Mobile Crowd Location Prediction with Hybrid Features using Ensemble Learning

Zhongliang Zhao*, Mostafa Karimzadeh, 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 future places of mobile users is of increasing importance to support proactive information services. In this paper, we model this problem as a supervised learning task and present an approach using ensemble learning methods with hybrid types of features. We first characterize the properties of users' visited places and movement patterns and then extract feature types (temporal, spatial, and system) to quantify the correlation between places and features. Finally, we propose to use ensemble methods to predict mobile users' future locations with extracted features. To evaluate the system performance, we use a real-life dataset provided by the Nokia Mobile Data Challenge. Experiment results unveil three interesting findings: (1) For an individual algorithm-based predictor, the J48 decision tree-based approach outperforms the Bayes-based approach when data quality is poor; (2) Ensemble learning-based approaches always outperform individual state-of-the-art classifiers; and (3) The ensemble learning of Stacking is better than other ensemble methods, such as Bagging and Boosting.

Keywords: Human-centered computing, Empirical studies in ubiquitous computing, Feature selection, Supervised learning by classification, Ensemble learning, Next place prediction, Heterogeneous data.

*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 this work, we formulate the location prediction problem as a standard supervised machine learning task, where a 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 learning 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 [1], 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 prediction 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 [2] [3]. 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 [4] [5]. However, issues such as the consideration of other rich contextual data, such as accelerometer, Bluetooth/WiFi connectivity, call/sms logs, information about running applications, etc. 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 that has been visited by the user. Therefore, our approach focuses on selecting a wide range of discriminative features, including temporal, spatial, and smart-phone system features.

The Nokia Mobile Data Challenge (MDC) data set [6] holds great potential for providing fine-quality information to predict a user's next place. 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 people visits of certain places follow some regular patterns. Moreover, people behaviors at specific locations also provide useful information for certain predictions. For example, if a person is taking public transportation, he/she must have a certain speed or speed variation.

In this work, we extract features that have discriminative information among different locations. With the extracted features, we use WEKA to xxx. The main contributions of this work are summarized as follows.

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

- Second, we evaluate prediction accuracies of multiple machine learning algorithms (individual and ensemble ones).

- Third, from the experiment results, we unveil some interesting findings about how the prediction performance is affected by various factors such as mobility trace qualities, extracted features, types of ensemble methods, etc.

The structure of this paper is as follows. Section 2 discusses existing efforts on location 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. Several prediction methods have been proposed for human mobility in different contexts. Ashbrook et al. [9] proposed 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 [10] 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. [11] [12] proposed

a Dynamic Bayesian Network-based model to predict the future cells of mobile users to optimize telecommunication network operations. He et al. [13] proposed 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 next place prediction task of Nokia Mobile Data Challenge 2012, the best methods relied only on spatial-temporal information to predict future locations [14], [15], [16], [17]. For instance, Lu et al. [17] 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. [18] 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. [19] 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. [20] proposed 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 [21] [22]. Therefore, we focus on applying different ensemble learning methods to optimize prediction accuracy.

3. MDC Dataset

Our experiment data is from the Nokia Mobile Data Challenge (MDC) [6], 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, that 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 discrete 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: more than 1500 instances
- Good: 1200-1500 instances
- OK: 1000-1200 instances
- Bad: 800-1000 instances
- Very bad: less than 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 a lot of different places for very few times during the data collection period. Their movements are heterogeneous and it's 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 show an example of homogeneous movement, since the user visits very few places frequently. Figure 2 shows an example of heterogeneous movement, where the user visited many different places occasionally.

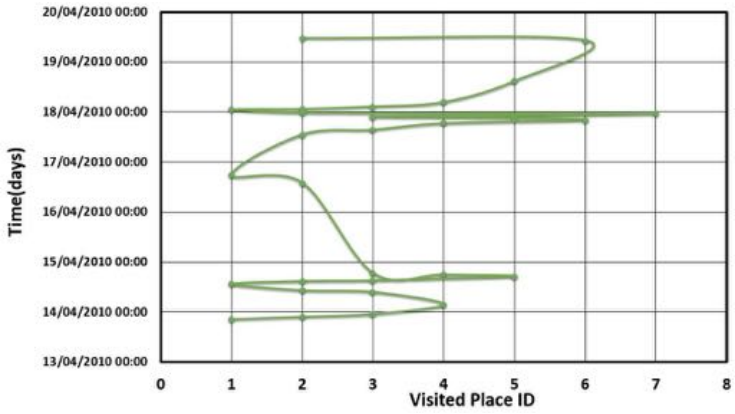


Figure 1: Homogeneous movements.

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

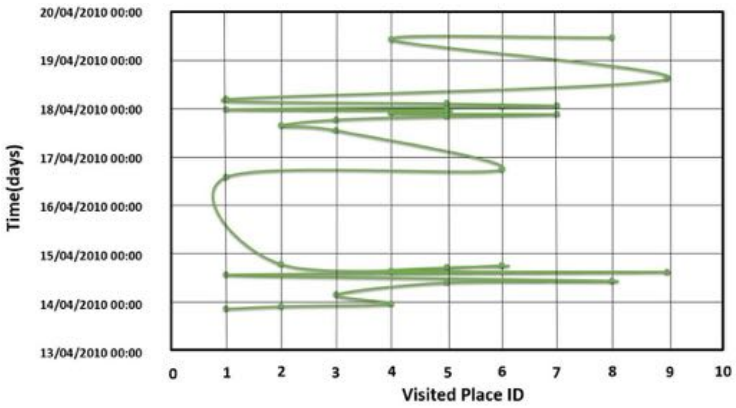


Figure 2: Heterogeneous movements.

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

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. Features 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 (e.g., number of detected WLAN, acceleration data, etc.) 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.

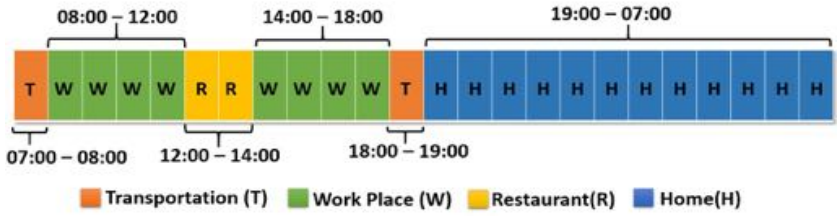


Figure 3: Day time decomposition.

- **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 during the data collection period.

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' embedded motion sensors. It can be used to detect changes of movement types, for instance if a user is detected to change from slow speed movement to fast speed movement, then most probably he is at 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 [23]. 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. 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.

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

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	–

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 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. Figure 4 shows the J48 decision tree with the extracted features. In this example, 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.

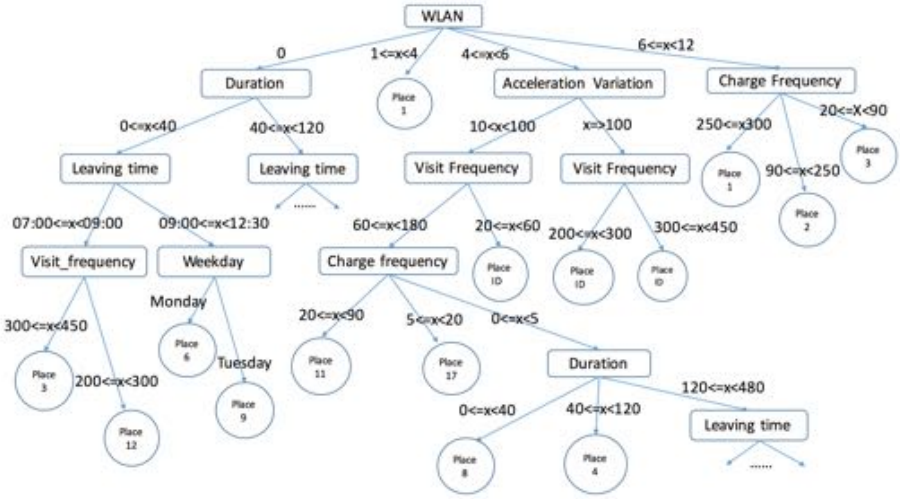


Figure 4: A J48 decision tree.

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.

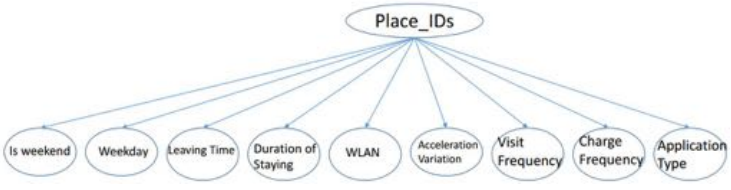


Figure 5: A Directed Acyclic Graph (DAG) of Bayesian networks.

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.

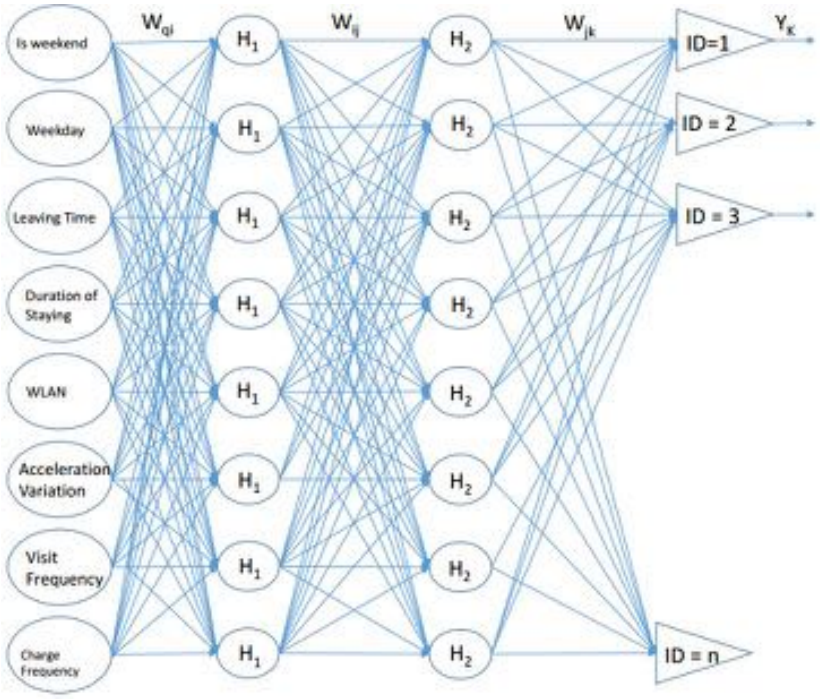


Figure 6: A typical two-layer Multilayer Perceptron Architecture.

5.2. Ensemble Predictors

Ensemble learning is an approach to combine individual predictors to achieve better performance. As different users tend to have different mobility patterns, there is no single predictor that could outperform the others for all users. Therefore, we focus on finding the suitable prediction model for a certain mobility pattern and efficiently combine all the models to deliver the optimized performance. The task of constructing an ensemble classifier can be broken into two sub-tasks: (1) selection of a diverse set of base models or classifiers with consistently 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

This 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. *Boosting* relies on iteration, which uses the outcome of the previously built models as inputs of the second model to improve performance. The *AdaBoost* algorithm was the first practical boosting algorithm, widely used and studied in numerous applications and research fields [24]. WEKA uses an updated version of *AdaBoost*, called *AdaBoostM1* scheme. In this work, we integrate individual algorithms like: *J48*, *Random Forest*, *Bayes Network*, *Naive Bayes* and *MLP* with *AdaBoostM1*.

5.2.2. Bagging

This is an ensemble method that divides the training data set into several different sets (called 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 Network*, *Naive Bayes* and *MLP*.

5.2.3. Stacking

The Stacking method 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 Network*, 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 use 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 with Intel vPro (64-bit-X68 architecture) core i7 CPU 3.2 GHz. The laptop is running Windows 8.1 Enterprise, with 16 GB memory. WEKA is installed and properly configured.

6.1. Machine Learning Approaches and Parameters

In this work we use WEKA [1], which is an open source machine learning framework, 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 *J48*, *Random Forest* (from Decision Tree predictors), *Bayes Network*, *Naive Bayes* (from Bayes predictors) and *Multilayer Perceptron (MLP)* (from Neural Network predictors) machine

learning 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*. Each time, one of the 10 subsets is used as the test set and the other 9 subsets are put together to form a training set. The advantage of this method is that it matters less how the data gets divided. Every data subset gets to be in a test set exactly once, and gets to be in training set 9 times. 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*
T	Number of trees to generate in *Random Forest*	20
L	Number of possible iterations for individual algorithms in *Boosting*	10
N	Number of new generated training sets in *Bagging*	10
J	Number of new generated training sets in *Stacking*	10

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. Prediction accuracy refers to the percentages of correct location prediction, and prediction time refers to the execution time of performing the prediction task.

6.2.1. Average Prediction Accuracy of Individual Algorithms

Fig. 7 and Fig. 8 show the average prediction accuracy of different individual algorithms using temporal, spatial, as well as hybrid features. The results clearly demonstrate that the mechanisms relying on the Decision Tree family (specially the *J48* algorithm) outperform others, while using the traced data with lower quality. We can also see that the *Bayes Network* scheme provides better performance ($> 84\%$ accuracy) for the traced data with higher quality. Moreover, it can be observed that the estimated accuracy is improved significantly when the hybrid features are used instead of using only temporal+spatial features. For instance, *Bayes Network* delivers an accuracy of 84.76% with hybrid features, while only 55.47% can be reached with temporal+spatial features.

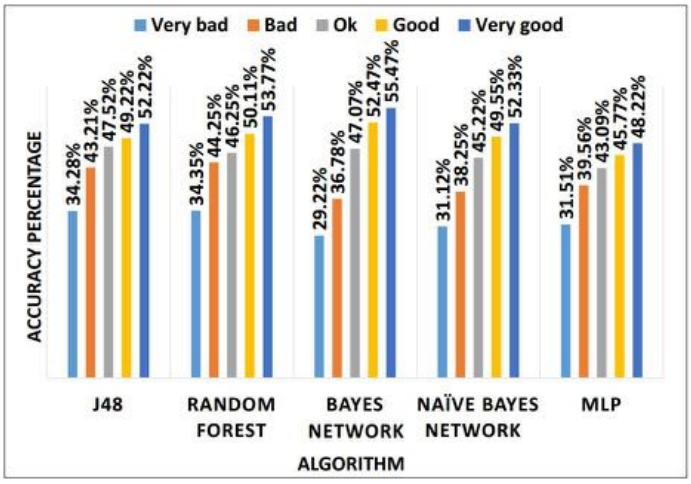


Figure 7: Prediction accuracy of individual algorithms using Temporal+Spatial features.

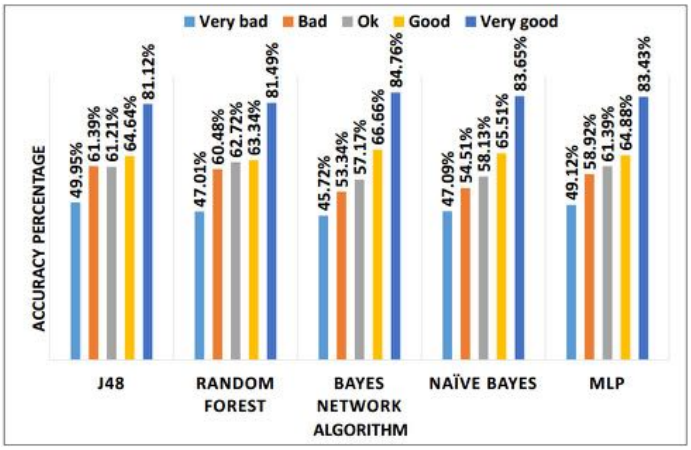
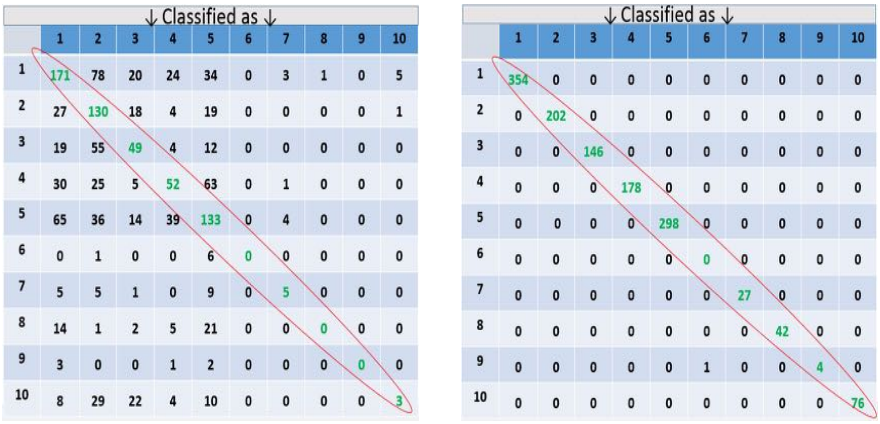


Figure 8: Prediction accuracy of individual algorithms using Hybrid features.

Fig. 9 shows two confusion matrices that help to explain the nature of the errors made by the classifier with temporal+spatial and hybrid features[25]. 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 whose real class type are 1 were wrongly predicted as class 2, and 171 places with real class type of 1 were correctly predicted. These matrices are generated by the *J48* algorithm over the 10 most visited places (indicated by IDs). For instance, the matrix of Fig. 9a shows that when the prediction algorithm uses only the temporal+spatial features, accuracy of prediction is lower and several incorrect predictions are observed. The number of correct predictions are significantly improved when the hybrid features are used, as shown in Fig. 9b.



↓ 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 9: Confusion matrices using different features.

6.2.2. Average Prediction Accuracy of Ensemble Learning Methods

In this subsection, we present the prediction accuracy of different ensemble learning algorithms. Fig. 10 and Fig. 11 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 Network* and *Boosting* outperforms the others.

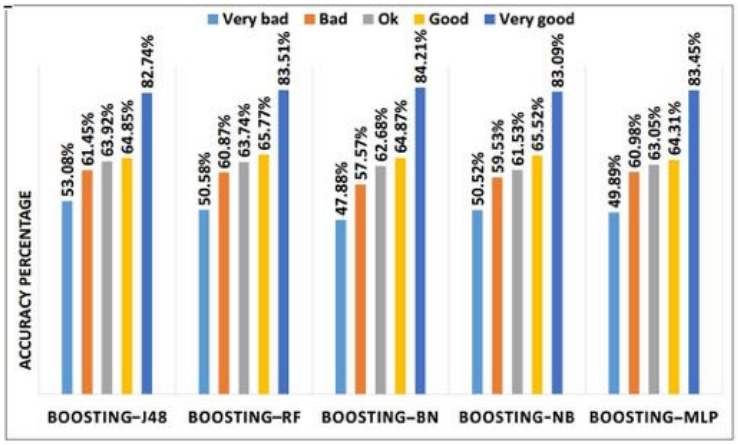


Figure 10: Prediction accuracy of *Boosting*.

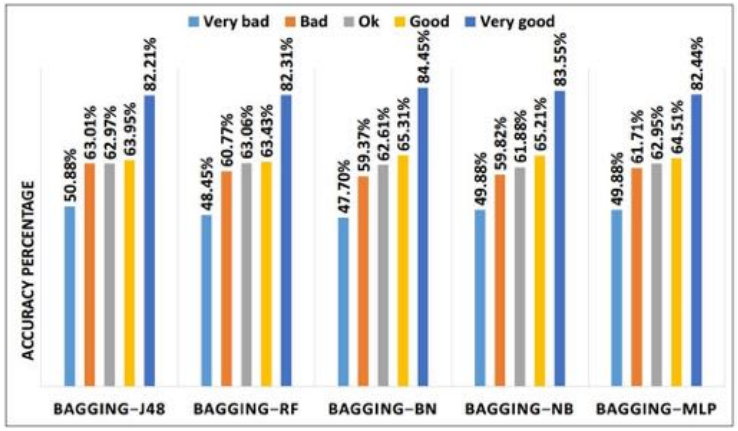


Figure 11: Prediction accuracy of *Bagging*.

Fig. 12 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 Network*, and *MLP*, we decided to integrate them using *Stacking*. The *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 we used an integration of *J48*, *Bayes*

Network and *MLP* mechanisms, particularly when the trace data is of high quality.

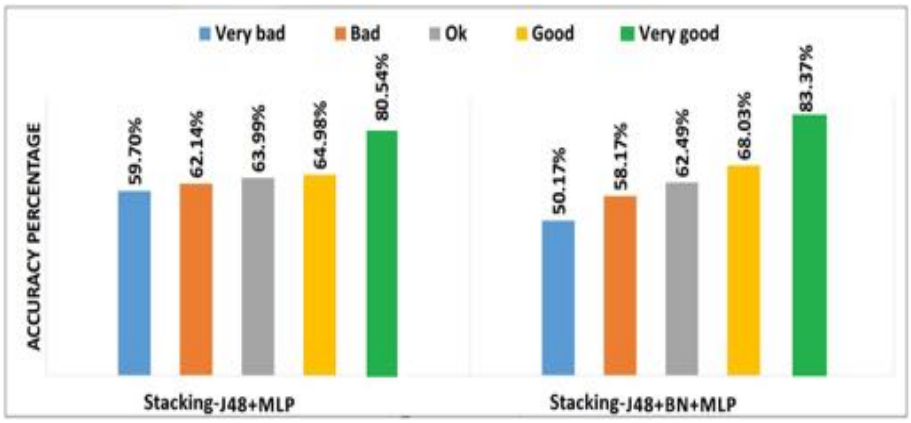


Figure 12: Prediction accuracy of *Stacking*

6.2.3. Average Execution Time of Individual Algorithms

In addition to prediction accuracy, we are also interested in the execution time to make the prediction. We measure the execution time of each individual algorithm using temporal+spatial features and hybrid features. The obtained results, as shown in Fig. 13 and Fig.14, indicate that the *Decision Tree* and *Bayes* families could generate the prediction faster. *MLP* is the one needing more execution time compared to the others.

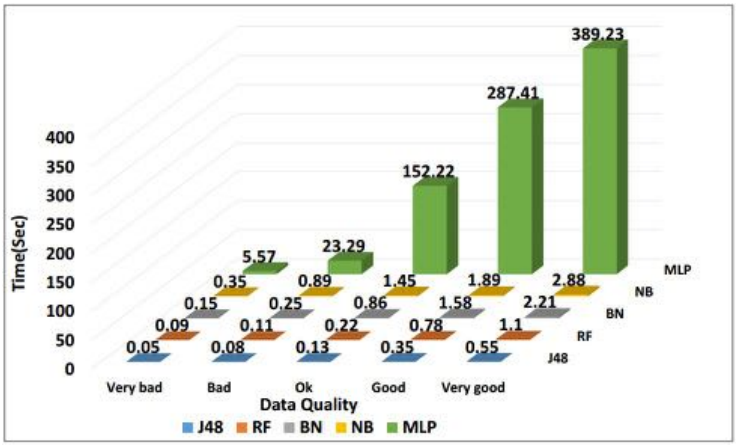


Figure 13: *Average execution time of individual algorithms using Temporal+Spatial features.*

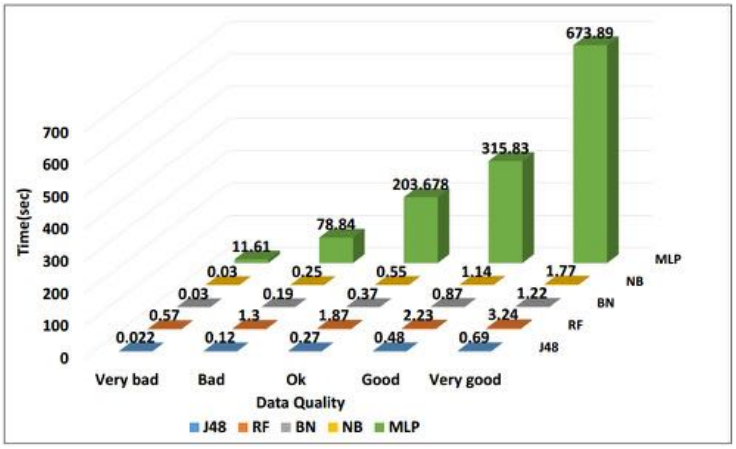


Figure 14: *Average execution time of individual algorithms using Hybrid features.*

6.2.4. Average Execution time of Ensemble Learning Methods

Fig. 15 16, Fig. 17 18, and Fig. 19 20 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 Network* 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.

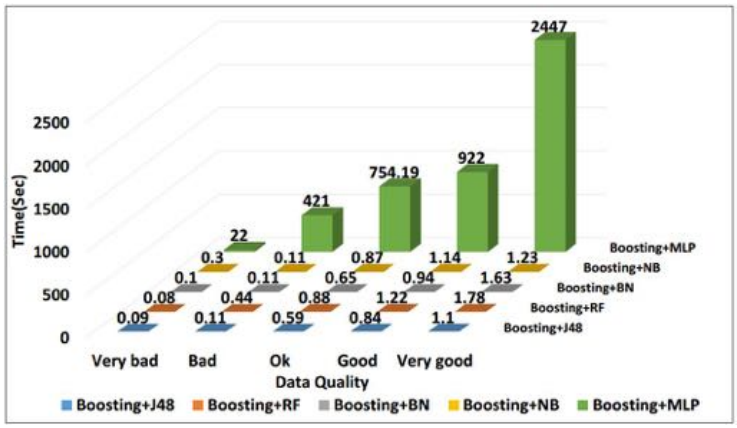


Figure 15: Average execution time of *Boosting with Temporal+Spatial features.*

Figure 16: Average execution time of *Boosting with Hybrid features.*

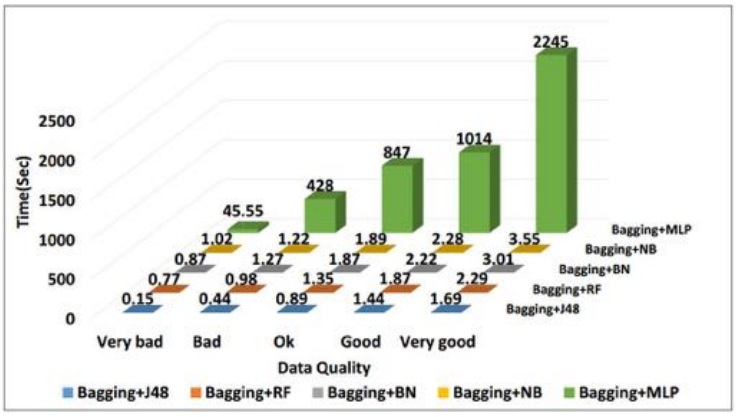


Figure 17: Average execution time of *Bagging with Temporal+Spatial features.*

Figure 18: Average execution time of *Bagging with Hybrid features.*

Figure 19: Average execution time of *Stacking with Temporal+Spatial features.*

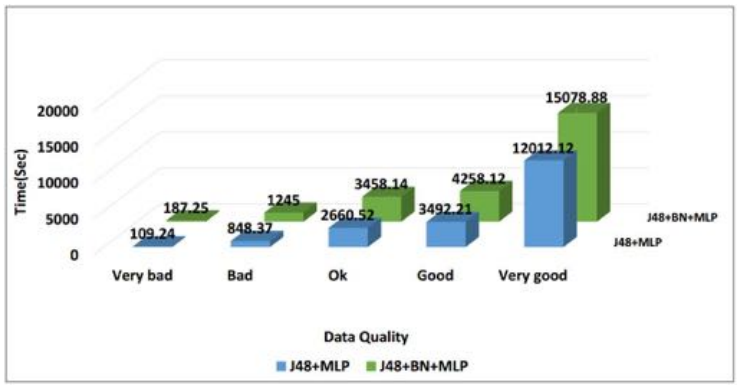


Figure 20: Average execution time of *Stacking with Hybrid features.*

6.3. Performance Comparison with Past Studies

In this section, we present the performance comparison with past correlated studies to show the superiority of our solutions. We take relevant location prediction accuracy results from [17] [5], 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.

Work	**Algorithms**	**Features**	**Best Accuracy (%)**
Our work	Stacking	Hybrid features	83.37
HKUST [17]	Gradient Boosting Trees	Limited hybrid features	76.32
EPFL [5]	Blending	Temporal features	56.22

As we can see from Table 5, our solutions significantly outperform the others. This is because in [5], 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 [17], 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.

6.4. Theoretical Analysis

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

6.4.1. 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 (specially the *J48* algorithm) outperform others when the trace data is of lower quality, while the *Bayes Network* 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 Network* by decomposing the mathematical components of each model to explain why different predictors

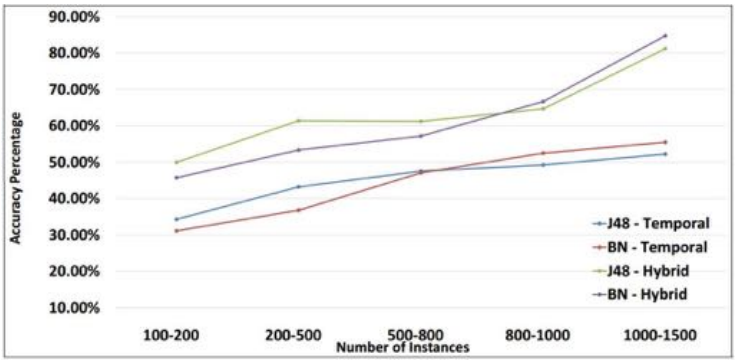


Figure 21: Prediction accuracy of *J48* and *Bayes Network*

have different performance. Fig. 21 shows the average prediction accuracy of the *J48* and *Bayes Network* algorithms using temporal or hybrid features as a function of trace qualities, which are summarized in Fig. 7. It is interesting to observe that for both cases, *J48* outperforms *Bayes Network* 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 [26]. However, the *Bayes Network* has no 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. Assuming that a_i denotes either numerical attributes (e.g., leaving time, duration of staying in each place, place id, etc), or nominal attributes (e.g., application type, is weekend, is weekday, 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 [27] 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.

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

$\widehat{V}_{i,j}$ 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. 21, the *J48* and *Bayes Network* algorithms generate similar results when 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 Network* overtakes *J48*, when the quality is better (e.g., with 700-1500 instances). This is due to the fact that *Bayes Network* follows a graphical model, making possible relations between the parameters with particular probabilities [28]. 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 set of local probability distributions to define the joint probability distribution [29]. The joint probability distribution is defined as in Eq. 2.

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

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.

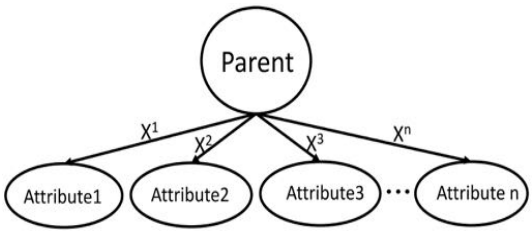


Figure 22: Directed Acyclic Graph (DAG) of Bayes Network.

475 *6.4.2. Analysis of Ensemble Learning Algorithm Performance*

As presented in Fig. 11 and Fig. 12, the experiment results show that the integration of the individual algorithms (e.g., *J48*, *Bayes Network* 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.

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

$f(x)$, $g(x)$, $E[g(x)]$ 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 learning methods can be applied to enhance the prediction performance of individual algorithms by mitigating the variance er-
480 ror.

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* performs this by creating N new subsets of training data sets with the same size, as shown in Table 4. The new data sets are generated from the original data, randomly sampled and replaced [30]. There-

fore, 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 [31].

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

Boosting applies a sequential model in the learning phases [32]. 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 6.

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

Table 6: 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 [33]. 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 [34] [35] 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 visited places and features. Finally, we propose to use ensemble machine learning approaches to predict users' future locations. Our system is extensively evaluated using real-world datasets, and experiment results unveil three interesting findings: (1) For individual algorithm-based predictors, a J48 decision tree-based approach outperforms Bayes-based approach when data quality is poor; (2) Ensemble learning-based approaches always outperform individual state-of-the-art classifiers; and (3) Stacking is better than other ensemble methods such as Bagging and Boosting.

Acknowledgments

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

References

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

[2] 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.

[3] 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. doi:10.1145/1557019.1557091.

[4] 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 Workshop at the Tenth International Conference on Pervasive Computing, 2012.

[5] 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. doi:https://doi.org/10.1016/j.pmcj.2013.07.006.

[6] 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.

[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. arXiv:http://www.sciencemag.org/cgi/reprint/327/5968/1018.pdf, doi:10.1126/science.1177170.

[9] D. Ashbrook, T. Starner, Fusing gps to learn significant locations and predict movement across multiple users, Personal Ubiquitous Computing.doi:10.1007/s00779-003-0240-0.

[10] 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, pp. 152–169.

[11] Z. Zhao, M. Karimzadeh, T. Braun, A. Pras, H. van den Berg, A demonstration of mobility prediction as a service in cloudified lte networks, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 78–80. doi:10.1109/CloudNet.2015.7335285.

[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, 2015, pp. 132–138.

[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] 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 at the Tenth International Conference on Pervasive Computing, 2012.

[15] W. Jingjing, P. Bhaskar, Periodicity based next place prediction, in: Proceedings of Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing, 2012.

[16] G. Huiji, T. Jiliang, L. Huan, Mobile location prediction in spatio-temporal context, in: Proceedings of Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing, 2012.

[17] L. Zhongqi, Z. Yin, Z. Vincent, Y. Qiang, Next place prediction by learning with multiple models, in: Proceedings of Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing, 2012.

[18] T. Le-Hung, C. Michele, M. Luke, A. Karl, Next place prediction using mobile data, in: Proceedings of Mobile Data Challenge by Nokia Workshop at the Tenth International Conference on Pervasive Computing, 2012.

[19] 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. doi:10.1016/j.pmcj.2014.09.001.
URL http://dx.doi.org/10.1016/j.pmcj.2014.09.001

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

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

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

[23] 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.

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

[25] S. Koço, C. Capponi, On multi-class learning through the minimization of the confusion matrix norm, CoRR abs/1303.4015.
URL http://arxiv.org/abs/1303.4015

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

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

[28] D. Heckerman, Learning in graphical models, MIT Press, Cambridge, MA, USA, 1999, Ch. A Tutorial on Learning with Bayesian Networks, pp. 301–354.
URL http://dl.acm.org/citation.cfm?id=308574.308676

[29] 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. doi:10.1007/978-3-540-78568-2_30.
URL http://dx.doi.org/10.1007/978-3-540-78568-2_30

[30] B. Efron, Bootstrap methods: Another look at the jackknife, Ann. Statist. 7 (1) (1979) 1–26. doi:10.1214/aos/1176344552.
URL http://dx.doi.org/10.1214/aos/1176344552

[31] 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.

[32] 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.
URL http://dl.acm.org/citation.cfm?id=863714.863719

[33] 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.

[34] L. Breiman, Stacked regressions, Machine Learning 24 (1) (1996) 49–64. doi:10.1007/BF00117832.
URL http://dx.doi.org/10.1007/BF00117832

[35] 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. doi:10.1007/s13721-013-0034-x.
URL http://dx.doi.org/10.1007/s13721-013-0034-x