Applying Machine Learning Techniques to Transportation Mode Recognition Using Mobile Phone Sensor Data

Article *in* IEEE Transactions on Intelligent Transportation Systems · October 2015

DOI: 10.1109/TITS.2015.2405759

Arash Jahangiri
San Diego State University
18 PUBLICATIONS **49** CITATIONS

Hesham Rakha
Virginia Polytechnic Institute and State University
305 PUBLICATIONS **3,702** CITATIONS

Applying Machine Learning Techniques to Transportation Mode Recognition using Mobile Phone Sensor Data

Arash Jahangiri and Hesham A. Rakha, *Member, IEEE*

***Abstract*— The paper adopts different supervised learning methods from the field of machine learning to develop multi-class classifiers that identify the transportation mode, including: driving a car, riding a bicycle, riding a bus, walking, and running. Methods that were considered include K-Nearest Neighbor (KNN), Support Vector Machines (SVMs), and tree-based models that comprise a single Decision Tree (DT), Bagging (Bag), and Random Forest (RF) methods. For training and validating purposes, data were obtained from smartphone sensors including accelerometer, gyroscope, and rotation vector sensors. K-fold Cross-Validation as well as Out-of-Bag error was used for model selection and validation purposes. Several features were created from which a subset was identified through the minimum Redundancy Maximum Relevance (mRMR) method. Data obtained from the smartphone sensors were found to provide important information to distinguish between transportation modes. The performance of different methods was evaluated and compared. The Random Forest (RF) and Support Vector Machine (SVM) methods were found to produce the best performance. Furthermore, an effort was made to develop a new additional feature that entails creating a combination of other features by adopting a Simulated Annealing (SA) Algorithm and Random Forest (RF) method.**

***Index Terms*—Cellular phone sensor data, machine learning algorithms, transportation mode recognition.**

I. Introduction

DISTINGUISHING between different types of physical activities using sensor data has been a recent research topic that has received considerable attention [1, 2]. Transportation mode detection can be considered as an activity recognition task in which data from smartphone sensors carried by users are utilized to infer what transportation mode the individuals have used. Micro-electromechanical systems (MEMS), such as accelerometers and gyroscopes are embedded in most smartphone devices [3] from which the data can be obtained at high frequencies. Smartphones, nowadays, are equipped with powerful sensors such as GPS, accelerometer, gyroscope, light sensors, etc. Having such powerful sensors all embedded in a small device carried in everyday life activities has enabled researchers to investigate new research areas. The advantages of these smart devices include ubiquity, ability to send and receive data through various ways (e.g. Wi-Fi/cellular network/Bluetooth), and storing/processing data [4].

The knowledge of individuals' mode of transport can facilitate some tasks and also can be adopted in several applications as follows:

1) Knowing the mode of transportation is an essential part of urban transportation planning, which is usually investigated through questionnaires/travel diaries/ telephone interviews [4, 5]. This traditional way of surveying is usually expensive, erroneous, limited to a specific area, and not so up-to-date [6].
2) As environmental applications, the carbon footprint as well as the amount of calories burnt of individuals can be determined by obtaining the mode of transport. Also, physical activities and health can be monitored, the hazard exposure can be tracked, and the environmental impacts of one's activities can be assessed [7].
3) Other applications include providing users with real-time information using the knowledge of speed and transport mode from the users as probes [4, 8], providing individuals with customized advertisements and messages based on the transportation mode they are using [4].

Many studies have used Global Positioning System (GPS) data for classification purposes. However, several limitations are associated with the use of GPS sensors. These limitations include: GPS information is not available in shielded areas (e.g. tunnels) and the GPS signals may be lost especially in high dense locations, which results in inaccurate position information. Moreover, the GPS sensor consumes significant power that sometimes users turn it off to save the battery [6, 8]. This paper focuses on developing detection models using machine learning techniques and data obtained from smartphone sensors including accelerometer, gyroscope, and rotation vector, without GPS data. Consideration of multiple sensors is beneficial in that even without using GPS, the transportation modes can be identified.

The present research demonstrates how to apply several machine learning techniques, including: K-Nearest Neighbor

Manuscript submitted May 11, 2014. This work was supported in part by the Connected Vehicle Initiative University Transportation Center (CVI-UTC), the Mid-Atlantic University Transportation Center (MAUTC), and the TranLIVE University Transportation Center.

A. Jahangiri and H. A. Rakha are with the Center for Sustainable Mobility, Virginia Tech Transportation Institute, 3500 Transportation Research Plaza, Blacksburg, VA 24061 USA (e-mail: arashj@vt.edu; hrakha@vtti.vt.edu).

(KNN), Support Vector Machines (SVMs), and tree-based models that comprise a single Decision Tree (DT), Bagging (Bag), and Random Forest (RF) methods to identify transportation modes using data obtained from smartphone sensors. The data include acceleration extracted from the accelerometer sensor, rate of device rotation extracted from the gyroscope sensor, and device's orientation extracted from the rotation vector sensor; all these were extracted around different coordinate axes; more details on feature selection are presented in sections "data collection, preprocessing, and feature extraction" on page 4 and "feature selection" on page 6. Previous studies lacked adequate simultaneous consideration of several factors, whereas this study is unique in that it comprehensively and simultaneously considers all these factors to obtain a naturalistic data which better reflects real world situations. To the best of our knowledge, items 4, 5, 6, and 8 (listed below) have not been considered in the past literature. To summarize, these factors include:

1) The research considered both motorized (car and bus) and non-motorized modes of travel (bike, walk and run).
2) The research did not require that the travelers maintain a fixed location for their phone as was done in other studies (e.g. phone must be in the traveler's pocket).
3) The research did not use data from GPS sensors because GPS sensors can deplete the phone battery and the signal may be lost in urban areas.
4) The research made use of data from gyroscope and rotation vector sensors, which was never used in previous transportation mode detection studies. Some features were created based on data from these two sensors.
5) The research considered motorized travel (car and bus) on different road types with different speed limits (e.g. 15, 25, 35, 45, and 65 mph speed limits). This wide range of speeds was selected to ensure that the algorithms developed would be robust to different travel conditions.
6) The data collection required travelers to collect bus, car, and bike data along routes where they had to stop at different intersections and thus data included data in traffic jam conditions.
7) The research considered all common machine learning procedures in the development of the models, namely; complete model selection, regularization (applied when using SVM), feature selection, and feature scaling (applied when using SVM and KNN).
8) The research created a large number of features from which the most representative features were selected for model development. More details about the examined features are presented in sections "data collection, preprocessing, and feature extraction" on page 4 and "feature selection" on page 6.
9) The research identified the features based on statistical measures of dispersion as well as derivatives to obtain variations over the time window of interest and consequently incorporated this knowledge (i.e. feature time dependency) into the models.

Some of the study challenges included: (1) Data synchronization; since it was not possible to store the data from different sensors at specific times and thus data were resampled at a desired frequency, (2) High frequency of data; due to having data at very high frequencies. Many data points were available even in small time window sizes. Consequently, using statistical measures of dispersion, raw data values were replaced with those measures corresponding to the time window of interest, (3) High computations; optimizing the parameters of different models in the model selection task, required high computations and thus to alleviate this problem, statistical measures of dispersion were replaced with all the data points within the time window of interest to decrease the number of data points. In addition, the mRMR feature selection method was employed to select the most representative features before developing different models. A characteristic of this method was that it was independent of the models and it was not required to carry out the feature selection task for each model; hence it was conducted only once, and (4) Data noise; a preprocessing task was conducted for noise reduction.

The remainder of the paper is organized in the following sections: Relevant literature is reviewed in the next section followed by the data collection section. Subsequently, it is shown how the detection models were developed. Subsequently, the results of the study are presented and finally the conclusions of the study are presented.

II. Relevant Work

Almost all studies used data from GPS sensors that have the aforementioned drawbacks. Also, they took advantage of Artificial Intelligence (AI) tools such as Fuzzy Expert Systems as in [9], Decision Trees as in [4-8, 10], Bayesian Networks as in [4, 10], Random Forests as in [4], Naïve Bayesian techniques as in [4, 7], Neural Networks as in [11, 12], and Support Vector Machine (SVM) techniques as in [7, 10, 13-16], of which the Decision Tree and SVM methods were used the most. To improve the model performance, other techniques were also combined with machine learning methods such as Discrete Hidden Markov Models as in [7] and Bootstrap aggregating as in [17]. Other than AI tools, statistical methods were also applied such as the Random Subspace Method in [18]. Some studies have used additional information from Geographic Information System (GIS) maps as in [4, 9, 19, 20]. However, GIS data is not always available, and also this approach may not be suitable for real-time applications because it mostly relies on the knowledge of the entire trip with respect to the GIS features such as bus stops, subway entrances, and rail lines.

The Decision Tree method was identified as the best method in [7, 10] compared to other methods including SVM. However, in developing the models, several factors need to be considered to obtain the best possible model performance. It appears that similar studies lack at least one of the following:

- Conducting a complete model selection
- Considering regularization
- Using feature selection methods

▪ Considering feature scaling

A complete model selection is equivalent to incorporating all the tuning parameters in order to obtain the best detection accuracy. Regularization is included in the model to deal with the issue of over-fitting (high variance). Feature selection methods are adopted to use the most representative features. Feature scaling is applied to normalize the range of different features (or attributes), which leads to higher model performance and training speed. However, it should be noted that in general not all these factors are always required. For example, selecting features based on intuition or expert knowledge may lead to as good results as using a feature selection method. It is also possible that these factors specially feature scaling (since it is a simple procedure) were part of the software package that was used in their work, but the authors were not clear whether the factors were applied or they just did not emphasize or focus on the importance of these factors. Nevertheless, these are important factors to be considered when solving machine learning problems.

Depending on the application of interest, different time window sizes have been used for detecting the mode of travel. For example, [12] found that longer monitoring durations lead to higher accuracy. Intuitively, the bigger the time window size the easier the detection becomes since with bigger window sizes more information is available. If the application is a survey for demand analysis the time window size can be as large as trip duration, whereas if the application provides real-time information for environmental or some transit applications, then smaller time window sizes are more desirable. The size should be as small as possible for some safety applications (e.g. crash prevention). An earlier study [13] used 200-meter and 150-second segments in their experiment. Whereas another study [6] used 10-second time windows to separate walking from non-walking segments and then applied a maximum size of 2 minutes. Other than the time window size, the overlaps of two consecutive windows have also been considered. Reference [8] obtained the best time window size and overlap to be 10.24 seconds and 50%, respectively. The entire trip duration appears to be considered in [5, 9, 11, 16, 20]. Higher accuracies are achieved by increasing the time window size as shown in [12]. However, the focus of this study is on small time window sizes, so the developed models have the potential to be used in a broader range of applications such as environmental and safety applications.

Other than the time window size, several factors that also influence the model performance are as follows:

1) *Number of classes:* as the number of classes increases, class differentiation becomes more difficult.
2) *Data Sources:* Less dependent models are more desirable as they can be applicable even with limited sources of data. For example, a model that requires two data sources is less dependent than a model that requires five data sources. Also, the data source reliability is of great importance; a model that requires data from sources such as accelerometer or gyroscope is more desirable than a model that requires data from the GPS sensor. The reason is that the data from accelerometer and gyroscope are available all the time, but GPS my not be available in some conditions. So a model which employs say the accelerometer (and/or gyroscope) is superior to a model that uses GPS data because accelerometer data are always available and the model can perform all the time.
3) *Ability to distinguish between motorized classes:* as different motorized classes have similar characteristics such as speed and acceleration, a model capable of differentiating between these modes is of great value. For example, distinguishing the bus mode from the car mode is significantly more difficult than discriminating walking from driving.
4) *Sensor positioning:* it shows how realistic the experiments are conducted. Positioning the devices in certain locations increases the detection accuracy because the movements monitored by the sensors show the movements of the transportation mode (or the person) they are attached to. However, it may not reflect realistic behavior of the travelers. Some of the studies required that the participants attach sensors/smartphones to different parts of their body.

Different detection accuracies have been reported by different studies. Although in almost all of them including our previous work, comparisons were drawn between the accuracies obtained from their approaches with those of others, such comparisons were excluded in the present study because of two reasons. First, in different studies, models were developed on different data sets. Second, several factors can affect model performance (e.g. time window size, number of classes, etc.). Here are some examples: Excluding those studies that assumed the time window size to be the entire trip, the highest reported accuracy of 96.9% was achieved by [15] with a time window size of 4 seconds. In their approach, they only used accelerometer data and did not rely on GPS and GIS data. Their method was capable of differentiating between motorized modes (car and train) and no specific sensor positioning was applied. Nevertheless, they only considered three classes. The second best accuracy was obtained by [14]. They also used accelerometer data without relying on GPS/GIS data. However, although different motorized modes were mentioned in the paper, it seems that the reported accuracies show only one motorized mode. Also, subjects in their study were asked to keep their device in their pocket of the non-dominant hip while collecting data which is more realistic compared to attaching sensors to the body, but still does not reflect a complete realistic behavior. An accuracy of 93.6% was reported by [7]. They applied the lowest time window size throughout the literature which is one second. However, their approach was dependent on data from GPS sensors. Further, different motorized classes were not considered.

III. Data collection, preprocessing and feature extraction

A smartphone application was developed for the purpose of data collection. The application stores the data coming from smartphone sensors including GPS, Accelerometer, Gyroscope, and Rotation Vector at the highest possible frequency. To collect the data, ten employees at Virginia Tech Transportation Institute (VTTI) were asked to carry a smartphone (two devices were used: a Galaxy Nexus and a Nexus 4) with the application installed on it on multiple trips. They were asked to select the travel mode they intend to use before starting the logging process, and then using the application buttons they were able to start and stop data logging. Although smartphones can be carried in other places, to make sure the data collection is less dependent on the sensor positioning, the travelers were asked to carry the smartphone in different positions that they normally do such as in pocket, in palm, in backpack, and different places inside car (e.g. on front right seat, coffee holder alongside of the driver) as they reported after the data collection. However, the amount of time that was spent for different positions were unknown since the participants were not asked to collect data in a particular position for a certain amount of time and the reason was to make the data collection as natural as possible. Data collection was conducted on different workdays (Mon through Fri) during working hours (8 AM to 6 PM) on different road types with different speed limits (i.e. car mode on roads with 15, 25, 35, 45, and 65 mph; bus mode on roads with 15, 25, 35, and 45 mph; bike mode on roads with 15, 25, and 35 mph) in Blacksburg, Virginia. Thirty minutes worth of data for each mode per person were collected. The original data frequency was about 25 Hz (for accelerometer, gyroscope, and rotation vector sensors), but the data from different sensors were not synchronized. Thus, in order to ensure that the data were gathered at identical sampling rates, linear interpolation was first applied to the data similar to [8] to produce continuous data sets and then the data were re-sampled at the desired rate (rate of 100 Hz was applied). Since the original frequency of 25 Hz was not a constant rate (i.e. a constant frequency was not possible to set for collecting data), the choice of 100 was made to make sure no information is lost. Furthermore, a low pass filter was used for noise reduction. In total, 25 hours of data (30 minutes per mode per person) were stored and used for training and testing purposes. In other words, total of ten travelers collected 30 minutes of data for each mode that equals (30x10x5)/60 = 25 hours.

Some features are considered to be basic/traditional features (e.g. mean speed, mean acceleration), which are more intuitive to be influential and were widely used in the literature and some are considered to be more advanced features (e.g. heading change rate) as presented by [13]. In our previous work [21], we used approximately 60 features that we created from the sensor data, mostly based on some statistical measures of dispersion. In the present study, we created a set of features that include those 60 features with some modifications. First, a feature should have a meaningful relationship to the transportation modes. Therefore, absolute values of the rotation vector sensor are excluded from the feature set because the absolute values correspond to the device's orientation and are unrelated to the transportation modes. Second, since a time window is being monitored, other features that can describe variations in time were created to incorporate the features' time dependency (e.g. based on derivatives). Also, spectral entropy was added, which can be used as a measure to show the peaky spots of a distribution [22]. Peaky spots are important since this measure can be different for different transportation modes. Intuitively, an abrupt braking (which in reflected in the accelerometer data) in a car mode is peakier than in the bike mode. A High value of spectral entropy for a distribution shows that the distribution is somewhat flat. Conversely, the spectral entropy decreases when the distribution becomes less flat [23]. In addition, the data from the sensors were treated as signals, consequently, the energy of the signal within the time window of interest was added to the feature set [24]. Also, the data from the GPS were excluded to only focus on the scenario where no GPS data are available. To summarize, using the data from different sensors and for each time window, TABLE I shows the measures that were used to create the feature set. Other than the "spectral entropy" and "energy" that was mentioned above, other measure include: $mean$ (or average), max (maximum), min (minimum), var (variance), std (standard deviation), $range$, iqr (interquartile range), and $signChange$ (number of times the sign of a feature changes over the time window). Also in this table, x_i^t represents the data array for the i^{th} feature (e.g. acceleration) from the time window t. Also, $\dot{x}_i^t$ represent the derivative of x_i^t. A total of 165 features were created: out of the 18 measures presented in TABLE I, all the 18 measures were applied to each of the sensor values; 7 measures were applied to rotation vector sensor values; 16 measures were applied to the summation values from accelerometer and gyroscope sensors (e.g. $m_{acceleration} = \sqrt{a_x{}^2 + a_y{}^2 + a_z{}^2}$); 4 measures were applied to the summation values from rotation vector sensor. As a result, the total number of features reached 18*6+7*3+16*2+4*1 = 165 features.

TABLE I
MEASURES USED TO CREATE FEATURES

No.	Measure	No.	Measure
1	$mean(x_i^t)$	10	$spectralEntropy(x_i^t)$
2	$max(x_i^t)$	11	$mean\left(\dot{x}_i^t\right)$
3	$min(x_i^t)$	12	$max\left(\dot{x}_i^t\right)$
4	$var(x_i^t)$	13	$min\left(\dot{x}_i^t\right)$
5	$std\ (x_i^t)$	14	$var\left(\dot{x}_i^t\right)$
6	$range(x_i^t)$	15	$std\left(\dot{x}_i^t\right)$
7	$iqr(x_i^t)$	16	$range\left(\dot{x}_i^t\right)$
8	$signChange(x_i^t)$	17	$iqr\left(\dot{x}_i^t\right)$
9	$energy(x_i^t)$	18	$signChange\left(\dot{x}_i^t\right)$

IV. MODEL DEVELOPMENT

Three methods were considered to construct the detection models. Support Vector Machine (SVM) and Decision Tree have been used in most of the literature to classify transportation modes, and some papers found the Decision Tree to be the best method. Consequently, tree-based models (single Decision Tree, Bagging, and Random Forest) and SVM were selected for model construction. In addition, the K-Nearest Neighbor (KNN) method was considered for the purpose of comparison given that it is a simple technique. Several methods were adopted in the model development process; maximum dependency minimum redundancy (mRMR) for feature selection, K-fold Cross- Validation for model selection, and Scaling for normalization. To conduct feature scaling, the feature values were normalized to be within the range of [-1, 1] (Scaling was conducted only when applying SVM and KNN).

A. K-Nearest Neighbor (KNN)

A simple yet effective method, namely the K-Nearest Neighbor (KNN), which has been applied to numerous classification and regression problems in different fields, was adopted to identify transportation modes. For each test observation (X_j^{test}) that includes different features (such as x_i^t), this method first identifies the K nearest train observations (X_j^{train}) in the training data set to the test observation and stores them in the N_K set. Taking the majority vote of the classes for the K nearest points identifies the class of the test observation. Calculating the average in the Equation (1) is equivalent to taking the majority vote in the case of classification (versus regression). y_j^{train} and y_j^{test} are the response (or target) values corresponding to the observations X_j^{train} and X_j^{test}, respectively. K is a tuning parameter that needs to be determined [25].

$$y_j^{test} = \frac{1}{K} \sum_{X_j^{train} \in N_K} y_j^{train} \tag{1}$$

B. Support Vector Machines (SVMs)

SVM is known as a large margin classifier, which means when classifying data, it determines the best possible decision boundary that provides the largest possible gap between classes. This characteristic contributes to a higher confidence in solving classification problems. To construct the SVM model, the following factors are taken into account: using a Gaussian kernel with complete model selection (which entails consideration of the regularization parameter and the Gaussian parameter), and applying feature scaling.

Equation (2) presents the SVM formulation to solve the classification problem and the associated constraints are shown in Equations (3) and (4) [26]. The objective function is composed of two terms: minimizing the first term is basically equivalent to maximizing the margin between classes, and the second term consists of an error term multiplied by the regularization (penalty) parameter denoted by C. The C parameter should be determined to provide the relative importance between the two terms. Equation (3) ensures that margin of at least 1 exists with consideration of some violations. The value of 1 was resulted from normalizing w. Equation (4) restricts the data points to the points that have positive errors.

$$\min_{w,b,\xi} \left(\frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n \right) \tag{2}$$

Subject to:

$$y_n \left(w^T \phi(x_n) + b \right) \geq 1 - \xi_n \, , n = 1, \dots, N \tag{3}$$

$$\xi_n \geq 0 \, , n = 1, \dots, N \tag{4}$$

Where,

w	Parameters to define decision boundary between classes
C	Regularization (or penalty) parameter
ξ_n	Error parameter to denote margin violation
b	Intercept associated with decision boundaries
$\phi(x_n)$	Function to transform data from X space into some Z space
y_n	Target value for the n^{th} observation

SVM applies the function $\phi(.)$ to transform data from the current n-dimensional X space into a higher dimensional Z space in which the decision boundaries between classes are easier to identify. This transformation could be computationally very expensive; consequently, to solve the problem, the SVM only needs to obtain vector inner products in the space of interest. Hence, SVM takes advantage of some functions known as Kernels that return the vector inner product in the desired Z space. Different types of kernels exist such as linear kernel, polynomial kernels, and Gaussian kernel. Linear kernel, as applied in [7, 10], is the basic mode which means no kernels are actually taken into account. In other words, vector inner product as appears in the dual formulation of the problem is considered without transforming data into another space. For a certain type of problems, SVM can produce better results with more advanced kernels such as Gaussian kernel. According to our data size and number of features, Gaussian kernel was believed to be the most appropriate kernel [27]. Also, if a complete model selection is carried out, there is no need to test the linear kernel because the results obtained from the Gaussian kernel include the results obtained from the linear kernel [28]. In this paper, the $\phi(x_n)$ function corresponds to the Gaussian kernel. The formulation of the Gaussian kernel is shown in Equation (5). When using this type of kernel, the tuning parameters are the Gaussian parameter (σ) and the regularization parameter (C) that should be determined to obtain the best possible detection performance.

$$K(x, x') = exp\left(-\frac{\|x - x'\|^2}{2\sigma^2} \right) \tag{5}$$

Where,

x, x'	n-dimensional vectors
$\|x - x'\|$	Euclidean distance between vectors x, x'
σ	Gaussian parameter

Two approaches were examined: (1) Developing a single SVM model using the entire dataset, (2) Developing an ensemble of SVM models using a smaller data set for each model. Similar to the idea behind the Bag approach, instead of developing a single SVM model, a series of SVM models was developed and the final result was determined based on the majority vote obtained from the SVM models. A number of studies have considered taking advantage of an ensemble of SVMs [29, 30]. As the number of observations in the training dataset (n) increases the training time increases with the power of two (n^2) [31]. Thus, if the data set is sufficiently large developing a number of SVM models using a subset of data can be faster than developing a single SVM model using the entire data.

C. Tree-based models

1) Decision Tree

Decision Trees were introduced for classification and regression problems in the mid-80s [32]. These approaches have several advantages; among all, they are easy to explain and interpret, they reflect the human decision making process, they can be graphically displayed, and there is no need to create dummy variables for qualitative predictors. However, as the tree becomes larger, it may over-fit the data and show poor performance on the test data set. Consequently, some strategies are used in the R and CART software to construct a large tree using recursive binary splitting and then pruning back to obtain a good sub-tree. This approach is known as Cost Complexity Pruning or Weakest Link Pruning. In the Recursive binary splitting method, a root node is the starting point where a predictor (feature) needs to be selected with a cut point to split the data into two parts or nodes. This procedure of selecting a feature and splitting is carried out successively to grow the tree. Different criteria can be used to choose the best split at each node, including: classification error rate, Gini index, and Cross-Entropy. In practice the two latter methods result in better performance. Consequently, Cross-Entropy was used in this study. Having K classes, at each node m which receives N^m observations (x_i^m, y_i^m) from its parent node, Cross-Entropy can be obtained through Equation (6) [33].

$$\sum_{k=1}^{K} \boldsymbol{P_k^m \, log \, P_k^m} \tag{6}$$

where,

P_k^m	Proportion of class k observations in node m
$P_k^m =$	$\frac{1}{N^m} \sum_{x_i^m} I(y_i^m = k)$
y_i^m	Target value of n^{th} observation in node m
$I(y_i^m = k)$	1 if $y_i^m = k$, and 0 otherwise

2) Bagging

Bagging or Bootstrap aggregating method, introduced in 1996 [34], takes advantage of aggregating results from different models to reduce the variance. The detection/prediction results of different models constructed on different training sets can be averaged. However, in practice, we usually have only one training set. Instead, bootstrapped training data (Pseudo training sets) can be obtained by taking repeated samples from a single training set [35] and a tree model can be constructed for each. Afterwards, the average performance of all models represents the overall performance, which is called Bagging or Bootstrap aggregation. There is no need for pruning of trees as the variance is reduced by averaging. Averaging is equivalent to taking a majority vote for classification problems, which is the case in the present study. The detection/prediction for a single data point $\boldsymbol{x}$ is obtained by averaging (taking a majority vote) the detections resulted from all bootstrapped samples as shown in Equation (7). The trees can be as large as possible, thus the only parameter to be determined is the number of trees.

$$\boldsymbol{\hat{y}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b(x)} \tag{7}$$

where,

$\boldsymbol{\hat{y}_{bag}(x)}$	Target value resulted from averaging
$\boldsymbol{\hat{y}_b(x)}$	Detected target value for observation $\boldsymbol{x}$ in bootstrap sample $\boldsymbol{b}$
$\boldsymbol{B}$	Total bootstrap samples

3) Random Forest

Similar to the Bagging method, the random forest method, as proposed in 2001 [36], creates an ensemble of trees and the result is obtained based on the majority votes. One issue relating to the Bagging is that the trees can be very similar since all the features are used to construct each tree; consequently, the trees can be highly correlated. To tackle this problem random forest restricts the number of features by randomly selecting a subset of features to grow each tree. The parameters to be determined are the number of features to use and the number of trees. Interestingly, in Random Forest and also Bagging approaches, adding more trees does not lead to over-fitting, but at some point not much benefit is gained by including more trees [33].

D. Feature Selection

Feature selection is considered to be a critical task as it can reduce the dimensionality of the problem, reduce the noise, identify more important predictors, and lead to more interpretable features [37].

Some methods have been applied to select the most relevant features to use such as ANOVA tests used in [16], correlation based feature selection (CFS) used in [7], and Chi Squared and Information gain methods applied in [4]. Using mutual information or some statistical tests to select the top-ranked features may not be sufficient as the selected features could be highly correlated among themselves [37, 38]. In other words, not much benefit is gained by combining highly correlated features. In the present study, the selection of the most representative features entailed using the minimum redundancy maximum relevance (mRMR) approach. This approach was used to deal with this issue; when selecting the

x_i feature from the feature set M, assuming set F has the already selected features, the goal is to simultaneously maximize the relevance between the feature and the target class (i.e. x_i, c) as shown in Equation (8) and to minimize the redundancy between that feature and the already selected features (i.e. x_i, x_j) as shown in Equation (9) [37]. Hence, using mRMR all the features that were created were ranked to choose the most useful ones; the top 80 features were selected out of the entire 165 features. The number 80 was chosen by experimenting different values. In other words, it was desired to achieve a good level of detection accuracy and at the same time to exclude less useful features.

$$\max_{x_i \in (M-F)} MI(x_i, c) \tag{8}$$

$$\min_{x_i \in (M-F)} \frac{1}{|F|} \sum_{x_j \in F} MI(x_i, x_j) \tag{9}$$

where,

$MI(x, y)$	Mutual Information of x and y
x_i	The feature to be examined
x_j	A previously selected feature
M	Set of all features
F	Set of the selected features
c	Target class

E. K-fold Cross-Validation

The K-fold cross-validation is a powerful technique to estimate the detection/prediction error. Consequently, it is used to select the best model and to determine the model parameters. The idea is to randomly divide the data of n observations into K approximately equal parts or folds $(F_{1,} F_2, \dots, F_K)$ with n_k observations in each fold. Subsequently, the data of the first fold are set aside as the validation data set, and the data of the remaining $K-1$ folds are used as the training data set to construct a model. The same procedure is conducted K times, each time a different validation data set is chosen. The performance of each model is evaluated on the corresponding validation set and the average detection error is obtained over the K models as shown in Equation (10). The special case is when the number of folds is exactly the same as the number of observations; this is called Leave-One-Out Cross-Validation (LOOCV). LOOCV requires high computations as it needs to construct the model n times which in this case is the number of observations. Also, since only one observation is left out at each k stage, the training sets are almost the same for each model and thus the estimates are highly correlated, consequently, the average over K folds can have high variance. In practice, the best choice for the number of folds is 5 or 10 [39, 40].

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} Err_k \tag{10}$$

Where,

$$Err_k = \sum_{i \in F_k} I(y_i \neq \hat{y}_i)/n_k$$

n	Number of observations
n_k	Number of observations in k^{th} fold
F_k	Set of observations in k^{th} fold
y_i	Actual target value
$\hat{y}_i$	Detected target value
$I(y_i \neq \hat{y}_i)$	1 if $y_i \neq \hat{y}_i$, and 0 otherwise

V. Results

Using mRMR, 80 features were selected as the most relevant features, which were used to construct the models. The performance of each model was quantified using different metrics depending on the model, namely: misclassification error obtained from 5-fold Cross-Validation and Out-Of-Bag error. Cross-Validation, as mentioned earlier, is a good technique to estimate the detection/prediction errors. Out-Of-Bag error is an accurate estimate of the errors suitable for Bagging and Random Forest that is almost identical to the Cross-Validation accuracy [33]. Moreover, in developing different models, 30% of the data were set aside to obtain a test error, and the remaining 70% were used as the training set for model development. Subsequently, Confusion matrices were obtained for each model, which shows the classification rates, misclassification rates, recall, and precision values. The recall measure is calculated by dividing the total number of true positives by the total number of actual positives. The Precision measure is computed by dividing the total number of true positives by the total number of predicted positives. Finally the models were compared to each other using different performance measures such as F-Score, Youden's index, and the discriminant power that will be presented in the "model comparison" section.

A. KNN Model

The KNN model using 5-fold Cross-Validation was implemented using the R software [41] and the class package [42]. The only tuning parameter in the KNN method is the number of K neighbors. Fig. 1 shows the misclassification error obtained from a 5-fold Cross-Validation (applied in the training set) of 25 runs for different numbers of neighbors. The highest accuracy was achieved when K was 7 resulting in an error rate of 8.8% (accuracy of 91.2%). The confusion matrix of the KNN model including the Recall and Precision values is shown in Table II.

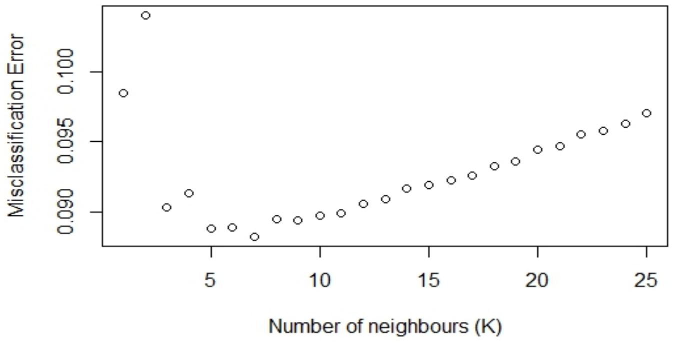

Fig. 1. Impact of number of neighbors on model misclassification error

TABLE II
CONFUSION MATRIX - KNN MODEL

KNN		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	93.31	2.33	2.30	0.99	2.96	91.72
	Car	2.73	84.07	0.77	0.38	11.74	84.32
	Walk	2.37	0.20	96.51	1.50	0.25	95.72
	Run	0.07	0.00	0.15	97.00	0.05	99.71
	Bus	1.51	13.40	0.27	0.13	85.01	84.69
	Recall	93.31	84.07	96.51	97.00	85.01	

B. SVM Model

In implementing the SVM, the LibSVM library of SVMs was used. For multiclass classification, considering h classes, LibSVM applies one-against-one method in which $h(h-1)/2$ binary models are built. Among these, LibSVM chooses the parameters that achieve the highest overall performance. Another well-known method is called one-against-all which is more intuitive and has similar performance. However, LibSVM takes advantage of one-against-one because of its shorter training time. [43]. Furthermore, 5-fold Cross-Validation was used for model development and assessment.

1) Single SVM

The 5-fold Cross-Validation was applied on the training set to develop a single SVM model. In order to conduct a complete model selection, the regularization parameter (c) as well as the Gaussian parameter (σ) should be optimized. The Gaussian kernel formulation used in libSVM [43] is slightly different from Equation (5); in their formulation, the parameter $gamma$ was used instead of $\frac{1}{2\sigma^2}$. Fig. 2 presents a contour plot that illustrates how different values of the regularization (c) and the Gaussian $(gamma)$ parameters impact the performance of the SVM model. The optimal values for $(gamma, c)$ were found to be (2.828, 512) that led to the overall accuracy of 94.62%. Parameters c and σ (or $gamma$) deals with the issues of over-fitting and under-fitting which is a bias-variance tradeoff. Detailed information regarding the bias-variance tradeoff can be found in [29]. TABLE III presents the confusion matrix for the SVM model.

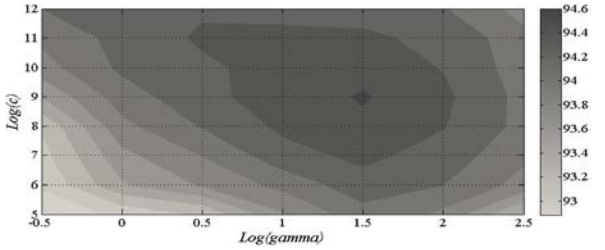

Fig. 2. Impacts of regularization and Gaussian parameters on model Accuracy

TABLE III
CONFUSION MATRIX - SVM MODEL

SVM		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	95.11	0.89	1.69	0.45	0.94	96.28
	Car	0.96	93.58	0.18	0.15	6.90	91.74
	Walk	1.89	0.28	97.11	1.34	0.55	95.79
	Run	0.37	0.15	0.77	97.55	0.17	98.50
	Bus	1.66	5.10	0.26	0.50	91.44	92.42
	Recall	95.11	93.58	97.11	97.55	91.44	

2) Ensemble of SVMs (E.SVM)

For the ensemble of SVM models, the 5-fold cross validation was adopted in a slightly different fashion: the training set (70% part) was divided into 5 folds; one fold was set aside as the validation set, and about 25% of the remaining 4 folds were used to train the first SVM model. Similarly, 25% of the 4 folds was sampled (bootstrap sample; sampling with replacement) to train the second SVM model. The procedure continues until 200 models were constructed. In addition, trial and error was used to pick model parameters $(gamma, c)$ for each SVM model. The average of these 200 models was validated with the data fold that was set aside. All these steps were carried out 5 times, each time with a different data fold as the validation set. Averaging the results of the five folds represented the cross validation results of the ensemble of SVM models. This method led to an overall accuracy of 94.41%. The confusion matrix corresponding for this approach is shown in TABLE IV.

TABLE IV
CONFUSION MATRIX – ENSEMBLE OF SVM MODELS

Ensemble of SVMs		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	95.63	0.68	1.71	0.56	1.08	96.07
	Car	0.83	91.72	0.13	0.15	7.75	91.04
	Walk	1.60	0.48	97.16	1.27	0.74	95.91
	Run	0.46	0.30	0.75	97.82	0.32	98.12
	Bus	1.48	6.83	0.25	0.20	90.11	91.25
	Recall	95.63	91.72	97.16	97.82	90.11	

C. Tree-based models

1) Decision Tree (DT)

The decision tree method was implemented in the R software along with two packages ("tree" and "maptree" packages) for tree analysis [41, 44, 45]. The resultant single tree was a very large tree with 48 terminal nodes with an overall accuracy of 87.27%. Table V shows the confusion matrix of the decision tree model.

TABLE V
CONFUSION MATRIX - DECISION TREE MODEL

Decision Tree		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	85.32	1.78	5.21	0.85	3.07	88.96
	Car	1.39	79.30	0.26	0.14	12.14	85.03
	Walk	8.65	0.10	91.99	2.87	0.13	88.54
	Run	0.40	0.00	1.17	95.30	0.00	98.32
	Bus	4.24	18.83	1.36	0.85	84.65	76.92
	Recall	85.32	79.30	91.99	95.30	84.65	

Since the tree is very large, Cost Complexity Pruning was applied to prune the tree from 48 terminal nodes to 24 without much loss in performance. Table VI shows the confusion matrix of the pruned tree resulted in 86.3% overall accuracy. This model was called DT.P to abbreviate the model title of the pruned decision tree.

Because the tree was big to illustrate, the tree was pruned again to reduce the number of terminal nodes to 9 just for illustration purposes, as illustrated in Fig. 3. In this case the accuracy of the model is 82.1%.

TABLE VI
CONFUSION MATRIX - PRUNED DECISION TREE MODEL

Decision Tree - Pruned		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	84.37	3.12	5.87	1.38	4.02	85.79
	Car	0.61	78.50	0.33	0.05	12.77	85.07
	Walk	9.80	0.07	90.44	2.85	0.10	87.42
	Run	0.53	0.00	2.30	95.12	0.00	97.04
	Bus	4.69	18.31	1.06	0.59	83.11	77.02
	Recall	84.37	78.50	90.44	95.12	83.11	

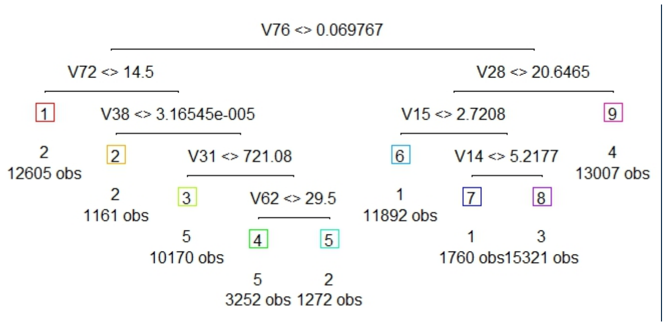

Fig. 3. Illustration of a single Decision Tree

2) *Bagging (Bag) and Random Forest (RF) models*

In implementing the Bagging and Random Forest methods, the R software and the package "RandomForest" were used, respectively [41, 46]. These two methods were examined together since the Bagging is in fact a special case of a random forest when the number of selected features equals the total number of features. As mentioned before, adding more trees will not cause over-fitting. However, a sufficient number of trees are needed. Fig. 4 shows a series of Random Forest models with different number of trees, from 1 to 500, using 5 features for each tree. After approximately 200 trees, no benefit is gained by including more trees. Thus, to apply these approaches, 400 trees were used, which is a sufficiently large number. On the far left of the diagram, when the number of trees is 1, it is equivalent to having a single decision tree.

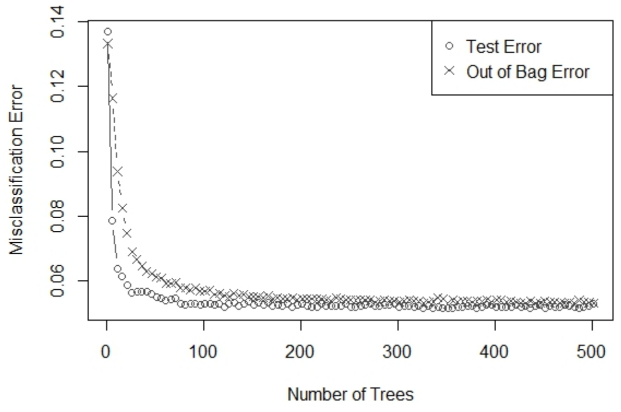

Fig. 4. Impact of number of trees on the misclassification error

For the random forest method, other than the number of trees, the number of features needs to be determined as well. Fig. 5 shows a series of random forest models with different number of features for each tree. Since a total of 80 features were used (as identified by mRMR), a total of 80 random forest models were constructed to find the best number of features to use. The far right of the figure shows the results of the Bagging approach, where all the 80 features were used. The minimum error rate was obtained with 12-25 features in use. The model with 12 features was selected as the best model since a less complex model with less features is always more disirable. The Confusion matrix for the bagging and the best random forest models are shown in Table VII and Table VIII. The overal accuracy of the best random forest model and the bagging model, obtained from 5-fold Cross-Validation, were 95.1% and 94.4% respectively.

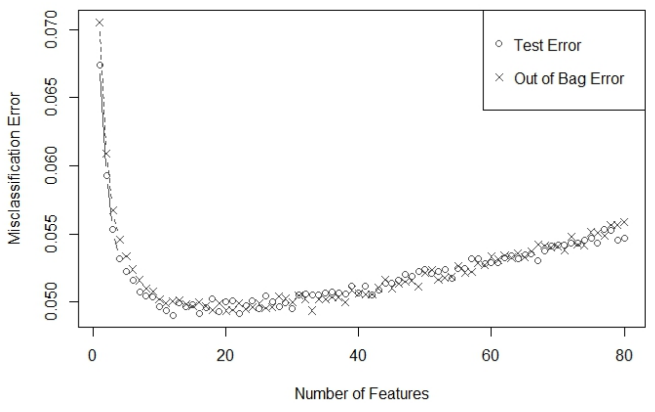

Fig. 5. Impact of number of features on the misclassification error

TABLE VII
CONFUSION MATRIX - RANDOM FOREST

Random Forest		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	95.47	1.46	2.63	0.97	2.29	93.06
	Car	0.37	93.84	0.12	0.05	4.47	94.93
	Walk	2.93	0.13	96.23	1.59	0.12	95.24
	Run	0.03	0.00	0.40	96.81	0.00	99.55
	Bus	1.19	4.57	0.63	0.58	93.12	93.02
	Recall	95.47	93.84	96.23	96.81	93.12	

TABLE VIII
CONFUSION MATRIX - BAGGING

Bagging		Actual					
		Bike	Car	Walk	Run	Bus	**Precision**
Predicted	Bike	94.63	1.48	2.81	1.00	2.34	92.75
	Car	0.48	92.64	0.12	0.03	5.08	94.18
	Walk	3.43	0.13	95.95	1.63	0.22	94.62
	Run	0.03	0.00	0.58	96.79	0.02	99.34
	Bus	1.42	5.74	0.55	0.54	92.34	91.76
	Recall	94.63	92.64	95.95	96.79	92.34	

3) *Feature Importance*

As was mentioned earlier, a total of 80 features were identified as the most relevant features by mRMR method. Fig. 6 shows the actual importance of the best 20 features associated to the best random forest model. The importance of the features were assessed based on two measures: (1) Mean Decrease Accuracy that shows how the detection accuracy is decreased if a feature was excluded, averaged over all trees, and normalized by the standard deviation of the differences in accuracy and (2) Mean Decrease Gini that shows how a single feature contributed to decrease the Gini index over all the trees. Table IX shows the feature names in the order of importance. Since the two measures determine the feature importance in different ways the identified features by the two measures are different. While both measures have been used in the literature, there have been arguments concerning the preference for one measure over another. It is recommended that the first method (i.e. Mean Decrease Accuracy) is more suitable for causal interpretations. More details about the arguments and some contradictions regarding these measures can be found in [47].

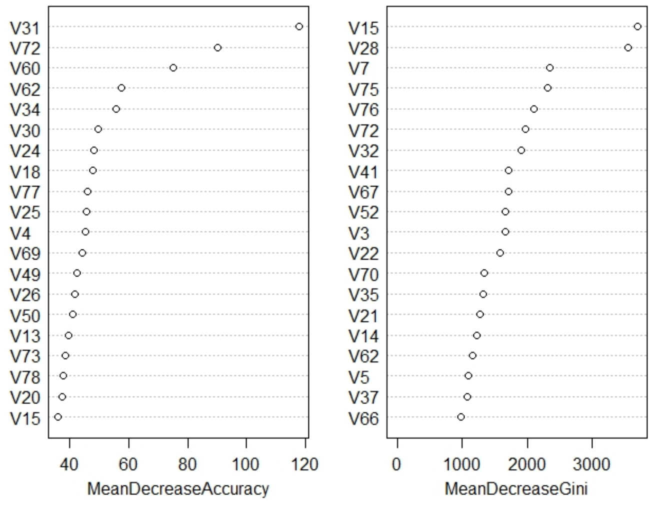

Fig. 6. Feature importance for two different measures

TABLE IX
IMPORTANT FEATURES

No.	Feature Name	No.	Feature Name
1	$spectralEntropy(a_x)$	11	$mean(a_z)$
2	$range(a_y)$	12	$iqr(\dot{a}_x)$
3	$max(a_y)$	13	$var(\dot{g}_x)$
4	$max(g_y)$	14	$min(\dot{a}_y)$
5	$min(g_y)$	15	$range(\dot{a}_x)$
6	$range(\dot{g}_x)$	16	$energy(a_x)$
7	$spectralEntropy(a_y)$	17	$range(g_x)$
8	$max(a_z)$	18	$mean(g_z)$
9	$mean(\dot{g}_x)$	19	$std(\dot{a}_y)$
10	$min(\dot{a}_z)$	20	$spectralEntropy(g_x)$

D. Model Comparison

The performance of the models was evaluated using four metrics, namely: the overall accuracy, the F-Score, Youden's index, and the Discriminant Power (DP). The overall accuracy is calculated by dividing the total number of correct detections by the total number of test data. The F-Score is a combined measure of the Recall and the Precision. The Youden's index is a measure to assess the ability of a model to avoid failure. The discriminant power shows how well a model discriminates between different classes by summarizing sensitivity and specificity of the model; the model is a poor discriminant if DP <1, limited if DP <2, fair if DP <3, good – in other cases. The sensitivity and specificity assess model performance on a single class, and are equivalent to the recall. By definition, assuming two classes (positive and negative) sensitivity is exactly the same as the Recall measure. Specificity is also the same metric but for the negative class. Fig. 7 illustrates a visual comparison between the models using different performance measures.

E. Feature Combination

An effort was made to develop a new additional feature which is a combination of other features. This was carried out by combining two approaches; a Meta heuristic approach called Simulated Annealing (SA) [48] and the Random Forest techniques. The new feature was created by multiplying other features; SA was adopted to select the best features to combine. The steps of this approach are as follows:

1. Define an initial solution: two random features were selected and placed in a set called *CF*. These features were combined (by multiplying by each other) to create a new feature. Subsequently, a RF model was developed using the previously used features (i.e. 80 features) and the newly defined feature to obtain the error rate for this initial solution.

2. Choose the algorithm's settings: trial and error was carried out to determine these algorithm parameters.

- ➢ Initial temperature ($t_{initial}$); (The term "temperature" is basically a control parameter which affects the probability of accepting or rejecting new solutions.)
- ➢ Final temperature (t_{final}) and stopping criteria
- ➢ Number of iterations at each temperature (M_k)
- ➢ Cooling schedule

After several trials, 5, 0.1, 15, and 0.8 were chosen as the $t_{initial}$, t_{final}, M_k, and the temperature reduction multiplier, respectively.

3. Repeat until stopping criteria are met

- ➢ Repeat until n=M_k, (n is a counter, starting from 0)
  - Generate a new solution: the new solution is generated by either randomly removing an already selected feature in the CF set or randomly adding a new feature to the CF set. Thereafter, the new feature is updated by multiplying all the features in the CF set.
  - Calculate Δ, the relative difference between the new and current error rates
  - If Δ≤0, the new solution is accepted, otherwise, the new solution can still be accepted with the probability of $e^{-\left(\Delta/_{\text{temperature}}\right)}$
  - *n = n+1*
- ➢ Decrease the temperature according to the cooling schedule: the temperature was decreased by multiplying the temperature value to 0.8 at each stage. Each stage corresponds to M_k iterations.

The error rate obtained by this approach was 4.7% which shows a very small improvement comparing to the results that was previously obtained by the RF model (i.e. 4.9%). The results showed that combining different features did not enhance the RF model significantly. The error could be attributed to having very similar data for different modes; cars, buses, and bicycles waiting at a traffic light; a traveler collecting the run mode may have stopped just a bit to catch their breath or stopped at a traffic light or a stop sign, which would be similar to the walk mode; a bus and a car travelling on the same road with very similar kinetic variables such as speed and acceleration.

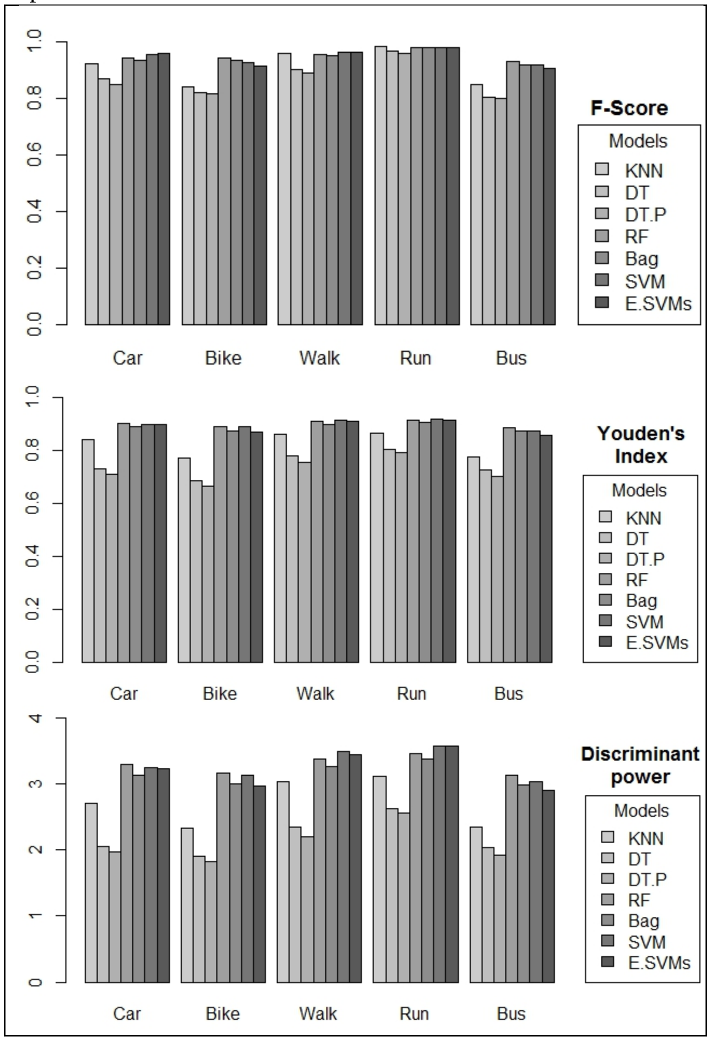

Fig. 7. Model comparison results

VI. Conclusions

Different classifiers were developed using machine learning techniques to identify different transportation modes including bike, car, walk, run, and bus. In training and testing the classifier, data were obtained from smartphone sensors such as accelerometer, gyroscope, and rotation vector which were found to have important information for the purpose of mode recognition. A time window of one second was chosen, so the model can fit in a broader range of applications. For each method, parameters that needed to be optimized were examined to conduct a complete model selection. K-fold Cross-Validation and Out-Of-Bag error were used for model evaluations. In addition, some performance measures such as the F-Score, Youden's index, and Discriminant Power were applied to assess model performances on the individual modes. Considering misclassification rates, the car and bus modes were the most difficult ones to distinguish, as would be expected. Even using more complex models such as SVM and RF, the car mode was misclassified as the bus mode in about 4-6% of the time. The Random Forest method was found to produce the best overall performance. However, for specific modes (i.e. walk and Run), the SVM outperformed the RF method. Several features were created and examined; among which 80 features were identified using the mRMR method as the most relevant feature. Other than some statistical measures of dispersion (e.g. range, max, variance, etc.), spectral entropy and energy were among the most important features.

Feature combination was investigated by employing SA algorithm and the RF method in an iterative mechanism. However, no significant improvement was obtained. Due to collecting data in a naturalistic way, inevitable situations with very similar data might have occurred, which can be the cause of the error.

Some recommendations for future directions that applies to this and similar research problems include: adding more data, applying approaches to examine the data as a sequence, considering more transportation modes (e.g. metro), and conducting error analysis to gain some insights about where different models fail to correctly classify the data and consequently incorporate that knowledge into the models to enhance the detection performance.

Acknowledgments

This research was co-funded by the Mid-Atlantic University Transportation Center (MAUTC), the Connected Vehicle Initiative UTC (CVI-UTC), and the Tran*LIVE* UTC.

References

[1] Bao, L. and S.S. Intille, *Activity recognition from user-annotated acceleration data*, in *Pervasive Computing, Proceedings*, A. Ferscha and F. Mattern, Editors. 2004. p. 1-17.

[2] Kwapisz, J.R., G.M. Weiss, and S.A. Moore, *Activity recognition using cell phone accelerometers.* SIGKDD Explor. Newsl., 2011. **12**(2): p. 74-82.

[3] Susi, M., V. Renaudin, and G. Lachapelle, *Motion Mode Recognition and Step Detection Algorithms for Mobile Phone Users.* Sensors, 2013. **13**(2): p. 1539-62.

[4] Stenneth, L., et al. *Transportation mode detection using mobile phones and GIS information*. in *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM SIGSPATIAL GIS 2011, November 1, 2011 - November 4, 2011*. 2011. Chicago, IL, United states: Association for Computing Machinery.

[5] Yu, X., et al. *Transportation activity analysis using smartphones*. in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*. 2012.

[6] Widhalm, P., P. Nitsche, and N. Brandie. *Transport mode detection with realistic Smartphone sensor data*. in *2012 21st International Conference on Pattern Recognition (ICPR 2012), 11-15 Nov. 2012*. 2012. Piscataway, NJ, USA: IEEE.

[7] Reddy, S., et al., *Using Mobile Phones to Determine Transportation Modes.* Acm Transactions on Sensor Networks, 2010. **6**(2).

[8] Manzoni, V., et al., *Transportation mode identification and real-time CO2 emission estimation using smartphones*. 2010, Technical report, Massachusetts Institute of Technology, Cambridge.

[9] Biljecki, F., H. Ledoux, and P. van Oosterom, *Transportation mode-based segmentation and classification of movement trajectories.* International Journal of Geographical Information Science, 2013. **27**(2): p. 385-407.

[10] Zheng, Y., et al., *Learning transportation mode from raw gps data for geographic applications on the web*, in *Proceedings of the 17th international conference on World Wide Web*. 2008, ACM: Beijing, China. p. 247-256.

[11] Gonzalez, P.A., et al., *Automating mode detection for travel behaviour analysis by using global positioning systems-enabled mobile phones and neural networks.* Iet Intelligent Transport Systems, 2010. **4**(1): p. 37-49.

[12] Byon, Y.J., B. Abdulhai, and A. Shalaby, *Real-Time Transportation Mode Detection via Tracking Global Positioning System Mobile Devices.* Journal of Intelligent Transportation Systems, 2009. **13**(4): p. 161-170.

[13] Zhang, L., M. Qiang, and G. Yang, *Mobility transportation mode detection based on trajectory segment.* Journal of Computational Information Systems, 2013. **9**(8): p. 3279-3286.

[14] Nham, B., K. Siangliulue, and S. Yeung, *Predicting mode of transport from iphone accelerometer data*. 2008, Tech. report, Stanford Univ.

[15] Nick, T., et al. *Classifying means of transportation using mobile sensor data*. in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. 2010. IEEE.

[16] Bolbol, A., et al., *Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification.* Computers, Environment and Urban Systems, 2012.

[17] Zheng, Y., et al., *Understanding transportation modes based on GPS data for web applications.* ACM Transactions on the Web (TWEB), 2010. **4**(1): p. 1.

[18] Nitsche, P., et al., *A strategy on how to utilize smartphones for automatically reconstructing trips in travel surveys.* Procedia-Social and Behavioral Sciences, 2012. **48**: p. 1033-1046.

[19] Gong, H., et al., *A GPS/GIS method for travel mode detection in New York City.* Computers, Environment and Urban Systems, 2012. **36**(2): p. 131-139.

[20] Lester, J., et al., *MobileSense-Sensing modes of transportation in studies of the built environment.* UrbanSense08, 2008: p. 46-50.

[21] Jahangiri, A. and H. Rakha. *Developing a Support Vector Machine (SVM) Classifier for Transportation Mode Identification by Using Mobile Phone Sensor Data*. in *Transportation Research Board 93rd Annual Meeting*. 2014.

[22] Misra, H., et al. *Spectral entropy based feature for robust ASR*. in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. 2004. IEEE.

[23] Lu, H., et al. *The Jigsaw continuous sensing engine for mobile phone applications*. in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. 2010. ACM.

[24] Shmaliy, Y., *Continuous-time signals*. 2006: Springer.

[25] Friedman, J.H., F. Baskett, and L.J. Shustek, *An algorithm for finding nearest neighbors.* IEEE Transactions on computers, 1975. **24**(10): p. 1000-1006.

[26] Hsu, C.-W. and C.-J. Lin, *A comparison of methods for multiclass support vector machines.* Neural Networks, IEEE Transactions on, 2002. **13**(2): p. 415-425.

[27] Hsu, C.-W., C.-C. Chang, and C.-J. Lin, *A practical guide to support vector classification*. 2003.

[28] Keerthi, S.S. and C.-J. Lin, *Asymptotic behaviors of support vector machines with Gaussian kernel.* Neural computation, 2003. **15**(7): p. 1667-1689.

[29] Valentini, G. and T.G. Dietterich, *Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods.* The Journal of Machine Learning Research, 2004. **5**: p. 725-775.

[30] Kim, H.-C., et al., *Constructing support vector machine ensemble.* Pattern recognition, 2003. **36**(12): p. 2757-2767.

[31] Claesen, M., et al., *EnsembleSVM: A Library for Ensemble Learning Using Support Vector Machines.* Journal of Machine Learning Research, 2013(accepted).

[32] Breiman, L., et al., *Classification and regression trees*. 1984: CRC press.

[33] Hastie, T., et al., *The elements of statistical learning*. Vol. 2. 2009: Springer.

[34] Breiman, L., *Bagging predictors.* Machine learning, 1996. **24**(2): p. 123-140.

[35] Efron, B. and R.J. Tibshirani, *An introduction to the bootstrap*. Vol. 57. 1994: CRC press.

[36] Breiman, L., *Random forests.* Machine learning, 2001. **45**(1): p. 5-32.

[37] Ding, C. and H. Peng, *Minimum redundancy feature selection from microarray gene expression data.* Journal of bioinformatics and computational biology, 2005. **3**(02): p. 185-205.

[38] Peng, H., F. Long, and C. Ding, *Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2005. **27**(8): p. 1226-1238.

[39] Kohavi, R. *A study of cross-validation and bootstrap for accuracy estimation and model selection*. in *IJCAI*. 1995.

[40] James, G., et al., *An introduction to statistical learning*. 2013: Springer.

[41] R Core Team, *R: A Language and Environment for Statistical Computing*. 2014, R Foundation for Statistical Computing.

[42] Venables, W.N. and B.D. Ripley, *Modern applied statistics with S*. 2002: Springer.

[43] Chang, C.-C. and C.-J. Lin, *LIBSVM: a library for support vector machines.* ACM Transactions on Intelligent Systems and Technology (TIST), 2011. **2**(3): p. 27.

[44] Ripley, B. *tree: Classification and regression trees*. 2014.

[45] Gramacy, D.W.a.R.B., *maptree: Mapping, pruning, and graphing tree models*. 2012.

[46] Liaw, A. and M. Wiener, *Classification and Regression by randomForest.* R news, 2002. **2**(3): p. 18-22.

[47] Neville, P.G., *Controversy of Variable Importance in Random Forests.* Journal of Unified Statistical Techniques, 2013. **1**(1).

[48] Glover, F. and G.A. Kochenberger, *Handbook of metaheuristics*. 2003: Springer.

Arash Jahangiri received the M.Sc. in civil and environmental engineering from Virginia Tech, Blacksburg, in 2012. He is currently a PhD candidate with the Charles E. Via, Jr. Department of Civil and Environmental Engineering at Virginia Tech. As a Graduate Research Assistant, he is currently with the Center for Sustainable Mobility (CSM) at Virginia Tech Transportation Institute (VTTI). His research interest comprises Intelligent Transportation Systems, Traffic Safety, Environmental impacts of Transportation, Artificial Intelligence, and Traffic Flow Theory.

Hesham A. Rakha (M'04) received the B.Sc. degree (with honors) in civil engineering from Cairo University, Cairo, Egypt, in 1987 and the M.Sc. and Ph.D. degrees in civil and environmental engineering from Queen's University, Kingston, ON, Canada, in 1990 and 1993, respectively. He is currently the Samuel Reynolds Pritchard Professor of Engineering with the Charles E. Via, Jr. Department of Civil and Environmental Engineering, a Courtesy Professor with the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, and the Director of the Center for Sustainable Mobility at the Virginia Tech Transportation Institute. He has authored/coauthored 4 books, more than 300 refereed publications in the areas of traffic flow theory, traffic modeling and simulation, traveler and driver behavior modeling, artificial intelligence, dynamic traffic assignment, traffic control, energy and environmental modeling, and safety modeling. Dr. Rakha in addition to being a member of IEEE is a member of the ITE, the ASCE, and the Transportation Research Board (TRB). He is a Professional Engineer in Ontario, Canada.