined threshold value (P_{th}) to decide either the first order or the second order Markov Chain model should be applied. The First Order Markov Chain is applied if the user's mobility pattern between two places is regular (homogeneous movements). For place pairs where the user's mobility pattern is irregular (heterogeneous movements), the Second Order Markov Chain is used. We use the periodicity detection approach proposed in [33] to identify the user movement types and detect the

changes of user movement patterns such that the corresponding Markov Chain model is applied. With this model, the probability of the next cell in a trajectory is given by:

$$Pr(cell_{i+1}) = \begin{cases} Pr(cell_{i+1} \mid cell_i) & \text{if } P \leq P_{th} \\ Pr(cell_{i+1} \mid cell_i, cell_{i-1}) & \text{if } P > P_{th} \end{cases}$$

In the experiments, for a given $T_{i,j} = \{t_1, t_2, ..., t_n\}$ we use the threshold value $P_{th} = \sum_{i=1}^{n} \frac{|t_i|}{n}$, which denotes the mean length of a trajectory in $T_{i,j}$.

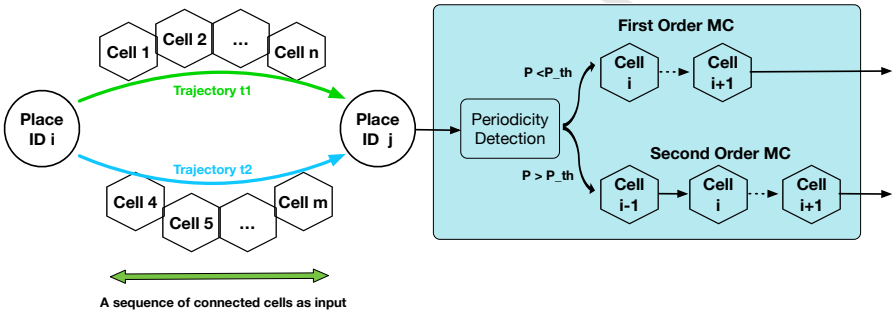

Figure 7: Adaptive Markov Chain-based Trajectory Prediction.

6.2. Evaluation Results

In this subsection, we present the evaluation results of different predictors. We focus on two metrics: prediction accuracy and prediction time. Location prediction accuracy refers to the percentages of correct location prediction, and prediction time refers to the execution time of performing the prediction task. For trajectory prediction accuracy, we use the metrics of precision and recall, as defined in the work of [30].

6.2.1. Location Prediction Accuracy of Individual Algorithms

This subsection details the prediction accuracy results of individual algorithms. We first present the average prediction accuracy of all the users with different trace qualities. Then, we discuss more details about the prediction accuracy of users with homogeneous and heterogeneous movement patterns.

Fig. 8 and Fig. 9 show the average prediction accuracy of all the users for different individual algorithms using temporal, spatial, and hybrid features. The results clearly show that the *Decision Trees* family (specially *J48*) outperform others, when using the trace data with lower quality, and *Bayes Networks* provides better performance ($> 84\%$ accuracy) when the data is with higher quality. Moreover, it can be observed that the estimated accuracy is improved significantly if the hybrid features are used instead of using only temporal+spatial features. For instance, *Bayes Networks* delivers an accuracy of 84.76% with hybrid features, while only 55.47% can be reached with temporal+spatial features.

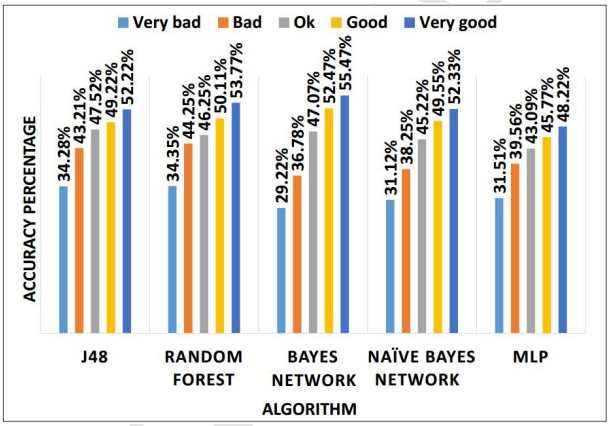

Figure 8: Prediction accuracy of individual algorithms using Temporal+Spatial features.

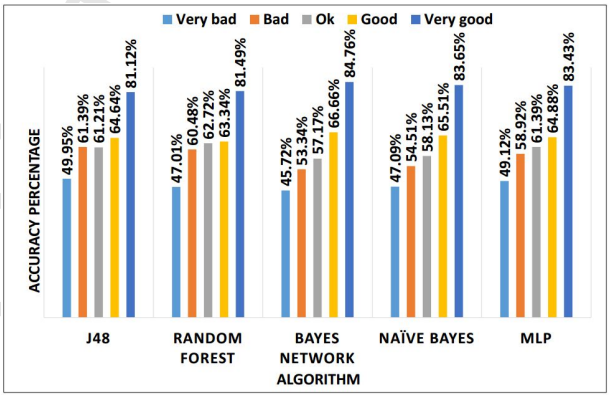

Figure 9: Prediction accuracy of individual algorithms using Hybrid features.

Fig. 10 shows two confusion matrices that help to explain the nature of the errors made by the classifier with different features[34]. A confusion matrix is a table that is often used to describe the performance of a classifier on a set of test data for which the true values are known. For instance, row 1 of the table shows that 78 places with real class type = 1 were wrongly predicted as class 2, and 171 places with real class type = 1 were correctly predicted. These matrices are generated by the *J48* algorithm over the 10 most visited places (indicated by IDs). For instance, Fig. 10a shows that when the predictor uses only the temporal+spatial features, prediction accuracy is lower and several incorrect predictions are observed. Fig. 10b shows that the number of correct predictions are significantly improved when the hybrid features are used.

↓ Classified as ↓	1	2	3	4	5	6	7	8	9	10
1	171	78	20	24	34	0	3	1	0	5
2	27	130	18	4	19	0	0	0	0	1
3	19	55	49	4	12	0	0	0	0	0
4	30	25	5	52	63	0	1	0	0	0
5	65	36	14	39	133	0	4	0	0	0
6	0	1	0	0	6	0	0	0	0	0
7	5	5	1	0	9	0	5	0	0	0
8	14	1	2	5	21	0	0	0	0	0
9	3	0	0	1	2	0	0	0	0	0
10	8	29	22	4	10	0	0	0	0	3

(a) *Temporal.*

↓ Classified as ↓	1	2	3	4	5	6	7	8	9	10
1	354	0	0	0	0	0	0	0	0	0
2	0	202	0	0	0	0	0	0	0	0
3	0	0	146	0	0	0	0	0	0	0
4	0	0	0	178	0	0	0	0	0	0
5	0	0	0	0	298	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	27	0	0	0
8	0	0	0	0	0	0	0	42	0	0
9	0	0	0	0	0	1	0	0	4	0
10	0	0	0	0	0	0	0	0	0	76

(b) *Hybrid.*

Figure 10: Confusion matrices using different features.

Next, we present the prediction accuracies of individual predictors for users with homogeneous and heterogeneous movement patterns. As shown in Fig. 11, the *Bayes Networks* scheme delivers the best performance for both movement patterns, which is consistent with its superior performance presented in Fig. 9.

6.2.2. Location Prediction Accuracy of Ensemble Methods

In this subsection, we present the prediction accuracy of different ensemble learning algorithms. Same as for the individual algorithms, we first present the average prediction accuracy of ensemble learning algorithms for all users. Then,

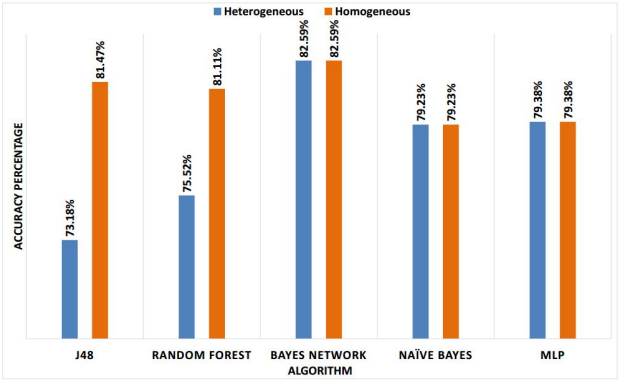

Figure 11: Prediction accuracy of individual predictors with hybrid features for homogeneous and heterogeneous movements

we discuss more details about the prediction accuracy of users with homogeneous and heterogeneous movement patterns.

Fig. 12 and Fig. 13 present the prediction results of *Boosting* and *Bagging* using hybrid features. The graphs show that using *Boosting*, prediction accuracy is improved by around 10% compared to when individual algorithms are applied. It can also be observed that *Boosting* outperforms *Bagging*. Different algorithms provide different prediction performance values. For instance, *J48* using *Boosting* performs better when the traced data is of low quality. However, using traced data with higher quality, the integration of the *Bayes Networks* and *Boosting* outperforms the others.

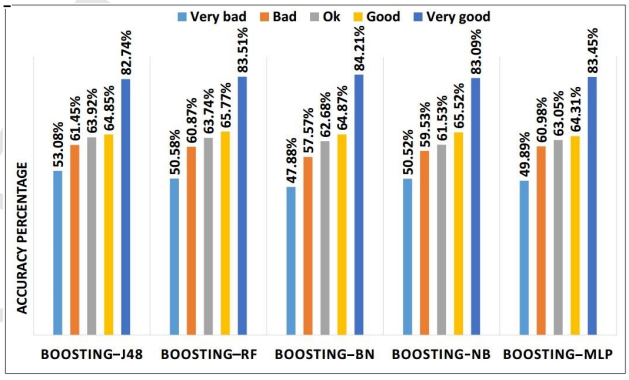

Figure 12: Prediction accuracy of *Boosting*.

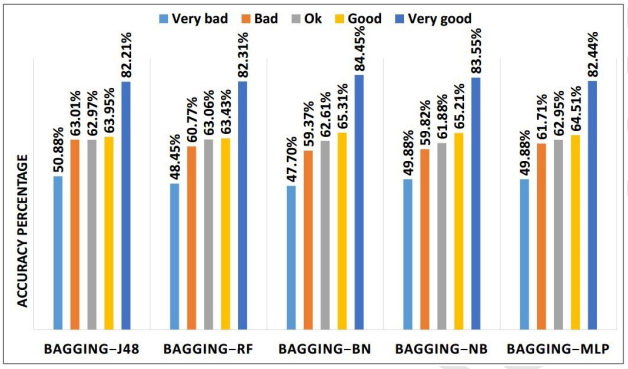

Figure 13: Prediction accuracy of *Bagging*.

Fig. 14 shows the evaluation results of the *Stacking* learning method built by *Simple logistics* as a meta-learner for the hybrid features. Due to generating higher accuracy results by *J48*, *Bayes Networks*, and *MLP*, we decided to integrate them using *Stacking*. *Random Forest* and *Naive Bayes* are ignored as they do not improve prediction accuracy. The graph shows that by integrating *J48* and *MLP*, prediction performance is improved by 10% to 14% compared to the individual algorithms even for trace data with low quality. Another significant improvement can be observed when *J48* is integrated with *Bayes Networks* and *MLP* mechanisms, particularly when the trace data is of high quality.

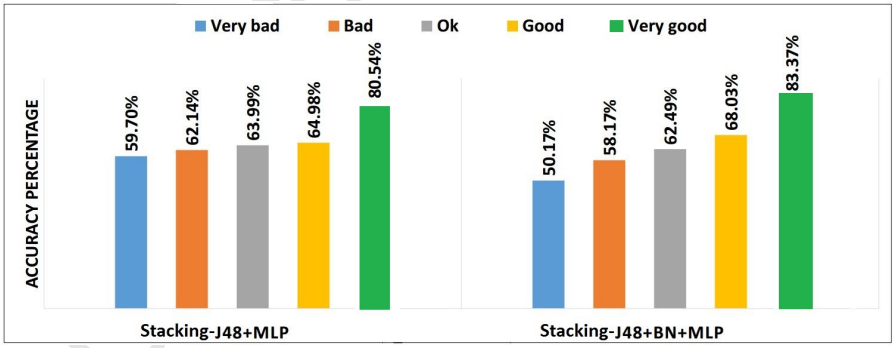

Figure 14: Prediction accuracy of *Stacking*

Next, we discuss the prediction accuracies of ensemble predictors for users with different movement patterns. We take user 5927 as an example, and as shown in Fig. 15, *Boosting* delivers better results than *Bagging* for both movement patterns, which is also consistent with the results presented in Fig. 12 - 13. Therefore, from Fig. 11 and Fig. 15 we see that *Boosting* significantly outperforms individual predictors for homogeneous movements, while for heterogeneous movements, their performance are similar to each others. Therefore, an adaptive model selection mechanism should be developed based on the detected movement patterns such that the appropriate predictors can be applied to guarantee optimal prediction performance.

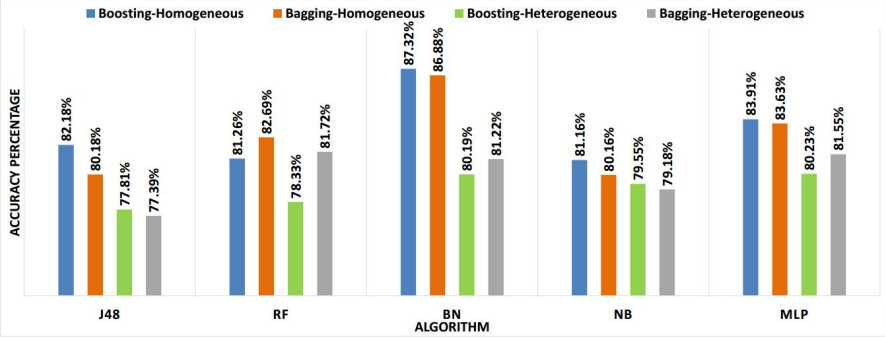

Figure 15: Prediction accuracy of *Boosting* and *Bagging* with hybrid features for homogeneous and heterogeneous movements

6.2.3. Location Prediction Execution Time of Individual Algorithms

In addition to prediction accuracy, we also measure the prediction execution time of each individual algorithm using temporal+spatial features and hybrid features. The obtained results, as shown in Fig. 16 and Fig.17, indicate that the *Decision Tree* and *Bayes* families could generate the prediction faster. *MLP* is the one requiring more execution time compared to the others.

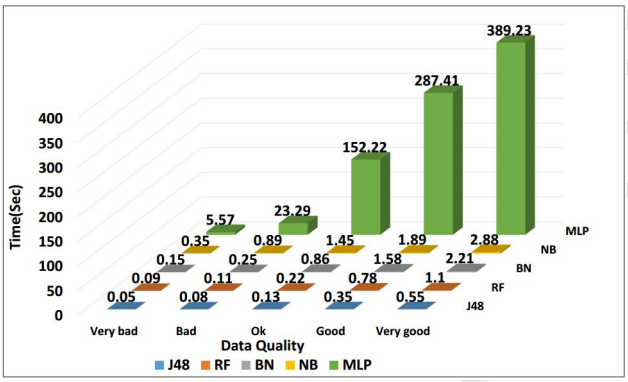

Figure 16: *Average execution time of individual algorithms using Temporal+Spatial features.*

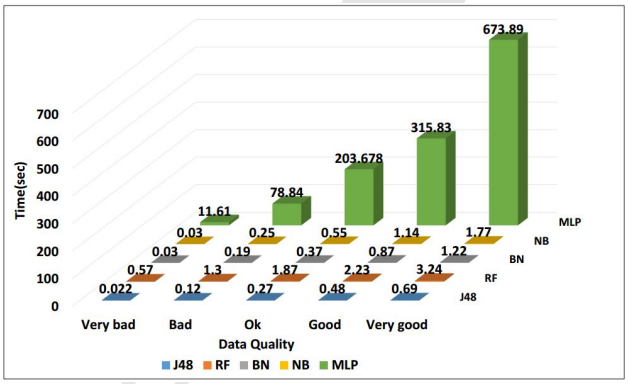

Figure 17: *Average execution time of individual algorithms using Hybrid features.*

6.2.4. Location Prediction Execution Time of Ensemble Methods

Fig. 18 - 23 present the average execution time of *Boosting*, *Bagging* and *Stacking* learning methods, using temporal+spatial and hybrid features. The results show that *Boosting* outperforms *Bagging* for different algorithms. When *J48* and *MLP* are combined using *Stacking*, the execution time is 12'012 seconds for very good quality traces and 109 seconds for very bad quality traces. When *J48*, *Bayes Networks* and *MLP* are combined with *Stacking*, the execution time is 15'078 seconds for very good quality traces and 187 seconds for very bad quality traces.

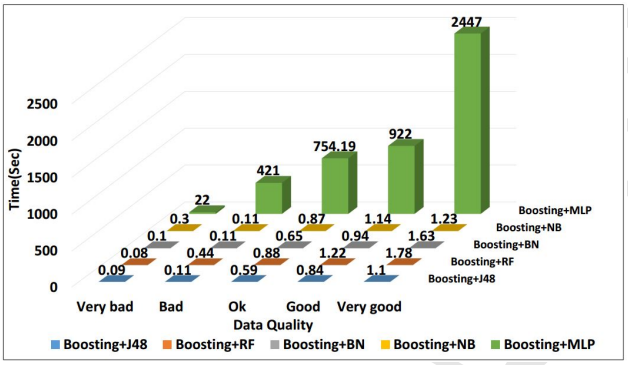

Figure 18: Average execution time of *Boosting with Temporal+Spatial features.*

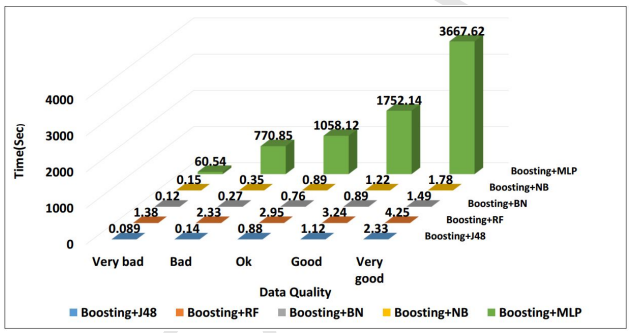

Figure 19: Average execution time of *Boosting with Hybrid features.*

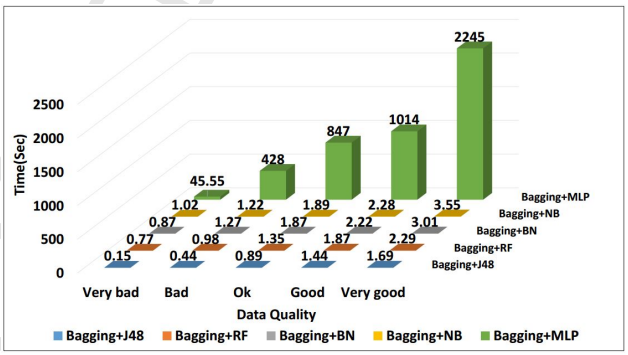

Figure 20: Average execution time of *Bagging with Temporal+Spatial features.*

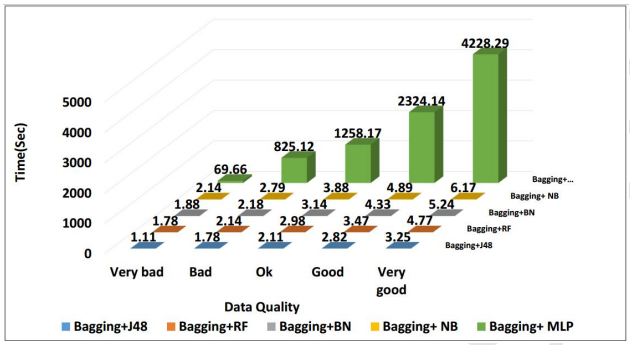

Figure 21: Average execution time of *Bagging with Hybrid features.*

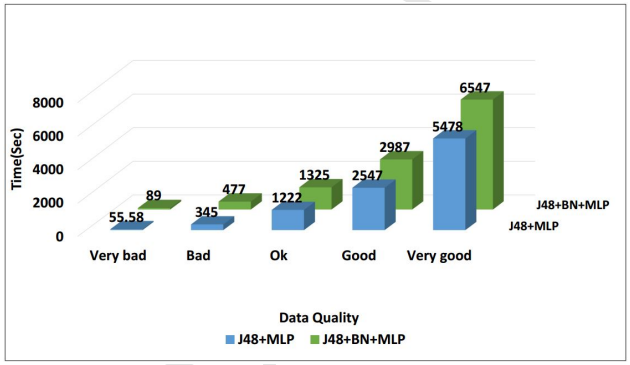

Figure 22: Average execution time of *Stacking with Temporal+Spatial features.*

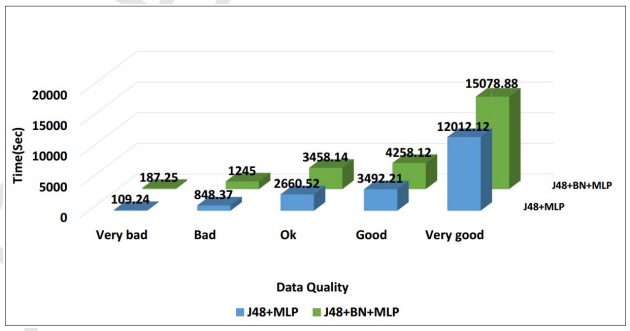

Figure 23: Average execution time of *Stacking with Hybrid features.*

6.2.5. Trajectory Prediction Accuracy

This subsection discusses the results of our trajectory prediction algorithm. We take user 5927 as an example, whose mobility trace includes both homogeneous and heterogeneous movement patterns. Fig. 24 - 25 show the predicted trajectories for one transition with connected cells between location ID 2 to 3 and 4 to 5 for user 5927, in which black dots are GPS coordinates, red circles indicate the frequently visited places, and the yellow circles are the sequence of cells that the user will be connected between the places. Since the exact coverage areas of the GSM cells are not known, we estimated their position by calculating the mean position of the user within a time window of one minute when a GSM entry was registered. As shown in Fig. 26, our proposed adaptive Markov Chain model could achieve a trajectory prediction accuracy of nearly 80% for homogeneous movements and 70% for heterogeneous movements for user 5927.

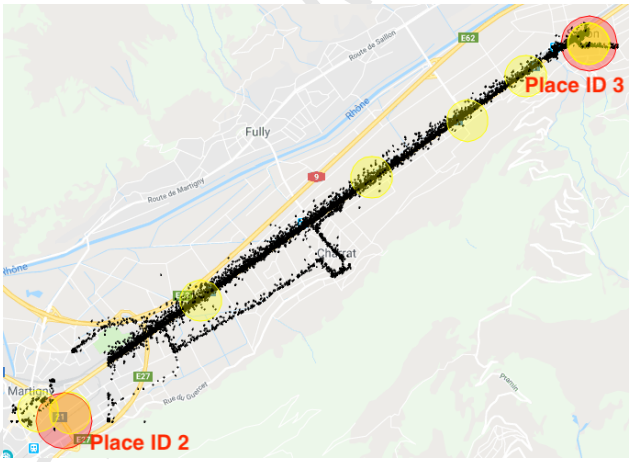

Figure 24: Trajectory prediction of user 5927 between location ID 2 and 3.

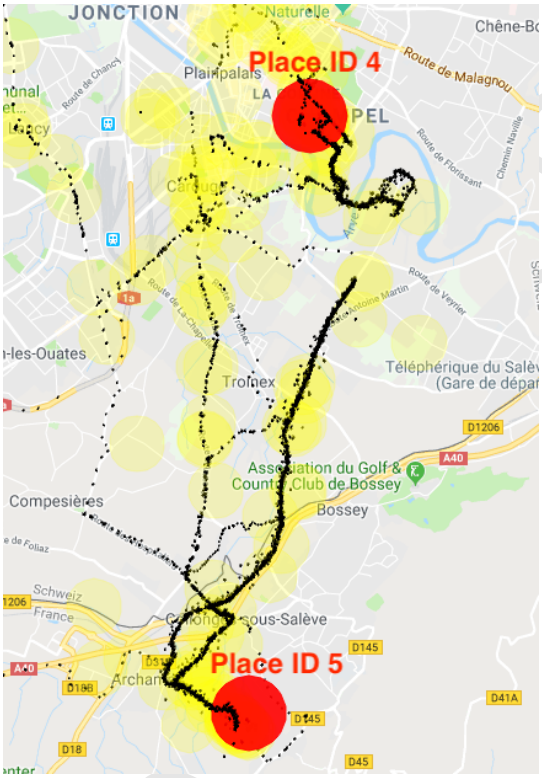

Figure 25: Trajectory prediction of user 5927 between location ID 4 and 5.

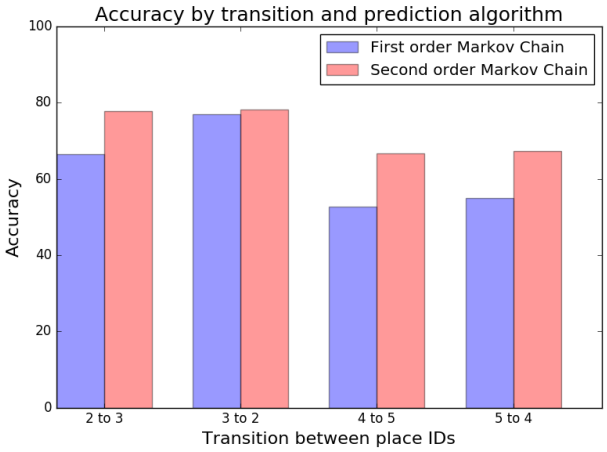

Figure 26: Trajectory prediction accuracy of user 5927 between place IDs

6.3. Location and Trajectory Prediction Accuracy Comparison with Past Studies

In this section, we present the location and trajectory prediction performance comparison with past correlated studies to show the superiority of our solutions. For location prediction, we take relevant location prediction accuracy results from [18] [6], which are the winners of the mobility prediction task in the Nokia Mobile Data Challenge. The results are shown in Table 5.

Table 5: Accuracy comparison of location prediction approaches.

Work	**Algorithms**	**Features**	**Best Accuracy (%)**
Our work	Stacking	Hybrid features	83.37
HKUST [18]	Gradient Boosting Trees	Limited hybrid features	76.32
EPFL [6]	Blending	Temporal features	56.22

As we can see from Table 5, our solutions significantly outperform the others. This is because in [6], authors applied the *Blending* technique, which is an ensemble learning approach similar to *Stacking*, to deliver the best accuracy using only temporal features. They considered information such as starting/ending time of a visit, the visit is on weekday or weekend. In [18], authors explored temporal and smartphone system features with the *Gradient Boosting Trees* approach. However, they did not consider a wide range of features as we did. For instance, they only used the mean and variance of visit duration at a place for the temporal features. Therefore, by applying ensemble learning using a wide range of hybrid features, our solutions provide the best performance.

For trajectory prediction, we compare our work to the trajectory estimation using the adaptive, mean and F-score optimization threshold [30]. All methods were implemented using the GSM cell representation of the trajectories as input data. We use the performance metrics of precision and recall, as defined in the work of [30]. The results are shown in Table 6.

Table 6: Precision and recall comparison of trajectory prediction approaches.

Work	**Method**	**Precision (%)**	**Recall (%)**
Our work	Markov Chain	81.92	68.00
Chapuis et al. [30]	Mean threshold	60.30	92.81
	F_1 optimization threshold	70.42	89.51
	Adaptive threshold	70.42	89.51

As shown in the Table 6, our methods outperform all trajectory estimation methods proposed in work of Chapuis et al. [30] significantly in terms of precision but it is outperformed in terms of recall. The lower recall number can be explained by looking at how the proposed adaptive Markov Chain predicts full trajectories. When dealing with predicting the full trajectory starting from one place, the adaptive Markov Chain sequentially adds the next most probable cell to the predicted trajectory. Considering the case that starting from some cells the transition probabilities to two different cells is high, one of them will be left out from the prediction, since only the cell with the highest transition probability is added. Therefore, for evaluating the performance of the adaptive Markov Chain-based trajectory prediction mechanism, it is better to use prediction accuracy as a metric. This is because as opposed to other methods [30], the cells in the trajectory are predicted in order. Using the Markov Chain as a predictor also has the advantage that after the mis-prediction of a cell, a new trajectory starting from the actual cell can be generated.

6.4. Algorithm Complexity Analysis

In this subsection, we present computational complexity of individual and ensemble algorithms. In machine learning, model complexity often depends on the number of extracted features and samples in the training set. *Decision trees* are the fastest known algorithms, the run time cost to construct a *decision tree* is $O(n_{samples} m_{features} log(n_{samples}))$. In general, the *Bayes Networks* are powerful algorithms and efficient in terms of execution time. Their run time is $O(2^{m_{features}-2}(m_{samples} n_{features}))$[35]. $n_{samples}$, $m_{features}$ represent number of records in training set and number of features, respectively. *MLP* has a high

Table 7: Time complexity comparison

Learning algorithm	**Complexity**
Decision tree (DT)	O(28,800)
Bayes network (BN)	O(614,400)
MLP	O(500×10^7)
Boosting + DT	O(144×10^3)
Bagging + DT	O(288×10^3)
Stacking + (DT + MLP + BN)	O(501×10^8)

time complexity. Suppose that there are $n_{samples}$ training samples, $m_{features}$ features, k hidden layers, each containing h neurons and o output neurons. The time complexity of *MLP* is $O(n \times m \times k^h \times o \times i)$, where i is the number of iterations. Since *MLP* has a high execution time, it is advisable to start with a smaller number of hidden layers for training [36]. In ensemble learning, execution time of *meta-learners* is negligible and they have not much impact on running time of base classifiers. Running time of *Boosting* is $O(L \times f)$, where f is the runtime of the base classifier and T is number of iterations. Time complexity for *Bagging* is $O(N \times f)$, where N is number of new generated training sets and f is run time of individual algorithm [37]. *Stacking* applies several individual learner to training data and then combines output of them using a meta-learner. The overall complexity of stacking is $O(f_1 + f_2 + f_3, ..., f_n)$ n=1,...,N, where f_n denotes time complexity of each individual learner. Table 7 presents time complexity comparison for individual and ensemble learning algorithms. For this experiment we choose user 5925 with 1200 records in the training set and 8 extracted features. As we can observe, the time complexity follows the same ordering of execution time as shown in Fig.17, 19, 21, and 23.

6.5. Theoretical Analysis

In this section, we analyze the performance of different predictors from a mathematical perspective. We aim to find out the impacting factors of difference predictors, and understand theoretically why they have different performance under different conditions.

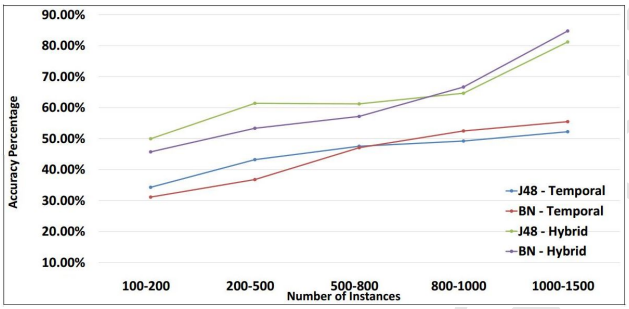

Figure 27: Prediction accuracy of *J48* and *Bayes Networks*

6.5.1. Analysis of Individual Algorithms Performance

Section 6.2.1 presents the prediction accuracies of individual predictors. As we can see from the results, Decision Tree-based approaches (especially the *J48* algorithm) outperform others when the trace data is of lower quality, while the *Bayes Networks* scheme provides better performance ($> 84\%$ accuracy) for trace data with higher quality. To better understand these behaviors, we highlight the performance comparison of *J48* and *Bayes Networks* by decomposing the mathematical components of each model to explain why different predictors have different performance. Fig. 27 shows the average prediction accuracy of the *J48* and *Bayes Networks* algorithms using temporal or hybrid features as a function of trace qualities, which are summarized in Fig. 8. It is interesting to observe that for both cases, *J48* outperforms *Bayes Networks* when the quality of traced data is low (e.g., with 100-500 instances). This is due to the fact that the algorithms relying on the decision tree use the *surrogate splits* approach, which is a method to estimate missing data, to overcome the deficit of missing data on the trace files [38]. However, *Bayes Networks* do not have a future action in presence of a trace file with a lot of missing data, and its prediction is based only on available data.

When making a prediction, J48 estimates the missing instances based on the present ones, resulting in higher accuracy of the prediction outcomes. The missing instances can be either numerical attributes (e.g., leaving time, duration

of staying in each place, place id, etc), or nominal attributes (e.g., application type, etc), whose values could be missing randomly. The missing attribute parameters with nominal value can be estimated based on available instances with the same attribute. Assuming that the day of visiting a particular place (e.g., Place-ID = 1) for a user is missing, the surrogate split approach [39] can estimate the missing value (e.g., day of a visit), knowing that (using users previous trajectories) on which day the user often visits the location with the same Place-ID. Our problem can be modelled by Eq. 1.

$$\widehat{V}_{i,j} \cong argmax_{v_{i,j} \in (a_i)} | \sigma_{a_i} = v_{i,j} \quad and \quad y = y_p^i \quad D| \tag{1}$$

$\widehat{V}_{i,j}$ defines the estimated parameter, $v_{i,j}$ represents the missing parameter of attribute a_i with index j, σ_{a_i} includes the subset of missing parameters for attribute a_i, y_p^i shows the value of the target attribute (e.g., *duration_time*, *application_type*) and D is the provided data set. If the missing parameter of attribute a_i has a numerical value, the estimation is performed by calculating the *mean* (*average*) of the existing data instances with the same attribute. The outcome of the estimation of the *decision tree-based* algorithms is more similar to the original data if there is no continuously missing data on the trace files. As shown in Fig. 27, the *J48* and *Bayes Networks* algorithms generate similar results if the trace data is of low quality (e.g., with 100-200 instances). *J48* performs better for improved quality of trace data (e.g., with 200-500 instances). However, it is interesting to observe that *Bayes Networks* overtake *J48*, if the quality is better (e.g., with 700-1500 instances). This is due to the fact that *Bayes Networks* follows a graphical model, making possible relations between the parameters with particular probabilities [40]. When the number of existing instances raises, the generated graph used in the model requires more computation overhead, but resulting in more accurate prediction. The graph is integrated with a set of local probability distributions to define the joint probability distribution [41]. The joint probability distribution is defined in Eq. 2.

$$Pr(X|m,\theta) = \Pi_{i=1}^{n} Pr(X^i|\Pi(X^i),\theta) \quad (2)$$

X^i, denotes attributes in DAG, $\Pi(X^i)$ shows the set of parents (e.g., Place-ID = 1, Place-ID = 2,...), θ is a vector of conditional probabilities, m represents the DAG model and local probability distributions are the distributions corresponding to the terms in the product of Eq. 2.

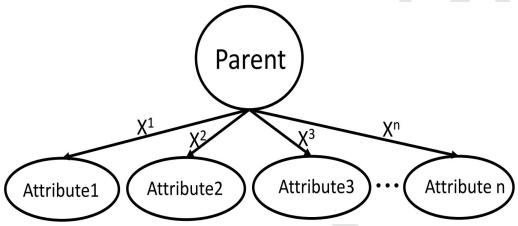

Figure 28: Directed Acyclic Graph (DAG) of Bayes Networks.

600 *6.5.2. Analysis of Ensemble Learning Algorithm Performance*

As presented in Fig. 13 and 14, the experiment results show that the integration of the individual algorithms (e.g., *J48*, *Bayes Networks* and *MLP*) using *ensemble learning* methods can efficiently improve prediction accuracy. This is because for machine learning algorithms, the bias error and variance error, as explained in Eq. 3, are the main components of the prediction errors. However, all ensemble learning methods are able to mitigate these errors such that the prediction performance could be enhanced. The bias error defines the difference between values of the expected prediction (*average of estimated predictions*) and the real one. The variance error determines the variability of the prediction accuracy due to small modifications in the training set.

$$\begin{aligned} Err(X) &= bias\ error^2 + variance\ error + noise\ error \\ &= (E[g(x)] - f(x))^2 + E[(g(x) - E[g(x)])^2] + \epsilon_e^2 \end{aligned} \quad (3)$$

$f(x)$, $g(x)$, $E[g(x)]$ and ϵ_e^2 denote the correct value to predict ($Place\_ID$), estimated prediction calculated by the algorithm, expected prediction, and noise

error, respectively. Ensemble predictors can be applied to enhance the prediction performance of individual algorithms by mitigating the variance error.

Even though ensemble learning could deliver better prediction accuracy than individual algorithms, they also perform differently according to how they address the variance error. *Bagging* does this by creating N new subsets of training data with the same size, as shown in Table 4. The new data sets are generated from the original data, randomly sampled and replaced [42]. Therefore, the total variance (Z) will be decreased as it is divided among the newly generated training data sets. Variance of each new subset can be calculated using Eq. 4.

$$Variance_j = \frac{1}{N} Var(Z) \quad j = 1, ..., N \tag{4}$$

For *Bagging*, the training phase is performed independently over all the new data sets. Later, as shown in Eq. 5, the final prediction accuracy ($Pr_{Bagging}$) is obtained by getting a *simple-averaging* over the outcomes computed in each new data set (e_j). This implies that there is no mechanism in *Bagging* to specify whether the parameters are classified correctly or not. This means that all the parameters appear with the same probability in newly generated data sets [43].

$$Pr_{Bagging} = \frac{1}{N} \Sigma_{j=1}^{N} e_j \quad j = 1, ..., N \tag{5}$$

Boosting applies a sequential model in the learning phases [44]. After each iteration, the weights of parameters are determined based on the current prediction error, as shown in Eq. 6. Next, the weights are assigned to uncorrected classified parameters. Therefore, the wrongly-classified parameters will appear in the new training set with bigger weights than the correctly classified ones. This repetition decreases the diversity of the parameters in the training sets, which results in a reduction of the variance and consequently a better prediction performance. The parameters used in this equation 6 are listed in Table 8.

$$w_t^{h+1} = \frac{w_t^h \beta_h^{(1-l_h^t)}}{\Sigma_{i=1}^{N} w_i^h \beta_h^{(1-l_h^t)}}, \quad w_t^1 \in [0, 1], \quad \Sigma_{t=1}^{N} w_t^1 = 1 \tag{6}$$

Table 8: The notations and definition of parameters

Parameter Name	Parameter Definition
$w_t^1 = [1, ..., w_N]$	Set of possible weights for the first step of iterations, usually $w_t^1 = \frac{1}{N}$
$h = 1, ..., L$	Number of iterations in *Boosting*
$l_t^h = 0, 1$	Prediction in iteration h is incorrect (=0) / correct (=1)
$\beta_h^{(1-l_h^t)}$	Current prediction error of algorithm in iteration h
w_t^h	Current weight at iteration h
w_t^{h+1}	Calculated weight for iteration h+1

For *Stacking*, different kinds of individual algorithms can be integrated to improve performance. *Stacking* achieves this through two steps. Firstly, the given data set of $D = \{(y_n, x_n), n = 1, ..., N\}$ is randomly split into J smaller data sets (parameters defined in Table 4). The generated sets have almost equal sizes, denoted by the $d_1, ..., d_J$. Thereafter, the individual algorithms (*level-0 algorithms*) carry out prediction on the generated data sets independently [45]. The outcomes of each prediction algorithm (e.g., visited place in our scenario) can be defined using Eq. 7:

$$z_{kn} = \{(P_k^{(d_1)}(x_n), ..., P_k^{(d_j)}(x_n)), k = 1, ..., K, \quad n = 1, ..., N\} \tag{7}$$

$P_k^{(d_j)}(x_n)$ denotes the prediction of individual algorithms for each instance x in the newly generated data sets (d_j). Later, a new data set is created using the IDs of the visited places (y_n) and the output of the K individual algorithms (z_{kn}). Formally, the new data set is represented as:

$$L_{Level-1} = \{(y_n, z_{1,n}, ..., z_{k,n}), n = 1, ..., N\} \tag{8}$$

$L_{Level-1}$ defines the input data for the second step, including the predicted values for each visited place. This input is different from the one for the first step. The input of the first step includes the Place-ID and extracted features from the trace data. Next, the *meta-learner* (*Level-1 algorithm*) uses the *Weighted Majority* method [46][47] to further improve prediction accuracy. *Weighted Majority* is an approach to decide weights of each algorithm based on their individual prediction performances.

Based on the aforementioned description, we could imagine that a particular algorithm could only provide a low prediction accuracy, due to the high variance of the data set used in the learning phase. Afterwards, the *Weighted Majority* method can be applied to enhance the accuracy of the final prediction by getting benefits of other algorithms, which provides more accurate results.

7. Conclusions

In this paper, we model the future place prediction problem as a standard supervised learning task and ensemble learning methods with hybrid types of features. Our approach characterizes the properties of users' movement patterns and visited places, then extracts rich types of features (temporal, spatial, and system features) to quantify the correlation between places and features. Finally, we propose to use ensemble learning approaches to predict users' future locations. Additionally, we also propose an adaptive Markov Chain-based model for trajectory prediction. Our system is extensively evaluated using real-world datasets, and experiment results unveil interesting findings: (1) For individual predictors, Bayes Networks outperform all others when data quality is good, while J48 delivers the best results when data quality is bad; (2) Ensemble predictors outperform individual predictors in general under all conditions; and (3) Ensemble predictor performance depends on user movement patterns.

Acknowledgments

This work has been supported by the Swiss National Science Foundation via the SwissSenseSynergy project (154458).

References

[1] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, M. Miettinen, The mobile data challenge: Big data for mobile computing research, in: Pervasive Computing, 2012.

[2] Weka 3: Data Mining Software in Java, `http://www.cs.waikato.ac.nz/ml/weka/index.html` (November 2016).

[3] A. Kirmse, T. Udeshi, P. Bellver, J. Shuma, Extracting patterns from location history, in: ACM SIGSPATIAL GIS 2011, http://www.sigspatial.org/, 2011, pp. 397–400.

[4] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti, Wherenext: A location predictor on trajectory pattern mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, ACM, 2009, pp. 637–646.

[5] Z. Ying, S. Yong, W. Yu, Nokia mobile data challenge: Predicting semantic place and next place via mobile data, in: Proceedings of Mobile Data Challenge by Nokia, 2012.

[6] V. Etter, M. Kafsi, E. Kazemi, M. Grossglauser, P. Thiran, Where to go from here? mobility prediction from instantaneous information, Pervasive and Mobile Computing 9 (6) (2013) 784 – 797.

[7] R. Lambiotte, A. Noulas, M. Pontil, S. Scellato, C. Mascolo, A tale of many cities: Universal patterns in human urban mobility.

[8] C. Song, Z. Qu, N. Blumm, A.-L. Barabasi, Limits of predictability in human mobility 327 (5968) (2010) 1018–1021.

[9] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, Mobility prediction based on machine learning, in: 2011 IEEE 12th International Conference on Mobile Data Management, Vol. 2, 2011, pp. 27–30.

[10] D. Ashbrook, T. Starner, Fusing gps to learn significant locations and predict movement across multiple users, Personal Ubiquitous Computing.

[11] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, A. T. Campbell, Nextplace: A spatio-temporal prediction framework for pervasive systems, in: Proceedings of the 9th International Conference on Pervasive Computing, 2011.

[12] M. Karimzadeh, Z. Zhao, L. Hendriks, R. d. O. Schmidt, S. la Fleur, H. van den Berg, A. Pras, T. Braun, M. J. Corici, Mobility and bandwidth prediction as a service in virtualized lte systems, in: Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on, IEEE.

[13] H. He, Y. Qiao, S. Gao, J. Yang, J. Guo, Prediction of user mobility pattern on a network traffic analysis platform, in: Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture, ACM, 2015, pp. 39–44.

[14] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, M. Kyriakakos, A. Kalousis, Predicting the location of mobile users: A machine learning approach, in: Proceedings of the 2009 International Conference on Pervasive Services, ICPS '09, ACM, New York, NY, USA, 2009, pp. 65–72.

[15] V. Etter, M. Kafsi, E. Kazemi, Been there, done that: What your mobility traces reveal about your behavior, in: Proceedings of Mobile Data Challenge by Nokia Workshop, 2012.

[16] W. Jingjing, P. Bhaskar, Periodicity based next place prediction, in: Proceedings of Mobile Data Challenge by Nokia Workshop, 2012.

[17] G. Huiji, T. Jiliang, L. Huan, Mobile location prediction in spatio-temporal context, in: Proceedings of Mobile Data Challenge by Nokia, 2012.

[18] L. Zhongqi, Z. Yin, Z. Vincent, Y. Qiang, Next place prediction by learning with multiple models, in: Proceedings of Mobile Data Challenge by Nokia, 2012.

[19] T. Le-Hung, C. Michele, M. Luke, A. Karl, Next place prediction using mobile data, in: Proceedings of Mobile Data Challenge by Nokia, 2012.

[20] T. M. T. Do, O. Dousse, M. Miettinen, D. Gatica-Perez, A probabilistic kernel method for human mobility prediction with smartphones, Pervasive and Mobile Computing 20 (C) (2015) 13–28.

[21] Y. Zhu, E. Zhong, Z. Lu, Q. Yang, Feature engineering for semantic place prediction, Pervasive and mobile computing 9 (6) (2013) 772–783.

[22] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of Artificial Intelligence Research 11 (1999) 169–198.

[23] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (1-2) (2010) 1–39.

[24] E. H.-C. Lu, C.-Y. Chen, V. S. Tseng, Personalized trip recommendation with multiple constraints by mining user check-in behaviors, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, SIGSPATIAL '12, ACM, New York, NY, USA, 2012, pp. 209–218.

[25] V. W. Zheng, Y. Zheng, X. Xie, Q. Yang, Towards mobile intelligence: Learning from gps history data for collaborative recommendation, Artif. Intell. 184-185 (2012) 17–37.

[26] Y. Zheng, X. Xie, Learning travel recommendations from user-generated gps traces, ACM Trans. Intell. Syst. Technol. 2 (1) (2011) 2:1–2:29.

[27] C. Anagnostopoulos, S. Hadjiefthymiades, Intelligent trajectory classification for improved movement prediction, IEEE Transactions on Systems, Man, and Cybernetics: Systems 44 (10) (2014) 1301–1314.

[28] T. Anagnostopoulos, C. B. Anagnostopoulos, S. Hadjiefthymiades, A. Kalousis, M. Kyriakakos, Path prediction through data mining, in: IEEE International Conference on Pervasive Services, 2007, pp. 128–135.

[29] C.-C. Hung, W.-C. Peng, W.-C. Lee, Clustering and aggregating clues of trajectories for mining trajectory patterns and routes, The VLDB Journal 24 (2) (2015) 169–192.

[30] B. Chapuis, A. Moro, V. Kulkarni, B. Garbinato, Capturing complex behaviour for predicting distant future trajectories, in: Proceedings of the

5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, MobiGIS '16, ACM, 2016, pp. 64–73.

[31] E. Tuv, A. Borisov, G. Runger, K. Torkkola, Feature selection with ensembles, artificial variables, and redundancy elimination, Journal of Machine Learning Research 10 (Jul) (2009) 1341–1366.

[32] R. E. Schapire, Explaining adaboost (2015).

[33] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid, Periodicity detection in time series databases, IEEE Transactions on Knowledge and Data Engineering 17 (7) (2005) 875–887.

[34] S. Koço, C. Capponi, On multi-class learning through the minimization of the confusion matrix norm, CoRR abs/1303.4015.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[36] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[37] T. G. Dietterich, Ensemble methods in machine learning, in: Proceedings of the First International Workshop on Multiple Classifier Systems, MCS '00, Springer-Verlag, London, UK, UK, 2000, pp. 1–15.

[38] L. Rokach, O. Maimon, Data Mining with Decision Trees: Theroy and Applications, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2008.

[39] R. J. A. Little, D. B. Rubin, Statistical Analysis with Missing Data, John Wiley & Sons, Inc., New York, NY, USA, 1986.

[40] D. Heckerman, Learning in graphical models, MIT Press, Cambridge, MA, USA, 1999, Ch. A Tutorial on Learning with Bayesian Networks, pp. 301–354.

[41] C. Fiot, G. A. P. Saptawati, A. Laurent, M. Teisseire, Learning Bayesian Network Structure from Incomplete Data without Any Assumption, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 408–423.

[42] B. Efron, Bootstrap methods: Another look at the jackknife, Ann. Statist. 7 (1) (1979) 1–26.

[43] B. Efron, S. for Industrial, A. Mathematics, The jackknife, the bootstrap, and other resampling plans, Philadelphia, Pa. : Society for Industrial and Applied Mathematics, 1982, notes from ten lectures given at Bowling Green State Univ., June 1980. Bibliography pp 91-92.

[44] R. Meir, G. Rätsch, Advanced lectures on machine learning, Springer-Verlag New York, Inc., New York, NY, USA, 2003, Ch. An Introduction to Boosting and Leveraging, pp. 118–183.

[45] K. M. Ting, I. H. Witten, Stacked generalization: when does it work?, in: in Procs. International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1997, pp. 866–871.

[46] L. Breiman, Stacked regressions, Machine Learning 24 (1) (1996) 49–64.

[47] S. Nagi, D. K. Bhattacharyya, Classification of microarray cancer data using ensemble approach, Network Modeling Analysis in Health Informatics and Bioinformatics 2 (3) (2013) 159–173.

Dr. Zhongliang Zhao received his Ph.D. degree from the University of Bern in 2014. Since 2014, he holds an appointment of a Senior Researcher with the University of Bern. He was appointed as a work package leader on several work packages of the EU FP7 MCN project and a Co-Primary Investigator (PI) of the Sino-Swiss Science and Technology Cooperation (SSSTC) project M3WSN. Currently, he is the Technical Coordinator of the SNSF SwissSenseSynergy project and the co-PI of the Orange-funded industry project Context Awareness Engine.

Mr. Mostafa Karimzadeh is currently a Ph.D. student in the University of Bern, under the supervision of Prof. Dr. Torsten Braun. He got the Master degree from Politecnico di Milano. His current research focuses on mobile user mobility prediction for location based services.

Mr. Florian Gerber is currently a bachelor student in the University of Bern, under the supervision of Prof. Dr. Torsten Braun. His bachelor thesis is about trajectory prediction for individual mobile users.

Prof. Dr. Torsten Braun got his Ph.D. degree from University of Karlsruhe (Germany) in 1993. From 1994 to 1995, he was a guest scientist at INRIA Sophia-Antipolis (France). From 1995 to 1997, he worked at the IBM European Networking Centre Heidelberg (Germany) as a project leader and senior consultant. Since 1998, he is a full professor of Computer Science at University of Bern. Currently, he holds an appointment of a vice president of the SWITCH (Swiss Research and Education Network Provider) Foundation since 2011. He was a Director of the Institute of Computer Science and Applied Mathematics at University of Bern between 2007 and 2011. He is serving as Deputy Dean of the Faculty of Science, University of Bern since 2017. He received best paper awards from LCN 2001, WWIC 2007, EE-LSDS 2013, WMNC 2014, and the ARMS-CC-2014 Workshop as well as the GI-KuVS Communications Software Award in 2009. In the scope of EU funded projects, he was leading WPs of FP6-EUQOS and FP7-MCN. Moreover, he coordinated national projects such as SNSF SwissSenseSynergy and SNSF CONTACT.

Dr. Zhongliang Zhao

Mostafa Karimzadeh

Florian Gerber

Prof. Torsten Braun

- An ensemble learning approach has been proposed to solve the crowd mobile users' location prediction problem.
- An adaptive Markov Chain-based trajectory prediction approach has been proposed to predict the future trajectory of mobile users.
- Hybrid types of features (temporal, spatial, and smartphone system features) of mobile users' mobility traces have been extracted and their correlations have been analyzed in the ensemble learning approach.
- Detailed performance evaluations of different machine learning approaches have been conducted over homogeneous and heterogeneous user movement patterns.
- Ensemble learning using hybrid features delivers the best performance when predicting future locations of mobile users.
- Complexity analysis of different machine learning approaches have been conducted to understand the drawbacks of each algorithm.