Using Smartphone-based Accelerometer to Detect Travel by Metro Train

by
Megha Vij

Under the supervision of: Dr. Vinayak Naik
Dr. Venkata M.V. Gunturi

Indraprastha Institute of Information Technology, Delhi
May, 2016

Using Smartphone-based Accelerometer to Detect Travel by Metro Train

by
Megha Vij

Submitted
in partial fulfilment of the requirements for the degree of
Master of Technology

to

Indraprastha Institute of Information Technology, Delhi
May, 2016

Certificate

This is to certify that the thesis titled "Using Smartphone-based Accelerometer to Detect Travel by Metro Train" being submitted by Megha Vij to the Indraprastha Institute of Information Technolody, Delhi, for the award of the Master of Technology,is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirementsof the regulations relating to the degree.
The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

May, 2016

Dr. Vinayak Naik
Department of Mobile & Ubiquitous Computing
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Dr. Venkata M.V. Gunturi
Department of Data Engineering
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Abstract

We look at the problem of using accelerometer in smartphones to detect whether the user is at a metro train station or in a metro train. Currently, we have solutions to detect simple activities, such as sitting or walking. Our work for this thesis investigates the more complex problem of discerning "in-train" from "in-metro-station" activities which internally are composed of several simple activities. We perform the task of distinguishing the "in-train" from "in-metro-station" patterns using classic classification techniques with two different data representations namely, statistical features and ECDF-based features. Another major contribution through this thesis is to solve the challenge of the considerable class imbalance with majority of samples belonging to the "in-train" patterns by improvising existing classification algorithms to counter the effect of class imbalance. Our findings are useful for any other problem of using sensor data to classify activities. We evaluated our solution using about 23 hours of data collected using six different models of smartphones from over seven different metro/subway stations situated in New Delhi, India. Our detection accuracy is over 98%.

Acknowledgements

I would like to express my sincere gratitude towards my advisors, Dr. Vinayak Naik and Dr. Venkata M.V. Gunturi for guiding me through the duration of my thesis. Their knowledge and expert advice helped me in understanding the different facets of my research work. Their valuable inputs helped me build on the approach to carry forward my thesis in the right direction.
I would also like to thank my parents for motivating from time to time throughout the course of my thesis which enabled me to pursue my research in an efficient and structured manner.

Megha Vij
M.Tech.(CSE)
Department of MUC

Contents

Introduction

In recent years, mobile sensing has been used in myriad of interesting problems that rely on collection of real-time data from various sources. The field of crowd-sourcing or participatory sensing is extensively being used for data collection to find solutions to day-to-day problems. One such problem of growing importance is analytic for smart cities. As per Wikipedia, a smart city is defined as an urban development vision to integrate multiple information and communication technology (ICT) solutions in a secure fashion to manage a city's assets and notable research is being conducted in this direction. One of the important assets is transportation. With the growing number of people and shrinking land space, it is important to come up with solutions to ease commuting between any two locations within the city. The most efficient mode of transportation in a city is metro train or metro or subway or tube, as it is called in different parts of the world. Cities, in developed countries, such as New York, London [1], etc, have millions of bytes of data being logged on a diurnal basis using RFID sensors installed at metro stations. This data can be leveraged to carry out an extensive analysis to understand usage of metro across the city's layout and aid in the process of better and uniform city planning. However, a common challenge faced in developing countries is the paucity of such detailed data due to lack of sensors in the infrastructure.

In the absence of these statistics, the most credible solution is to collect data via participatory sensing using low-cost sensors like accelerometer, gyroscope, magnetometer, GPS, and WiFi that come packaged with modern day smartphones. These sensors can be used to collect data on behalf of users, which upon analysis can be leveraged in the same way as data made available through infrastructure-based techniques. Through this work, we wish to predict the metro station activity in the city of New Delhi, India using accelerometer logs collected from a smartphone app. However, the problem is not limited to only smart cities in developing countries, e.g., our work is applicable for personal health. Smart bands and watches are used these days to detect users' activities, such as sitting, walking, and running. The learning models presented in this thesis can detect travel by train. The lessons learned in this work can further be applied to detect travel by other modes of transportation, such trams, buses, etc. that follow a designated pattern which can be detected easily using sensors by adjusting a few parameters.

Our approach is to detect commuter's entry into metro station and thereon measure time spent by the commuter till he/she boards the train. Similarly, we measure the time spent after disembarking the train till exiting the metro station. These two time duration are indicatives of the rush at metro stations, i.e., more the rush, more the time spent. A common technique used is leveraging geofencing APIs on the smartphones to detect entry into and exit from the metro stations. These APIs efficiently use GPS to detect whether the user has crossed a boundary, in our case perimeter around the metro station. For detecting boarding and disembarking, we use accelerometers on the smartphones to detect whether the commuter is in a train or not. Unlike geofencing, the latter is an unsolved problem. We treat this problem as a two class classification, where the goal is to detect

whether a person is traveling in a train or is at the metro station. Furthermore, the "in-metro-station" activity is a collection of many micro-activities including walking, climbing stairs, queuing, etc and therefore, needs to be combined into one single category for the classification. We map this problem to machine learning and explore an array of classification algorithms to solve it. We attempt to use discriminating, both probabilistic and non-probabilistic, classification methods to effectively distinguish such patterns into "in-train" and "in-metro-station" classes.

Main contributions through this work

- We show that ECDF-based representation, on an average, performs better than a feature set based on order statistics, for detecting movement in train. The Ensemble Improvement for SVM yields a TPR of 0.98 for ECDF against 0.72 for order statistical features.

- We compare among popular classification techniques of SVM, RDF, and Regularized Logistic Regression and find that RDF gives the best performance with about 94% predictive accuracy.

- We encounter class imbalance problem. We handle it by using ensemble learning techniques, namely, (SVM + Ensemble). It gives the best performance, among available option, with an accuracy of 98.65% with ECDF representation and 91.08% with Statistical Features.

Though the main focus of this thesis is on accelerometer data for inferring episodes of travel in a metro, it is important to note that the conclusions drawn in this thesis can be generalized to other mobility-related application scenarios as well. For instance, one of the main challenges our classification problem was addressing the diversity of activities (e.g., sprinting, waiting in queue, climbing stairs, motion in train, etc.) recorded in the data. This diversity makes it non-trivial to discern a pattern of interest (e.g., motion in train) from others, because decision boundaries can no longer be expected to be linear in any realistic scenarios. Such cases can also arise in application scenarios, such as estimating traffic congestion pattern in an urban transportation network from accelerometer data, predicting queuing delay at metro stations, and favorite modes of transportation in an urban city. Basically any application, which needs to deal with diversity in classes could benefit from this study. Furthermore, given the diversity in classes it is quite likely to also have a class imbalance problem, where some data points from some classes be disproportionately high in number, thus making the model biased towards one class. Our findings in this regard (SVM + Ensemble) are also of general value to other studies on mobile sensor data facing similar issue.

1. Problem Definition

This section formally defines our problem statement. The input for our problem is raw sensor data from the accelerometer sensor embedded in smartphones. The objective is to build a classification model so as to distinguish between metro train activity and metro station activity. In other words, given raw data from the accelerometer sensor, over a time-window as explanatory variable(s), a target variable is used to determine whether user of the smartphone is currently "in-train" or "in-metro-station."

1.1 Challenges posed by the Problem

It is important to note that building a classification model on accelerometer data is a *challenging task* for two reasons. First, stream data coming from an accelerometer would intrinsically contain readings from a wide variety of motion activities. For instance, even if the accelerometer sensor is read only when the phone is in metro train journey (in-metro-station + in-train), it would capture data from a myriad of activities like, motion of the train, walking, climbing, queuing at a metro station, standing-up, sitting-down in the train, dropping or using the phone. This makes it non-trivial to build a classification model as it would have to discern the train-activity from mixture of several other motion activities each having its own signature accelerometer values. Secondly, many times, depending on data collection strategies, one could have a class imbalance problem. This means that the learning data set would have disproportionately high number of samples from one of the classes, thereby making the model biased towards the majority class.

Our study considers the following four facets of this problem:

1. What is the ideal quantization of the raw sensor data which comes as a stream? In other words, what is the ideal window size from which features need to be extracted?

2. Given a time-window, what features best discern between "in-a-train" and "in-a-metro-station" classes?

3. Given a time-window and its feature representation, which classification model addresses the challenge of diversity in the motion activities captured by the accelerometer and gives best accuracy for our two-class problem?

4. If there is a class imbalance in the data samples, then what is the best way to resolve it during learning of models?

1.2 Defining the Scope of the Problem

The classes "in-train" and "in-metro-station" are interpreted as typical mobility patterns observed during the course of data collection. We understand that there can be non-typical motion patterns in either of these classes, for instance, rare events like dropping or using the phone while in the train, walking towards the

platform, waiting for the train at platform, using the escalator/elevator, but those would not be the main focus of the models learned. Instead, our model would implicitly consider them as outlier data points, i.e., they would not try to accommodate them towards improving prediction accuracy.

1.3 Understanding with a Sample Trace

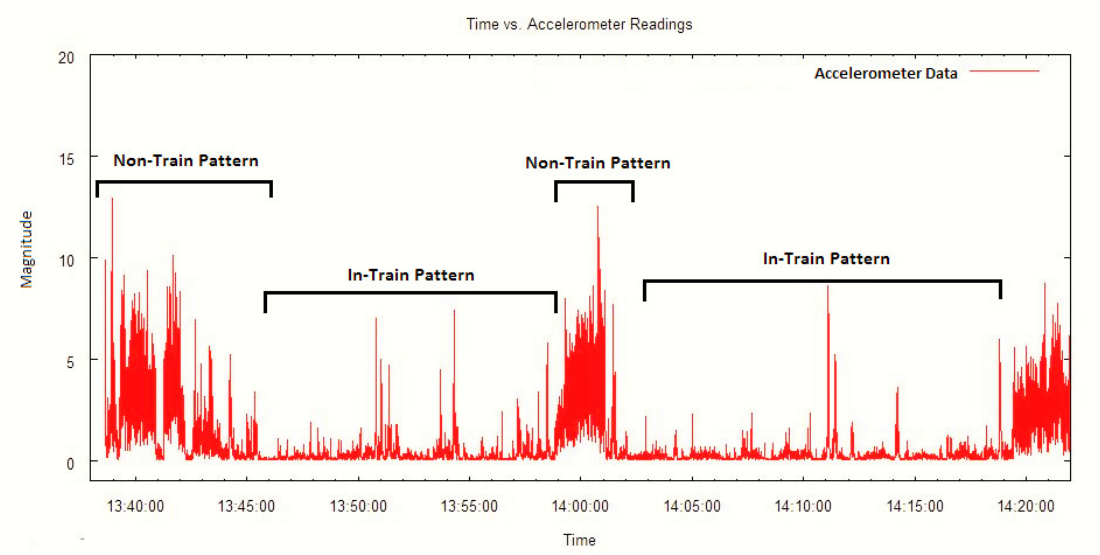

Figure 1.1: Sample accelerometer data annotated with the two mobility patterns,"in-train" and "in-station"

Sample data: Figure 1.1 illustrates "in-train" and "in-metro-station" classes on sample accelerometer trace that we collected. One may observe a stark difference in accelerometer values across the two classes. The values for the "in-train" class have less variance, whereas for the "in-metro-station" class, we observe heavy fluctuations (and thus much greater variance). This was expected since a train typically moves at a uniform speed, whereas in a station we would witness several small activities such as walking, waiting, climbing, sprinting etc. This diversity affects the classification algorithms in unique ways resulting in dominance of some techniques over others. We need to handle this diversity.

2. Related Work

Participatory sensing using smartphone sensors is gaining a lot of importance due to their ubiquity. For the problem of tracking mode of transportation, smartphone based sensors such as accelerometer, GPS, and WiFi are used extensively for mode detection, start/stop detection, and arrival time predictions. In this work, we focus on how to efficiently process the accelerometer data and draw inferences about travel by metro trains from the processed data.

2.1 Data Mining Techniques

A majority of research in the sensor data analysis revolves around finding signature patterns in time-series data using data mining techniques such as SAX (Symbolic Aggregate Approximation), etc. This area of research comes under motif analysis [2], which aims at finding recurrent patterns within the time-series data. It may be used to encode various activities using motifs for a feature-space description. However, we use the time-series accelerometer readings as an input to a classification method. This entails describing the feature-space using statistical and distribution based measures. Once, we obtain a feature set, an appropriate classifier model is built to facilitate the prediction of "in-train" and "in-metro-station" patterns with appreciable accuracy.

2.2 Feature Representation for Accelerometer Data

Apart from using data mining techniques, there is a body of work done based on extraction of characteristic or statistical features from the sensor measurements. Sensor data is typically a multi-dimensional function of time. There are two types of features that can be extracted from sensors,that is, in the time domain and frequency domain. A study based on feature extraction was conducted by Hemminki et al. [3], which delved into the gravity estimation of the acceleration and the derivation of features from the horizontal components of acceleration in order to effectively model the different transportation modes. The researchers used statistical features in addition to frequency, time, peak, and segment-based features to decompose the task into a hierarchical model consisting of a separate classifier for walking, stationary, and kinematic motion.

Xia et al. [4] employed frequency-domain based feature set including DFFT components from specific time windows, sample entropy features that are then fed to a suitable classifier to detect outdoor transportation modes.

Prentow et al. [5] have used kinetic, position, and signal-strength based features for detecting mode of transportation in indoor environment. They also explored the use of ECDF-based representation which is essentially a data transformation technique and have shown that ECDF has better generalization capability than traditional feature sets. Signal-strength based sensors like GPS and WiFi perform well for indoor settings and the paper claims to achieve an appreciable accuracy with a feature set based on kinetic attributes.

Hammerla's paper [6] on the use of ECDF for accelerometer data shows that this representation effectively captures the spatial distribution of the data and adapts to the available training data better than other approaches. ECDF is known to perform very well in sensor data analysis and has been extensively used in activity recognition. Through our work, we analyze the use of ECDF representation for metro travel patterns and compare it against the feature representation based on order statistics.

2.3 Other Sensors for Transportation Mode Detection

The use of location sensors in transit tracking is common as in Thiagarajan's [7] work, where a combination of kinetic and location sensors is used to detect if a person is traveling by a bus or not. The walking/stationary activities are measured using an accelerometer. These accelerometer readings are used as a trigger to start GPS sensing. Apart from accelerometer and location sensors, there has been use of other sensors as well. Higuchi et al. [8] have used magnetometer to detect stopping and starting of train using passengers' smartphones. While their work dealt with the precise detection of the intermittent stops during a metro train journey, our work in the area of metro railways is different in the scope of classification as we deal with accurately detecting whether a person is at the metro station or traveling in the train. Moreover, the use of magnetometers may not be feasible for "in-metro-station" detection. Using accelerometers gives us a sense of different activities like walking, sprinting, train motion, etc. by measuring the magnitude of acceleration that can be combined into one broad class of activities governing "in-metro-station" behavior. Further, while accelerometer is found on all smartphones, magnetometer is available only on high-end ones.

3. Approach & Methodology

This section details the methodology used in this thesis work. We first present a description of the features used for classification. Following this, we briefly explain the primary classification techniques used in this study. At the end, we detail the techniques used to address the class imbalance challenge of the data set.

Table 3.1 illustrates the "space" of classification algorithms × features used in this study. For finding out features, we studied order statistics and empirical cumulative distribution function. For classification algorithms, we studied variants of SVM, Random Decision Forest (RDF), and Regularized Logistic Regression. For accommodating class imbalance, we explored ensemble and differential error cost (DEC) techniques (see Table 3.1).

	SVM	RDF	(SVM + DEC)	(SVM + Ensemble)	Reg. LR	Reg.LR+Ensemble
Order Statistics	Algo1.1	Algo1.2	Algo1.3	Algo1.4	Algo1.5	Algo1.6
ECDF	Algo2.1	Algo2.2	Algo2.3	Algo2.4	Algo2.5	Algo2.6

Table 3.1: Classification Algorithms across the Feature-space

3.1 Feature Representation

Our approach involves using following two feature sets to represent the accelerometer data:

- Order Statistics Features - Given a window (as quantized from a time series), this method consists of computing typical statistical metrics. These statistical metrics become our features representing a particular window. In this study, we used mean, median, standard deviation, and mode to create the feature set. Mode of the acceleration is selected because certain activities for "in-metro-station" patterns, such as running and climbing escalators, have different peaks for different activities like, walking, climbing, and waiting.

- ECDF Based Features - In contrast to order statistics, these features are extracted as different quantiles from an empirically cumulative distribution function (ECDF) for values in a given window. Features based on ECDF can be considered to be more robust, in the sense that, they are able to capture the different modes in the underlying data by determining a complete distribution function of values over the given window. This technique is particularly useful when the probability density function over a given window has more than one mode. Forcibly summarizing more than one mode using a single statistic (as done by the previously described order statistics based technique), such as mean, median, etc., destroys the rich structure representing the data.

  Given a time window (represented as a time series of accelerometer data) ECDF is computed as given in 3.1. Assume that X denotes the random variable representing the accelerometer values in the given window. We first

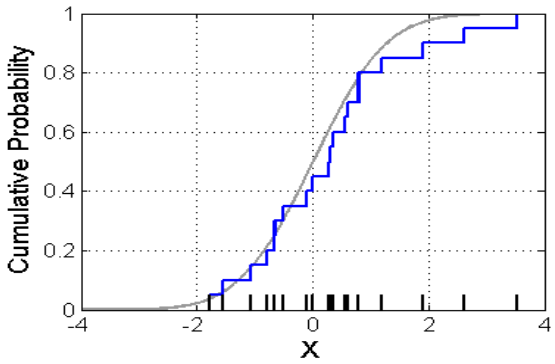

Figure 3.1: Sample Empirical Cumulative Distribution function shown in blue stepped line

empirically determine a cumulative distribution function for the random variable X over the given window.

$$F(n) = \frac{n(X \leq d)}{n} = \frac{1}{n} \sum_{i=1}^{n} 1_{x_i \leq d} \tag{3.1}$$

Here, $n(X \leq d)$ denotes the number of points in the sample (the given time-window) less than or equal to d and n is the total number of points in the given window.

Note that since our given window is only a discrete sample of a continuous random variable (accelerometer can have infinitely many values). Therefore, we can only estimate the cumulative distribution function (CDF). In other words, we cannot compute the CDF using the traditional way, where $CDF = \int_a^b f(x)dx$, where $f(x)$ is the probability density function (PDF). Using the previous equation, we compute the value of F for all the data points in the given window. A plot of these values of F against the values of X (sample points) comes out as a step function that increases from from 0 to 1. Figure 3.1 illustrates this plot for a hypothetical window.

After computing the cumulative distribution function over a window, we perform a cubic interpolation using Kaplan-Meier Estimate [6]. This cubic interpolation fits a smooth curve (Figure 3.1) over a stepped function (shown in blue in Figure 3.1). We then choose θ equally spaced points (our quantiles) on the y-axis of Figure 3.1 and find the corresponding points of the x-axis using the inverse of the cubic function interpolated over the ECDF. These points on the x-axis denote our features. θ is a hyperparameter that needs to be tuned through experimental analysis. We found that $\theta \in [20\ 30]$ is a range that gives acceptable accuracy across the learning algorithms. This number is also in keeping with the curse of dimensionality, which relates the size of the training data with the number of features in the data. In general, it is customary [9] to have θ (# features in each data point) such that $(\#windows - in - training - data/\theta) > 10$.

3.2 Classification Techniques

This subsection provides details on the classification algorithms used to learn models on the window level data represented using the feature representations discussed in the previous subsection. Following are the classification methods.

- Support Vector Machines (SVM) - SVM is a popular classification method for a two-class problem. SVM attempts to find a decision boundary that maximizes the margin of separation between two classes. The characteristic data points that define the maximum margin separator are known as support vectors. This allows SVM to be more robust on unseen data, and thus has fewer misclassifications in the testing phase. Following equation shows the objective function of SVM. Here, W is the weight vector, C is the cost of misclassification, and ξ's are the slack variables. The term W^TW ensures that the margin $(2/||W||)$ is maximized.

$$\begin{aligned} & f(w,b,C) = \min_{w,C}(\frac{1}{2}W^TW + C\sum_{i}^{n}\xi_i) \\ & \textit{subject to} \\ & Y_i(W^TX_i + b) \geq 1 - \xi \forall i \textit{ and } \xi \geq 0 \end{aligned} \tag{3.2}$$

  Depending on the separability of the classes in the underlying data, we can choose whether to train using a Linear SVM or Kernel-based/non-linear SVM. Figure 3.2 and Figure 3.3 show visualization of the two types of features against the two categories. Figure 3.2 illustrates the separation between "in-metro-station" and "in-train" patterns based on the mode computed for each window. Figure 3.3 shows the separation based on one of the θ features in the ECDF representation. Both Figure 3.2 and Figure 3.3 show that the classes don't have a linear boundary. The potential linear decision boundary by a linear SVM as shown in 3.2 will incur lot of misclassifications and may not be beneficial for our classification task. Thus, we may need to explore a complex non-linear decision surface that clearly shows the separation between the classes. To this end, we used Kernel-based SVM that projects the data in a higher-dimensional space. It is typical that classes, which are not linearly separable in lower dimensions, become separable in higher dimensions.

- Random Decision Forest (RDF) - RDF [10] is a bagging ensemble learning technique that uses several sub-samples of the data set (usually, with replacement) for training weak decision tree learners. Given a collection of windows (the training + test data set), RDF algorithm first samples this data set to get about 66% of the data points (each data point is a window represented using either order statistics or ECDF based features). It then goes about to learn a decision tree on this data. Following this, the remaining 33% data is passed through this decision tree to get a predicted class label for each of these 33% of data points and store it separately. Note that this is still not the "testing part," i.e., we would not compare the predicted

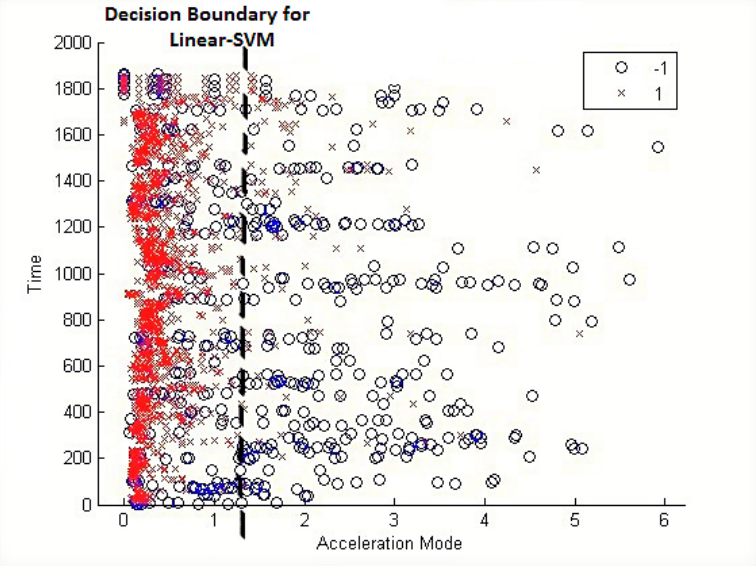

Figure 3.2: Distribution of Quantized Data (40-seconds) across "in-train" and "in-metro-station" classes based on Mode of Acceleration

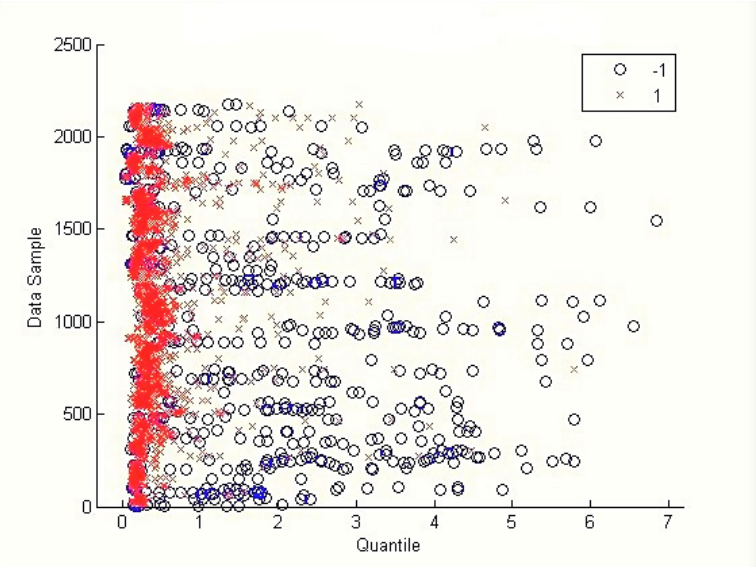

Figure 3.3: Distribution of Quantized Data (40-seconds) across "in-train" and "in-metro-station" classes based on an ECDF feature dimension

class label of these data points with the true class. The algorithm repeats this process of picking 66% of data, learning a decision tree and predicting classes for 33% of left out data and storing them for about 20 rounds. At the end of about 20 rounds we would have following two things: (a) 20 decision trees, each learnt on a small sub-sample of the data, (b) predicted class values of several of the data points. Now, for each of the data points for which some (or all) of the individual trees have predicted a class value, we set the final predicted class according to majority voting and compute the misclassification errors by comparing with their true class labels. This becomes the final test error of the model. The set of 20 decision trees, each of which was learnt during a round, become the final classification model where final class is always decided through majority voting.

- Regularized Logistic Regression - Here, we model the problem as predicting the probability of being in the train (i.e. $Y = 1$) as a function of a few independent/explanatory variable(s), i.e. X, which is order statistical or ECDF-based features. We try to model the relationship between Y (dependent variable) and X by fitting a logistic curve that is also capable of learning a non-linear relationship.The equation governing the relationship between X and Y is given as follows:

$$ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X \tag{3.3}$$

$$p = \frac{\exp^{\beta_0+\beta_1 X}}{1+\exp^{\beta_0+\beta_1 X}} \tag{3.4}$$

where β_0, β_1 = coefficients of the regression equation
X = independent/explanatory variable
p = probability that Y = 1
(1-p) = probability that Y = 0

We use the maximum likelihood estimate (MLE) to find optimal values of β_0 and β_1, such that the function closely estimates the observed data. We take the log-likelihood of the objective function as it is easier to deal with it mathematically. The formulation of the problem is given below:

$$L = \prod_{i=1}^{n} p(y|x)^{Y_i}(1-p(y|x))^{(1-Y_i)} \tag{3.5}$$

$$l \;=\; log(L) \;=\; \sum_{i=1}^{n} Y_i log[p(y|x)] \;+\; \left(n - \sum_{i=1}^{n} Y_i\right) log[1 \;-\; p(y|x)] \tag{3.6}$$

The above equation is solved using the first derivative test through iterative computing to find the maximum possible value of the likelihood function. The initial values of the coefficients are chosen to be zero or any random real number.

For this study, we have taken a regularized estimate of regression to understand the relative importance of each feature thereby preventing the problem of over-fitting. Though the number of features being considered for the problem is limited, we need to provide a check for the parameters not contributing to over learning. The regularization parameter, α is added to the objective function and it acts as a penalty for those features that do not add value to the model estimation. The revised formulation is as follows:

$$l \;=\; \sum_{i=1}^{n} Y_i log[p(y|x)] \;+\; \left(n - \sum_{i=1}^{n} Y_i\right) log[1 \;-\; p(y|x)] \;+\; \alpha \sum_{i=1}^{n} \beta_i \tag{3.7}$$

3.3 Mitigating the Effect of Class Imbalance

As we observe that our accelerometer data is highly biased towards the "in-train"(positive) category, a biased learning will lead to very large amount of samples being incorrectly classified into the majority class label. Skewness of any learning algorithm is likely to degrade its performance towards the minority class. To counteract the effect of class imbalance, we use two well-known methods and study the effect on the overall performance of SVM and Logistic Regression. Note the problem of class imbalance did not plague the RDF technique in our study.

- Different Error Cost (DEC) - This is an internal algorithmic modification to the existing SVM formulation to take into account the class imbalance [11]. It ensures that while learning the objective function for the given data, the effect of inherent bias in the model is mitigated. The premise of this modification is to assign a higher misclassification rate (error cost) to the samples from the minority class as compared to those from majority class. The improved formulation is given as follows:

$$\begin{aligned} & f(w, C^+, C^-) = \frac{1}{2} w^T.w + C^+ \sum_{i|y_i=+1}^{n} \xi_i + C^- \sum_{i|y_i=-1}^{n} \xi_i \\ & \textit{subject to } y_i(w.\phi(x) + b) \geq (1 - \xi_i) \textit{ and } \xi_i \geq 0 \end{aligned} \tag{3.8}$$

- Ensemble Learning - This is an external data preprocessing technique, similar in working to the traditional ensemble learning techniques such as Bagging (as in RDF) [12] and Boosting. The minor difference lies in the way the data set is partitioned, commonly referred to as downsampling. For this technique [11] to counteract class imbalance, we compute the proportion of minority class data and divide our majority class data set in such a way that each sub-sample created from the data set has the same number of samples as the minority class samples. The sub-sampling of the majority class data is done randomly without replacement. These sub-sampled data sets are combined with the minority class samples to form a uniform data set. Therefore, models are trained taking all possible combinations of the majority class with the minority class to remove any unaccounted bias. n such models are trained using a Kernel-SVM and while testing, the final predicted label for the data samples is decided by majority voting.

4. Experimental Evaluation

This section elucidates the experiment conducted, providing details about how data is collected from the metro stations and processed for further analysis. Through the remainder of the section, we discuss in detail the findings of the extensive study carried out with regard to the classification methods used and the evaluation metrics inferred from them. We also endeavor to understand why one classification model performs better or worse than the others.

4.1 Collection of Data

For the purpose of detecting train activity at the Metro stations all across New Delhi, India, we built an Android app to log accelerometer data when users enter a metro station. For the purpose of training, the app requires the volunteers to manually record the entry/exit times from/into the station and train. This information is used as "ground truth" or "true" class labels for evaluation. The app lets users record their entry/exit stations. This information helps to rule out the effect of a station's infrastructural layout on the accelerometer readings.

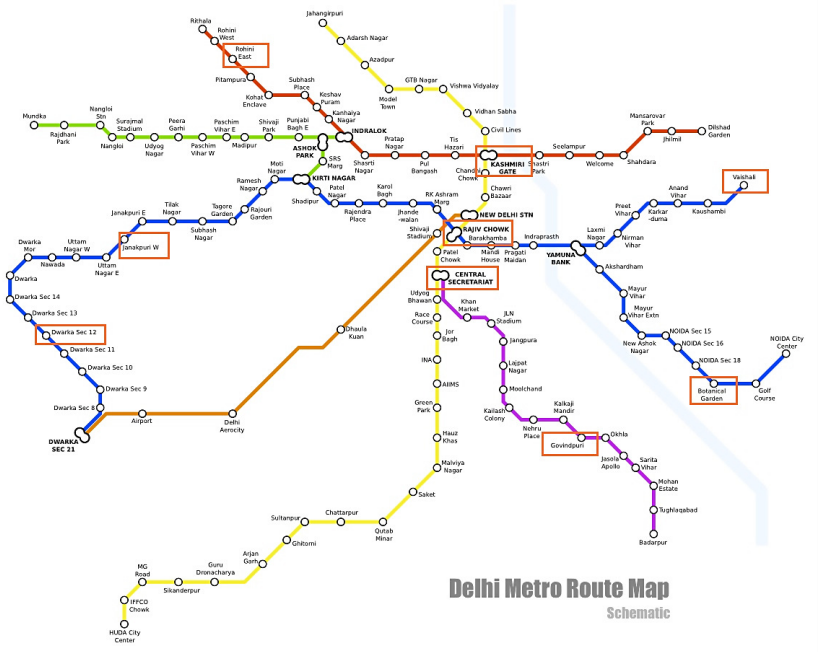

Figure 4.1: Schematic Map of Delhi Metro annotated with the stations, in which we collected data

Seven volunteers logged data using the app at different metro stations. The data logging was performed with the following devices- Sony Xperia, One Plus Two, Samsung GT Series, and Motorola. The purpose of this exercise is to eliminate the effect of accelerometer sensors across different phone models on the detection accuracy. Majority of the accelerometer data was collected from the metro stations as illustrated in Figure 4.1. Among the nine junction metro stations in New Delhi, we procured data from three metro stations, namely, Rajiv

Chowk, Central Secretariat, and Kashmere Gate. Among all the metro stations, these three stations witness the highest annual footfall. The remaining stations witness relatively low rush but they were selected to cover all the major metro railway lines across the city. After a trace is collected, it is uploaded to a centralized Django server through the app and stored as raw dump files consisting of the accelerometer readings and the timestamps for entry and exit into the stations. Through this experiment, we collected about 23 hours of accelerometer data.

4.2 Preparation & Preprocessing of Data

A preliminary analysis of the data reveals that on an average, 25-30% data collected falls in the "in-metro-station" category. This suggests that the task of metro station activity recognition encounters a class imbalance problem identified in Chapter 3.

The raw data was logged from the accelerometer at a maximum sampling frequency of 50Hz notwithstanding small variations across different accelerometer models. It is measured along each of the three axes using a gravity sensor. However, in order to calculate the real acceleration of the device, we isolate the gravity component from the logged values assuming that the force of gravity is acting only along the y-axis (g = 9.81m/s^2) and is removed from all other two axes.

This was done considering the most generic position of the phone, i.e., the downward vertical direction while the phone is placed in a shirt/pant pocket. We calculate the linear acceleration by subtracting the gravity components weighted by a parameter called α, which is decided by the type of low-pass filter being used for computation. The linear acceleration along the three axes and magnitude of the acceleration are formulated as elucidated by the following equations.

$$
\begin{aligned}
acc(x) &= \alpha * acc(x) - (1-\alpha) * gravity(x) \\
acc(y) &= \alpha * acc(y) - (1-\alpha) * gravity(y) \\
acc(z) &= \alpha * acc(z) - (1-\alpha) * gravity(z) \\
mag(acc) &= \sqrt{(}acc(x)^2 + acc(y)^2 + acc(z)^2)
\end{aligned}
$$

Once the magnitude of linear acceleration is computed for the entire time-series data, we need to quantize data into windows of suitable length. We first compress the readings of each second by taking a mean of all the values. Thereafter, we take *non-overlapping* windows of 20, 30, and 40 seconds each. These windows constitute individual data points for the time-series data set. The decision to use non-overlapping windows is to counteract the effect of scrambling of data that may occur by sharing of time-segments across adjacent frames. It is shown that "in-metro-station" patterns are an amalgamation of several micro activities, which are beyond the scope of our classification task.

In spectral analysis (frequency domain), the purpose of using overlapping windows is to account for the loss of information across the edges of windows. For the purpose of our study which focuses on distinguishing metro station activity from train activity, overlapping time-segments from a previous window in the current window can have a disparaging effect on the prediction behavior of classifiers as we may have several windows with fractions of both train and station mobility

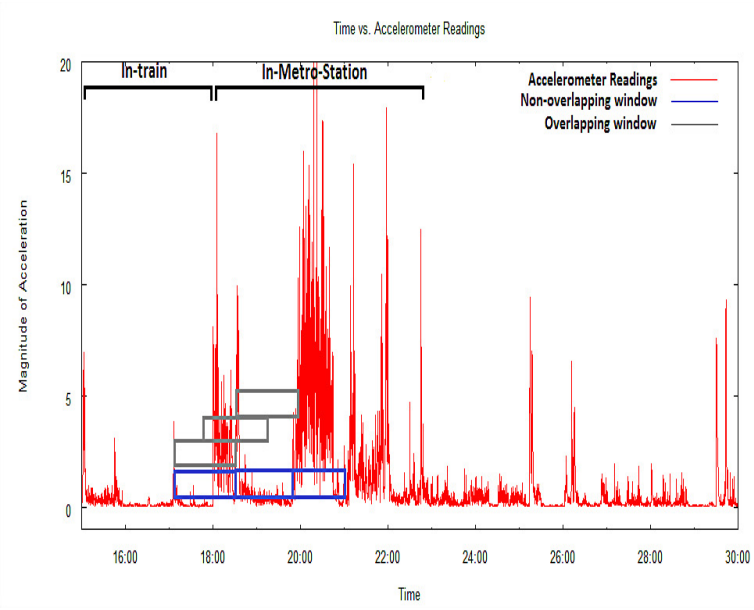

Figure 4.2: Consideration of Non-overlapping vs.Overlapping Windows for a given Accelerometer Trace

patterns. Due to this, it will be hard for the classification model to effectively separate the two classes. Whereas, the non-overlapping windows attempt to disambiguate better, especially for patterns that are marked by transition from train to station and vice-versa. This effect is illustrated through a sample of accelerometer trace 4.2. Hence, we expect that the non-overlap, which is a courser level of granularity, is well-suited to our problem and is likely to give a better separation between the two classes.

Feature Representation	WS=20s	WS=30s	WS=40s
Order Statistics	4327x4	2891x4	2174x4
ECDF	4327x21	2891x21	2174x21

Table 4.1: #windows × #feature set dimensions for different window sizes

4.3 Evaluation Metrics

We have tested our models using n fold-cross validation on the entire data set. Here, n refers to the number of folds or partitions of data created from the entire data set. In n-fold validation we train the model on $(n-1)$ folds and test on the remaining fold. This process is repeated n times and errors are averaged. For our analysis, we have set $n = 10$. This type of validation helps in ensuring that the learned model does not suffer from any bias or does not over-fit the data. The evaluation of our initial experiment and results reported are done using the entire data set. Basically, through this exercise, we aim to estimate the performance of classification models by calculating the true error rate (or equivalently prediction accuracy) i.e. the classifier's accuracy over the entire population.

We have used confusion matrices to report our results. Each of the confusion matrices reported in this section is essentially the sum of confusion matrices

obtained across the individual folds of n-fold cross validation. We also report the True Positive Rate (TPR) and True Negative Rate (TNR) on these summed up confusion matrices. Evaluation on the basis of confusion matrix is useful for the prediction of metro train activity from the perspective of addressing the challenge of class imbalance towards "in-train" patterns present within the data and how it can be diminished by using the improvised versions of the classic learning algorithms. On careful analysis of the results illustrated in Tables 4 through 12 for all learning models described in Table 3.1, the following observations can be made.

1. Impact of Window Size - Across the three levels of quantization, we note that a 40 second-long window/frame provides the best generalized bound on accuracy. A plausible explanation for this could be that a 40-second window is long enough to capture the transition of getting into the train and the train picking its uniform speed (and its reverse while getting off the train). In smaller windows, we would get accelerometer values for several mini-activities such as initial acceleration (and deceleration before stop) of the train creating a lot of variance inside a class, thus making it difficult for classifiers to distinguish effectively.

2. Data Representation - With regard to the data representation most suited to this problem, we observe that ECDF gives much more generalized accuracy bound than the model based on statistical features. This was expected as single statistical measures (e.g, mean, median, mode etc.) are either vulnerable to outliers or assume only one mode to be present in the underlying distribution. On the contrary, ECDF-based representation aims at capturing the actual shape of the data distribution and makes it robust to outliers. Although, the improvement in accuracy of models seems modest, the analysis of TPR and TNR will justify the prominence of ECDF over the traditional statistics based model.

3. Classification Model - The ensemble learning methods, such as RDF and SVM+Ensemble, perform considerably better than the other classification methods. The simple/base models show a degraded performance because of the class imbalance problem that introduces a bias in prediction towards the majority class, which in this case, is the "in-train" class. On the contrary, ensemble methods work by sub-sampling the data, such that each bootstrap sample represents a uniform distribution across the two categories. Furthermore ensemble based techniques turned out to be robust against variance (i.e., one class having several mini-activities) in the data.

   We observed that classification models that implicitly use random sub-sampling like RDF do not suffer from the class imbalance challenge. The type of random sub-sampling (downsampling) used for our study is where we randomly subset samples from the majority class so that its frequency of occurrence is the same as that of the minority class. This gives us a balanced data set to train on and attain a better model fit. We attain an accuracy of 94.25% with the RDF classifier and over 98% accuracy with the SVM+Ensemble approach. This also shows that the class imbalance

problem confronting our detection activity can effectively be solved by employing these improvements.

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	266	254
	In-Train	78	1576
30	In-Metro-Station	340	341
	In-Train	110	2100
20	In-Metro-Station	490	530
	In-Train	142	3165

Table 4.2: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo1.1 (Kernel-SVM + Order statistics with TPR=0.51, TNR=0.95

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	282	238
	In-Train	94	1560
30	In-Metro-Station	357	322
	In-Train	132	2078
20	In-Metro-Station	532	488
	In-Train	194	3113

Table 4.3: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo2.1 (Kernel-SVM + ECDF) with TPR=0.54, TNR=0.94

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	384	136
	In-Train	0	1654
30	In-Metro-Station	500	181
	In-Train	0	2210
20	In-Metro-Station	723	297
	In-Train	0	3307

Table 4.4: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo1.2 (RDF + Order Statistics) with TPR = 0.74 and TNR = 1

4.4 Performance of Regularized Logistic Regression

Upon analyzing the confusion matrices for all algorithms, we observe that the performance of Logistic Regression shows an awry trend, indicating that it may

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	392	125
	In-Train	0	1654
30	In-Metro-Station	504	176
	In-Train	0	2210
20	In-Metro-Station	741	279
	In-Train	0	3307

Table 4.5: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo2.2 (RDF + ECDF) with TPR = 0.75 and TNR = 1

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	305	215
	In-Train	114	1540
30	In-Metro-Station	368	313
	In-Train	128	2082
20	In-Metro-Station	549	471
	In-Train	186	3121

Table 4.6: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo1.3 (SVM-DEC + Order Statistics) with TPR=0.58 and TNR=0.93

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	303	217
	In-Train	114	1540
30	In-Metro-Station	387	294
	In-Train	153	2057
20	In-Metro-Station	563	457
	In-Train	226	3081

Table 4.7: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo2.3 (SVM-DEC + ECDF) with TPR=0.58 and TNR=0.93

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	375	145
	In-Train	33	1621
30	In-Metro-Station	470	211
	In-Train	38	2172
20	In-Metro-Station	683	337
	In-Train	93	3214

Table 4.8: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo1.4 (SVM-Ensemble + Order Statistics) with TPR=0.72 and TNR=0.98

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	507	11
	In-Train	8	1646
30	In-Metro-Station	660	21
	In-Train	15	2195
20	In-Metro-Station	968	52
	In-Train	30	3277

Table 4.9: Confusion matrix showing a 40-second window performing better than other quantized windows for Algo2.4 (SVM-Ensemble + ECDF) with TPR = 0.97 and TNR = 0.99

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	236	284
	In-Train	66	1588
30	In-Metro-Station	293	388
	In-Train	88	2122
20	In-Metro-Station	437	583
	In-Train	129	3178

Table 4.10: Confusion matrix for Algo1.5 (Reg. Logistic Regression + Order Statistics) across all window sizes with TPR=0.45 and TNR=0.96 for 40-sec

Window Size(sec)	Class	In-Metro-Station	In-Train
40	In-Metro-Station	234	286
	In-Train	64	1590
30	In-Metro-Station	292	389
	In-Train	79	2131
20	In-Metro-Station	435	585
	In-Train	121	3186

Table 4.11: Confusion matrix for Algo1.5 (Reg. Logistic Regression + ECDF) across all window sizes with TPR=0.45 and TNR=0.96 for 40-sec

Window Size(sec)	Class	In-Metro-Station	In-In-Train
40	In-Metro-Station	344	176
	In-Train	193	1461
30	In-Metro-Station	447	234
	In-Train	262	1948
20	In-Metro-Station	650	370
	In-Train	377	2930

Table 4.12: Confusion matrix for Algo1.6 (Reg. Logistic Regression-Ensemble + Order Statistics) with TPR=0.66 and TNR=0.88 for 40-sec

not be a suitable learning model for our problem. Logistic regression aims to estimate the probability of a sample belonging to a particular class by fitting a sigmoid (logistic) curve to the basic regression equation given a set of explanatory variables. Samples yielding a logistic value above a certain threshold are said to

Window Size(sec)	Class	In-Metro-Station	In-In-Train
40	In-Metro-Station	364	156
	In-Train	236	1418
30	In-Metro-Station	474	207
	In-Train	343	1867
20	In-Metro-Station	694	326
	In-Train	473	2834

Table 4.13: Confusion matrix for Algo2.6 (Reg. Logistic Regression-Ensemble + ECDF) across all Window Sizes with TPR=0.7 and TNR=0.85 for 40-sec

Data Representation	SVM	RDF	SVM+DEC	SVM+Ensemble	Reg. LR	Reg.LR+Ensemble
Statistics	84.10	93.13	84.84	90.06	83.53	82.73
ECDF	84.22	93.55	84.11	98.10	83.67	81.53

Table 4.14: Predictive Accuracies with Window Size = 20s

Data Representation	SVM	RDF	SVM(DEC)	SVM(ENS)	Reg. LR	Reg.LR+Ensemble
Statistics	84.27	93.74	84.78	91.38	83.50	82.84
ECDF	83.74	96.22	84.51	98.75	83.78	80.97

Table 4.15: Predictive Accuracies with Window Size = 30s

Data Representation	SVM	RDF	SVM(DEC)	SVM(ENS)	Reg. LR	Reg.LR+Ensemble
Statistics	84.32	93.75	84.83	91.81	83.53	83.03
ECDF	84.70	94.25	84.73	99.12	83.67	81.96

Table 4.16: Predictive Accuracies with Window Size = 40s

belong to the positive class and those below it belong to the negative class.

However, when we use MLE (maximum likelihood estimate) to find an optimal fit for our data, we observe that values of the logistic function for several samples from the "in-metro-station" class overlap with those of the "in-train" class, which makes it difficult to come up with a definite threshold. Due to this reason, there is a higher likelihood of "in-metro-station" samples (minority class) being misclassified in to "in-train" class.

The high degree of overlap for both the feature representations is shown through Figures 4.3 and 4.4 which consequently leads to performance degradation in case of Logistic Regression. We further observe through the accuracy analysis that the overall performance of Logistic Regression for ECDF representation is reduced as compared to the model based on order statistics. The model based on ECDF representation attempts to balance out the misclassifications in the two classes (by approximating the real shape of the data distribution), but in the absence of a clear-cut threshold for distinguishing the samples, the model suffers.

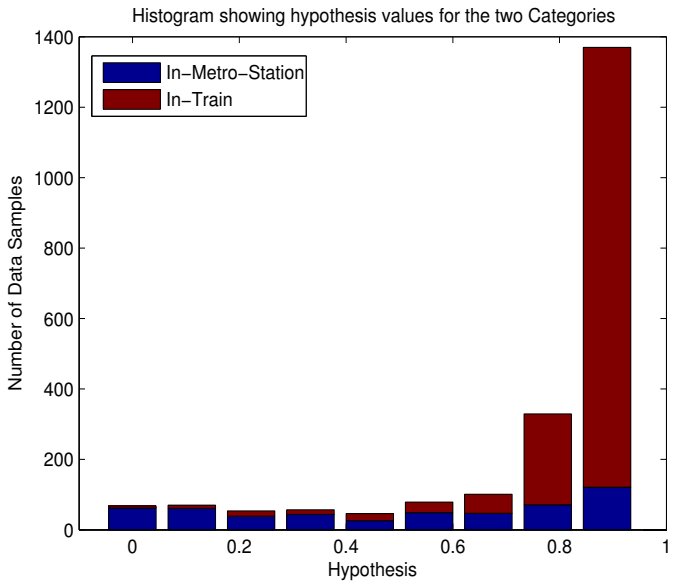

Figure 4.3: Histogram illustrating Overlap among hypothesis values across the two classes for order statistics data

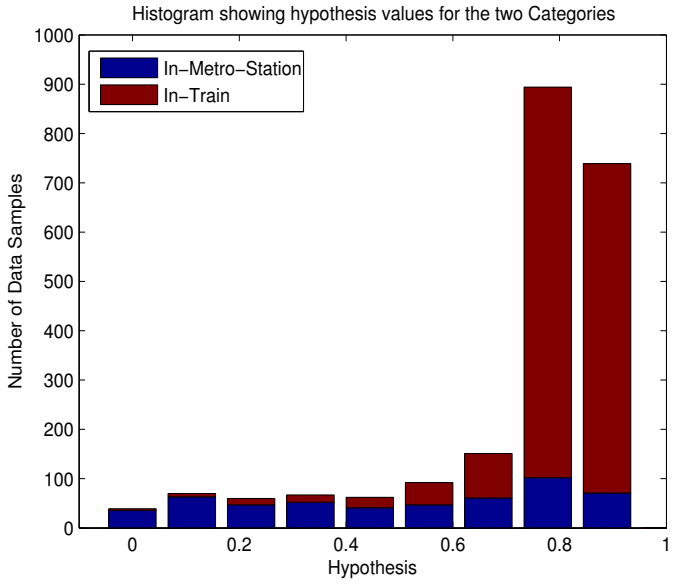

Figure 4.4: Histogram illustrating Overlap among hypothesis values across the two classes for ECDF Representation

5. Temporal Auto-correlation

We conducted a small experiment to study the effects of temporal auto-correlation on the time-series data. Formally, temporal auto-correlation refers to the correlation of a time signal with itself. It studies the similarity or relation between observations as a function of time lag between them. The equation **??** gives the mathematical formulation of temporal auto-correlation where k is the lag value at which to study the temporal auto-correlation.

$$r_k = \frac{\sum_{i=1}^{n-k}(x_i - \mu)(x_{i+k} - \mu)}{\sum_{i=1}^{n}(x_i - \mu)^2} \tag{5.1}$$

The motive behind the experiment was to justify the usage of a windowed approach for analyzing the time-series accelerometer data. Through this exercise, we attempt to claim that there is little or no information loss by considering time-series data as a set of windows or frames.

5.1 Experimental Observations

Lag(sec)	Mean r (trace)	Mean r (in-metro-station)	Mean r (in-Train)
3	0.3932	0.4892	0.5352
4	0.343	0.4455	0.4891
5	0.3338	0.3891	0.4467

Table 5.1: Behavior of Temporal Auto-correlation with the Lag value

Studying the table 5.1, we make some interesting observations about the accelerometer data at hand which are listed as follows:

1. The value of temporal auto-correlation decreases with increase in the lag value which typically is the behavior we expect.

2. "In-train" samples show a higher auto-correlation on an average which can be attributed to the uniform motion of the metro train.

3. "In-metro-station" samples have comparatively lesser auto-correlation which can be attributed to the atypical motion patterns like walking, sprinting, dropping/using the phone, etc.

From this experiment, however, we may not be able to draw a relationship between the optimal window size and effect of temporal auto-correlation.

Conclusion

This study explored the potential of accelerometer data for delineating the activity of traveling in a metro train from other typical non-train related motion patterns, such as waiting in a queue, climbing stairs, etc in the station. This problem was modeled as a two-class classification. Our thesis work investigated the challenges of class imbalance and composite nature of the classes, i.e., each class could be made of multiple motion patterns, present in the accelerometer data. Our results indicate that ensemble based techniques, such as Random Decision Forests and Kernel-SVM with Ensemble perform the best, especially if the data is represented using ECDF features. The lessons learned can be applied in detecting other activities and transportation modes using smartphone-based accelerometer. Our work has applications in smart transportation and smart healthcare.

Bibliography

[1] "Crunching commuter data to track changing communities," 2012. [Online]. Available: https://www.newscientist.com/article/mg21428605-400-crunch-commuter-data-to-track-changing-communities/

[2] J. L. E. K. S. Lonardi and P. Patel, "Finding motifs in time series," in *Proc. of the 2nd Workshop on Temporal Data Mining*, 2002, pp. 53–68.

[3] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proc. of the 11th ACM Conf. on Embedded Networked Sensor Systems*. ACM, 2013, p. 13.

[4] H. Xia, Y. Qiao, J. Jian, and Y. Chang, "Using smart phone sensors to detect transportation modes," *Sensors*, vol. 14, no. 11, pp. 20 843–20 865, 2014.

[5] T. S. Prentow, H. Blunck, M. B. Kjærgaard, and A. Stisen, "Towards indoor transportation mode detection using mobile sensing," in *Mobile Computing, Applications, and Services*. Springer, 2015, pp. 259–279.

[6] N. Y. Hammerla, R. Kirkham, P. Andras, and T. Ploetz, "On preserving statistical characteristics of accelerometry data using their empirical cumulative distribution," in *Proc. of the 2013 Intl. Symposium on Wearable Computers*. ACM, 2013, pp. 65–68.

[7] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson, "Cooperative transit tracking using smart-phones," in *Proc. of the 8th ACM Conf. on Embedded Networked Sensor Systems*. ACM, 2010, pp. 85–98.

[8] T. Higuchi, H. Yamaguchi, and T. Higashino, "Tracking motion context of railway passengers by fusion of low-power sensors in mobile devices," in *Proc. of the 2015 ACM Intl. Symp. on Wearable Computers*. ACM, 2015, pp. 163–170.

[9] A. K. Jain, R. P. Duin, and J. Mao, "Statistical pattern recognition: A review," *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, vol. 22, no. 1, pp. 4–37, 2000.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] V. Palade, "Class imbalance learning methods for support vector machines," 2013.

[12] C. Chen, A. Liaw, and L. Breiman, "Using random forest to learn imbalanced data," *University of California, Berkeley*, 2004.

[13] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Trans. on Sensor Networks (TOSN)*, vol. 6, no. 2, p. 13, 2010.

[14] T. Plötz, N. Y. Hammerla, and P. Olivier, "Feature learning for activity recognition in ubiquitous computing," in *IJCAI Proc. of Intl. Joint Conf. on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1729.

[15] H. Park, "An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain," *Journal of Korean Academy of Nursing*, vol. 43, no. 2, pp. 154–164, 2013.

[16] G. Lee and D. Han, "Subway train stop detection using magnetometer sensing data," in *Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on*. IEEE, 2014, pp. 766–769.

[17] C. M. Bishop, "Pattern recognition," *Machine Learning*.

List of Figures

List of Tables