INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD • INDIA

Real Time Location Prediction with Taxi-GPS Data Streams

Arnab Kumar Laha

Sayan Putatunda

W.P. No. 2017-03-02

March 2017

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

Real Time Location Prediction with Taxi-GPS Data Streams

Arnab Kumar Laha

Sayan Putatunda

Production & Quantitative Methods Area
Indian Institute of Management Ahmedabad
arnab@iima.ac.in
sayanp@iima.ac.in

Abstract

The prediction of the destination location at the time of pickup is an important problem with potential for substantial impact on the efficiency of a GPS enabled taxi service. While this problem has been explored earlier in the batch data set-up, we propose in this paper new solutions in the streaming data set-up. We examine four incremental learning methods using a Damped window model namely, Multivariate multiple regression, Spherical-spherical regression, Randomized spherical K-NN regression and an Ensemble of these methods for their effectiveness in solving the destination prediction problem. The performance of these methods on several large datasets are evaluated using suitably chosen metrics and they were also compared with some other existing methods. The Multivariate multiple regression method and the Ensemble of the three methods are found to be the two best performers. The next pickup location problem is also considered and the aforementioned methods are examined for their suitability using real world datasets. As in the case of destination prediction problem, here also we find that the Multivariate multiple regression method and the Ensemble of the three methods gives better performance than the rest.

Keywords: Directional Data Analysis, Incremental Learning, Intelligent Transportation Systems, Multivariate Multiple Regression, Sliding Windows, Streaming Data

1 Introduction

In recent times, across the world we have seen a spurt in the usage of GPS-based taxi (interchangeably also called cabs in this paper) services viz. Uber, Lyft, Ola, Didi etc. For GPS enabled taxis it is possible to continuously collect the geo-spatial location data for every trip. These recordings are often referred to as GPS traces. This is rich source of data which can give insights on passenger demand and their mobility patterns. The GPS traces generated by a vehicle is a rich source of streaming data. Streaming data is often subject to concept drift as discussed in Section 2.1 below. Some scenarios where this concept drift can be clearly visualized are in case of an accident or some other event leading to congestion in an usually not so busy conference center or an unexpected weather change such as heavy rain, storm or snow leading to blockage of some routes (Moreira-Matias et al., 2016b).

Geo-spatial data streams have many applications in the passenger transportation industry. The intelligent transportation systems literature is replete with various applications viz. traffic monitoring (Herring et al., 2010), passenger finding (Veloso et al., 2011), vacant taxi finding(Phithakkitnukoon et al., 2010), hotspots identification (Chang et al., 2010), trajectory mapping (Liu et al., 2012), etc. where GPS traces have been instrumental in finding interesting insights (see Chen (2014) for more details).

Generally, some of the most important questions for a transport dispatch system as elucidated in Moreira-Matias et al. (2016b) are (a) What's the destination passengers are traveling to i.e. where the vehicle would be vacant?, (b) Travel Time Estimation i.e. how long the vehicle would be occupied? and (c) What is the demand at a particular location in the time interval t? (Liu et al., 2009, Mendes-Moreira et al., 2012, Moreira-Matias et al., 2013a, 2014). The answers to the above questions give insights on the transportation system properties and the passenger mobility. The knowledge of a trip destination before the passenger boards the vehicle can help the transport dispatch system in its operational planning. Also, the knowledge of the next pickup location of a vehicle can help the transport dispatch system in guiding the taxi drivers to find their next passengers in a systematic and efficient manner. These problems are well explored in the literature but mostly in the batch learning set-up (Gandhi, 2015, Veloso et al., 2011) (see section 3 for more details). But analyzing the GPS trace data in streaming data context enables a transport dispatch system to take decisions in real time.

In this paper, we are interested in the real time destination / next pickup location prediction problem. The analysis of GPS data streams of taxis or any public transport for real time prediction opens up new research opportunities for improving the reliability of a transport dispatch system such as introduction of real time decision models to support "operational control" (Moreira-Matias et al., 2016a). Eberlein et al. (1999) gives more details about the various models to solve real-time transit operational control problems using real-time vehicle location information.

Real-time location prediction can have other interesting applications for the transport dispatch system such as in vehicle allocation for the future rides (Powell, 1986, 1987), diversion in real time (Ichoua et al., 2000), ride-sharing (Tran, 2015) and reduction of the total idle time (Miao et al., 2015). Powell (1986) is a key reference on the dynamic vehicle allocation problem which has seen many extensions and applications over the years. In recent years, we have seen that several on-demand transportation services such as Uber (i.e. UberPool), Ola (i.e. OLA Share), etc. provide ride-sharing as well. Tran (2015) worked on the real-time ride-sharing schedule and dispatch problem by using the location information of the passengers who requested a pickup and the location of drivers in the near-by region. Ichoua et al. (2000) worked on diversion issues in real-time vehicle dispatching system. Miao et al. (2015) worked on the taxi dispatch problem using real-time sensor information to reduce the taxi idle driving time. Overall, real-time location prediction helps the transport dispatch system in increasing the reliability and efficiency of their services which finally leads to an increase in profits.

The cost of collection of mobility trace data in real-time is decreasing with the advancement of technology and nowadays, the availability of such data is increasing. Also, it is expected that in near future we would be easily able to collect real-time mobility trace data from multiple sources such as taxis, buses, individual smart-phones, etc. (Moreira-Matias et al., 2016b) The analysis of these data streams offer a great opportunity for development of new methodologies that have applications in the area of Intelligent Transportation Systems as mentioned above and this is our primary motivation for exploring newer methods for solving the destination / next pickup prediction problem.

In this paper we discuss four new methods which to the best of our knowledge have never been applied for solving the destination / next-pickup location prediction problem in a streaming data context. One of these methods has its origin in the literature on directional data analysis, another one is an adaptation of a popular machine learning algorithm for analysis of streaming spherical data, the third one is an adaptation of a multivariate statistics method which has been previously implemented only with static (a.k.a. batch) data, and finally we build an ensemble using the above mentioned three methods. We give details of these methods in Section 5. An extensive performance study of these methods is carried out with five real world datasets and recommendation for use is also given. Our contribution is not only in development of new approaches for the real-time destination / next-pickup location prediction problem but also in development of a framework for selection of suitable methods for different scenarios.

The paper is structured as follows. Section 2 gives a backgound of various concepts that we will be using in this paper. This is followed by a brief review of literature in Section 3. Section 4 describes the Destination Prediction problem statement. Section 5 discusses the methodology and Section 6 describes the datasets used in this paper. Section 7 discusses the results of the various experiments conducted. Section 9 presents the Next Pickup Prediction problem. Finally, Section 10 concludes the paper.

2 Background

In this section, we give a brief overview of streaming data, Spherical data analysis, K-nearest neighbor regression, Multivariate Multiple Regression and Ensemble learning framework. In Section 2.1, we discuss the concept of streaming data and the data analytic challenges emanating from the same. In Section 2.2, we have given a brief overview of Spherical data analysis with particular reference to spherical spherical regression (see Section 2.3). Then we discuss KNN regression, Multivariate multiple regression and Ensemble methods in Sections 2.4, 2.5 and 2.6 respectively.

2.1 Streaming Data

In recent years development of various devices has made collection of continuously flowing data possible. Advances in data storage technology and plummeting cost of data storage has enabled organizations to store such data for possible future use. Streaming data (a.k.a. Data Streams) can be defined as continuous flow of data from a source that arrives at a very high speed (Aggarwal, 2007). The input stream $s1, s2, s3, \ldots$ arrives in a sequential order. In some situations there may be multiple streams of input data (Muthukrishnan, 2003). Some of the major sources of streaming data are operational monitoring using sensors, online advertising, customer clickstreams, GPS data stream, mobile voice and data traffic, social networks and machine to machine communication. The growing scope of Internet of things (IoT) is likely to expand streaming data sources and volumes greatly in the coming years. Streaming data analysis can benefit decision making in multiple domains viz. web mining, network monitoring, high frequency finance and remote monitoring of machines (Ellis, 2014).

The traditional methods of analysis with static data assumes that the full data is available for analysis and multiple passes over the full dataset are possible. However both these assumptions are not true in the streaming data context. At no point of time we can have the full data and multiple passes over the dataset are not possible. Thus the well known methods of data analysis for static data does not apply directly to streaming data. Some of the major challenges of analyzing streaming data identified by Aggarwal (2007),Gama (2010) and Babcock et al. (2002) are (a) the data needs to be processed in one pass since multiple passes are not feasible, (b) the distribution of the input streaming data may change i.e. it may evolve over time (*Concept Drift*) and algorithms for streaming data need to be designed to take care of this and (c) high speed of the streaming data with concept drift requires that the analysis is done within a very short time for that to be of any use in decision making. Thus, Stream data analysis algorithms are required to update the model parameters frequently, discard the older data points according to some predefined scheme and should be able to deal with concept drift.

2.2 Spherical Data

Directional data analysis is an area of statistics in which the observations are in the form of directions. These directions can either be in two, three or higher dimensions. Directional data in two dimensions can be viewed as points on a unit circle and hence such type of data are often referred to as *circular data*. Similarly, directional data in three dimensions can be viewed as points on an unit sphere and hence the name *spherical data* (Jammalamadaka and Sengupta, 2001, Mardia and Jupp, 2000). Such data arise in a variety of contexts such as in geological studies of paleomagnetism in rocks (Mardia and Jupp, 2000), wildlife tracking using GPS data (Urbano and

Cagnacci, 2014), understanding structure of proteins in bioinformatics (Oldfield and Hubbard, 1994), etc.

Let $v^T = (v_1, v_2, v_3)$ be a unit random vector that is taking values on the surface of a unit sphere S^2 centred at the origin. This vector v can be viewed as a vector comprising direction cosines i.e. $v^T = (\cos\alpha, \cos\beta, \cos\gamma)$ where α, β and γ are the angles made by v with the X, Y and Z axes in three dimensional space respectively. Analysis of spherical data is substantially different from that of conventional linear data. Even the basic descriptive analysis of spherical data is different from that of trivariate linear data. For example the spherical mean of n spherical data points $x_1, \ldots, x_n$ is $\bar{x}_0 = \frac{\bar{x}}{||\bar{x}||}$ and not simply $\bar{x}$ which is the case with trivariate linear data(Mardia and Jupp, 2000). A detailed account of statistical analysis of spherical data can be found in Fisher et al. (1993) and Mardia and Jupp (2000). All the currently available methods are with static data i.e. in the batch learning setting. These methods cannot be applied directly when working with streaming data. Adaptation of these methods to the streaming data context has not been previously reported in the literature to the best of our knowledge.

2.3 Spherical-Spherical Regression

Spherical-spherical regression is an extension of the linear regression modeling idea to the case when both predictor and response variables are spherical. The objective is to describe the relationship between the predictor and response variables when both are spherical random variables i.e. these take values on a unit sphere Mardia and Jupp (2000). Some practical applications of this method that have been reported in different areas are: plate tectonic data analysis (Chang, 1986) and spatial rock magnetism data analysis (Stephens, 1979) in geology, Vectorcardiogram data analysis in medicine (Downs, 2003), orientation relationship data analysis in Crystallography in (Mackenzie, 1957).

The simplest spherical-spherical regression model introduced by Chang (1986) is of the form

$$y = Ax \tag{1}$$

where $x, y \in S^2$, and $A \in SO(3)$ where $SO(3)$ is the set of all 3×3 orthogonal matrices with $det(A) = 1$. The matrix A is unknown and has to be estimated from the data. This method is also known as the "Rigid rotation" method. In

Section 5.2 of this paper, we have extended this method to the streaming data scenario using a Damped window model and have also applied the same on a real-world problem arising in intelligent transportation system operations as discussed in Section 7.

2.4 KNN Regression

The k nearest neighbor (k- NN) algorithm is an extension of the nearest neighbor rule developed by Cover and Hart (1967) which is a simple non parametric classification technique. The basic idea behind classification using the nearest neighbor rule is that a new observation ($\mathcal{T}$) is classified based on the category of its nearest neighbor which is identified using a distance metric d. Formally if there are n training pairs $(x_1, \theta_1), \ldots, (x_n, \theta_n)$ where x_j is of dimension p and θ_j indicates the class category of x_j. Now suppose that $\mathcal{T}$ is a test pattern whose class category is not known. Now, if

$$d(\mathcal{T}, x_u) = min\{d(\mathcal{T}, x_j)\} \tag{2}$$

where $j = 1, \ldots, n$, then $\mathcal{T}$ is assigned class θ_u (Cover and Hart, 1967, Murty and Devi, 2011).

It has been theoretically proved by Cover and Hart (1967) that when large samples are involved, the asymptotic error rate of the one nearest neighbor rule is bounded above by twice the bayes error rate. When $k > 1$ for a new observation ($\mathcal{T}$) k-NN algorithm creates a set $S_{\mathcal{T}}$ consisting of k observations from the training set which are nearest to $\mathcal{T}$ as per the distance metric d. The class assignment for $\mathcal{T}$ can be done using a 'voting' technique i.e. the class category that occurs most frequently in $S_{\mathcal{T}}$ is assigned to $\mathcal{T}$. Ties may be broken by choosing one of the tied classes at random. This same method can be used in a regression set-up by predicting the value of the response for a given test instance by averaging the values of the response for the observations in $S_{\mathcal{T}}$. This method is referred to as k-NN Regression (Navot et al., 2006).

The choice of the distance metric can greatly affect the performance of the KNN model and the choice of the same should be taken in cognizance of the application in hand. The most commonly used distance metric is the Euclidean distance. Alternative distance metrics that have been considered in the literature include Manhattan distance, Chebychev distance, Mahalanobis distance and many more (Weinberger and Saul, 2009). In Section 5.3 of this paper we have used the *spherical distance* as the distance metric. In this context it may be mentioned that Lunga and Ersoy (2010) have explored the idea of spherical nearest neighbor classification using a different distance metric.

Moreover, the K-NN method is computationally very expensive for large sized training datasets. Since the whole of training dataset is being used for finding similarity with the test instance, so for n training patterns with p dimensions, the time complexity of the K-NN algorithm is $\mathcal{O}(np)$ (Kusner et al., 2014). So one way to increase the computational speed is to reduce the size of the training dataset without impacting the accuracy of the algorithm and also by using efficient algorithms (Murty and Devi, 2011). Some methods for dealing with the slow computational speed of the K-NN algorithm are the branch and bound technique (Fukunaga and Narendra, 1975, Miclet and M.Dabouz, 1983), the cube algorithm (Yunck, 1976), the projection algorithm (Friedman et al., 1975), ordered partitions (Kim and Park, 1986) and cluster based trees (Zhang and Srihari, 2004). Sometimes even specialized data structures like k-d trees (Shakhnarovich et al., 2005) and hashing methods (Papadopoulos and Manolopoulos, 2005) are used to increase the computation speed. Also, there have been some works on online / incremental K-NN learners for handling large data streams (Bosnic et al., 2011, Förster et al., 2010). In section 5.4, we propose a modified k-NN regression method that is much faster than conventional k-NN regression and have an almost similar predictive accuracy.

2.5 Multivariate Multiple Regression

The Multivariate multiple regression model can be seen as a generalization of the multiple linear regression model where instead of a single response variable, we have several response variables. We model the relationship between multiple responses $(Y_1, \ldots, Y_m)$ and the predictors $(x_1, \ldots, x_p)$. Each of these responses $Y_1, \ldots, Y_m$ is assumed to follow its own regression model as shown in Equation 3 (Johnson and Wichern, 2007). It may be noted that it is possible for the multiple responses $(Y_1, \ldots, Y_m)$ to be correlated among themselves.

$$\begin{aligned}
Y_1 &= \beta_{01} + \beta_{11}x_1 + \ldots + \beta_{p1}x_p + \epsilon_1 \\
Y_2 &= \beta_{02} + \beta_{12}x_1 + \ldots + \beta_{p2}x_p + \epsilon_2 \\
&\vdots \\
Y_m &= \beta_{0m} + \beta_{1m}x_1 + \ldots + \beta_{pm}x_p + \epsilon_m
\end{aligned} \tag{3}$$

where, the $\epsilon^\top = [\epsilon_1, \ldots, \epsilon_m]$ has $E(\epsilon) = 0$ and $Var(\epsilon) = \Sigma$. So, the error terms associated with different responses may be correlated (Johnson and Wichern, 2007).

Let, $[x_{i0}, x_{i1}, \ldots, x_{ip}]$ be the values of the predictor or independent variables for the i^{th} trial. And let, $Y_i^\top = [Y_{i1}, Y_{i2}, \ldots, Y_{im}]$ be the response or target variables and let $\epsilon_i^\top = [\epsilon_{i1}, \epsilon_{i2}, \ldots, \epsilon_{im}]$. The multivariate multiple regression model as described in Johnson and Wichern (2007) is given below.

$$\underset{(n\times m)}{Y} = \underset{(n\times(p+1))}{X} \underset{((p+1)\times m)}{\beta} + \underset{(n\times m)}{\epsilon} \tag{4}$$

where, X is the design matrix, $\underset{(n\times m)}{\epsilon} = \left[\epsilon_{[1]} \vdots \epsilon_{[2]} \vdots \ldots \vdots \epsilon_{[m]}\right]$ and $E(\epsilon_{[i]} = 0)$ and $Cov(\epsilon_{[i]}, \epsilon_{[k]}) = \sigma_{ik}I$, where $i, k = 1, \ldots, m$. Simply put, the j^{th} response $Y_{[j]}$ follows the linear regression model

$$Y_{[j]} = X\beta_{[j]} + \epsilon_{[j]}, \quad j = 1, \ldots, m \tag{5}$$

where $Cov(\epsilon_{[j]}) = \sigma_{jj}I$ (Johnson and Wichern, 2007). And the least squares estimate is as follows.

$$\widehat{\beta} = (X'X)^{-1}X'Y \tag{6}$$

2.6 Ensemble Methods

Ensemble learning is a widely used technique in machine learning and statistics and has been used in both classification and regression problems. The intuition behind using the ensemble method is that often (though not guaranteed) it performs better than the individual base models. The goal of the ensemble learning approach is to combine various methods for improving the predictions for out-of-sample data. The basic idea here is to take the predictions from the various alternative models and combine them (Hastie et al., 2009). Over the years various technique have been developed for creating ensembles e.g- Bagging(Breiman, 1996), Boosting (Freund and Schapire, 1997, Schapire, 1990), Random Forests (Breiman, 2001) and Stacking (Wolpert, 1992).

In this paper, we focus on multi-model ensembles. The simplest way to create such an ensemble is to use a variety of regression models on the training data and then combine their predictions using an averaging scheme. Among the various combination methods, the simplest to implement is the averaging rule since it needs no prior training (Hjort and Claesken, 2003, Kotsiantis and Pintelas, 2005). The other combination techniques include weighted average and Bayesian Model Averaging (BMA) (Hjort and Claesken, 2003).

2.7 Stochastic Dominance

While comparing predictive performance of the different methods considered in this paper we have used the notion of stochastic dominance of the error distributions. We briefly introduce the concept below. For more details the reader may look at Davidson (2008).

Let P and Q be two distributions having cumulative distribution functions (CDFs) F_P and F_Q respectively. The distribution Q is said to be stochastically dominant over distribution P at first order, if for every x, $F_P(x) \geq F_Q(x)$ (Davidson, 2008). Graphically the stochastic dominance of the first order can be visualized by examining the plots of the CDFs for the two distributions P and Q together. If P is stochastically dominant over Q of the first order then the CDF of P is to the right of that of Q and they do not cross each other. Since in many situations, as in this paper, the CDFs P and Q are not known they need to be estimated from the data. Since by the Dvoretzky-Kiefer-Wolfowitz inequality (Wasserman, 2010, p. 99) we know that the empirical cumulative distribution function (ECDF) approximates the CDF very well when sample size is large, we use the ECDF for graphically checking the stochastic dominance of first order in this paper in Sections 7, 8 and 9.

3 Related Work

Analysis of GPS trace data is a challenging problem with lots of applications in the transportation domain. Most of the currently available methods use batch learning methods (Gambs and Killijian, 2012, Krumm, 2008, Krumm and Horvitz, 2006, Li et al., 2012, Simmons et al., 2006). It's only recently, that some papers have taken into account the streaming nature of GPS trace data where online and incremental learning methods for solving some specific problems have been proposed (Lam et al., 2015, Moreira-Matias et al., 2013b, 2016b, Sun et al., 2012).

Predicting destination from partial trajectories is a problem that has tremendous potential of real world applications. Several solutions of this problem from different perspectives has been proposed in the literature using Origin-Destination (OD) matrix (Barceló et al., 2010, Jin et al., 2008, Moreira-Matias et al., 2016b, Park et al., 2014, Yue et al., 2009, Zhang et al., 2011), Bayesian inference (Hazelton, 2008, Parry and Hazelton, 2012), Support Vector Machine (SVM) (Li et al., 2011), Clustering (Chang et al., 2010, Li et al., 2012), Mobility markov chains (Gambs and Killijian, 2012), Optimization (Miao et al., 2015), ensemble learning (Lam et al., 2015) and many more. In this section, we have given a brief literature review of some of the works done in this field.

Krumm and Horvitz (2006) implemented a method termed *predestination* that predicts where the driver is going on the go. A Bayesian inference model is built that uses driving behavior data along with GPS data. Other papers that have used Bayesian inference for solving similar problems are Marmasse and Schmandt (2002) and Liao et al. (2004). Gambs and Killijian (2012) discusses the next place prediction problem using the information of coordinates of visited places applying the n-MMC (Mobility Markov Chain) algorithm. n-MMC is a modified version of the Mobility Markov Chain model where they keep a track of n-previous locations visited.

Simmons et al. (2006) used a hidden markov model (HMM) for prediction of route and intended destination. Markov models have also been used in other studies for making short term route predictions (Krumm, 2008). Xue et al. (2013) proposed the sub-trajectory synthesis method for destination prediction. Chen et al. (2011) worked on extracting route pattern from user's personal trajectory data using a probabilistic model which they termed as "Continuous Route Pattern Mining (CRPM)".

In recent years, some work has been reported in the literature which have given emphasis to the streaming nature of the GPS traces data and has shifted the focus towards real time or near real time prediction. Moreira-Matias et al. (2013b) worked on predicting the passenger demand in a streaming data context and proposed a ensemble method with sliding window technique. Sun et al. (2012) worked on taxi trajectory anomaly detection in real time from GPS data streams. A *nearest neighbor technique (NNT)* was proposed by Tiesyte and Jensen (2008), where they used euclidean distance as a distance measure for travel time prediction of vehicles. Moreira-Matias et al. (2016a) worked on eliminating bus bunching in real time using an online learning approach.

The destination prediction problem in streaming data context has not been explored much in the literature. Lam et al. (2015) worked on real time prediction of destination and travel time estimation from given partial trajectories using an ensemble learning model, while Persad-Maharaj et al. (2008) used a geometric representation approach to predict an individual‘s travel path and destination in real time. In Lam et al. (2015), for the destination prediction problem, the authors first used Kernel regression (KR) for feature extraction and then the latitude and the longitude of the destination was obtained independently by using Random Forests (RF). They also experimented with Support Vector Regression (SVR).

The next pickup prediction problem is well explored in the literature in the batch learning set-up. The next pickup prediction problem is a problem of “Operational dynamics” which can be grouped under Passenger/ Taxi-Finding Strategies as in Castro et al. (2013). A common approach taken by most of the papers in the literature for these kind of problems is to first extracting hotspots i.e. high demand zones using cluster analysis and then perform further analysis on these clusters (Chang et al., 2008, Liu et al., 2010, Palma et al., 2008). Li et al. (2011) used L1 Norm SVM to determine what course of action the driver should take using both time and location information. Palma et al. (2008) used the density based clustering algorithm (DBSCAN) for speed based clustering of trajectories.

Hu et al. (2012) worked on a tree based recommendation system that helps taxi drivers in selecting suitable routes for picking up passengers. Veloso et al. (2011) explored the relationship between pickup and drop-off locations and analyze the behavior between the previous drop-off location and the following pickup location. They also performed a predictability analysis for next pickup area type given the current drop-off location information. Gandhi (2015) worked on predicting driver's next pickup location given various factors such as time of the day, weather information, etc. using Artificial Neural Networks (ANN). While most of the work done in the literature focuses on the batch learning set-up but in this paper we concentrate on the streaming data set-up.

4 Dynamic Location Prediction

Prediction of final destination of a vehicle while in service based on the location of the pickup point can improve the efficiency of a transport dispatch system through improved vehicle allocation for future pickups. Thus, in this paper we focus on predicting the drop-off location coordinates of a trip taking the pickup location coordinates as input. Since there is substantial variation in passenger movements depending on variety of factors such as time of the day, weekday or weekend, special events etc. a static model is unlikely to perform well over time than a dynamic model which uses the most relevant recent information is required. It is in this context we attempt to develop dynamic models which use the recent information and perform prediction in a user defined prediction horizon. A similar problem arises when we are interested in predicting the next pickup location for a cab given the dropoff location coordinates of the previous trip as input (see Section 9). Thus we refer to our problem as the *Dynamic Drop / Next-pickup Prediction Problem (DDNPP)*.

We focus on building an incremental learning algorithm for this problem. Predictive models for data streams need to adapt to the evolving data over time i.e. it needs to handle the inherent concept drift in the data streams. Otherwise, the predictive accuracy of the model decreases. Also, the model parameters need to be updated by taking into account the recent information in the data and the predictive model should ideally use a fixed amount of memory for any storage (Gama et al., 2014). We need to use either an online or an incremental learning algorithms for modeling data streams. Incremental learning algorithms differ from online learning algorithms in that the model parameters are updated in batches instead of whenever a new observation arrives.

The most common approach for handling evolving data over time is by forgetting or discarding the older observations (Gama et al., 2014). One way to do it is by using a gradual forgetting mechanism where weights are assigned to the observations and a fading function is used to discard the older observations. The fading function can be either linear decay (Koychev, 2000) or exponential decay (Klinkenberg, 2004). The forgetting mechanism is implemented using a windowing technique which is a powerful tool for handling concept drift. An incremental learning algorithm can be approximated by using a non-incremental learner along with a sliding window (Büttcher et al., 2010, p. 337). Hence, we use a learner $\mathcal{L}$ which is fed a sequence of points s_1, s_2, ... , s_n using a windowing technique (see Section 5.1 for more details). Note that n may vary from window to window.

5 Methodology

We use the four different methods, namely Spherical-Spherical Regression-SSR (see section 5.2), Spherical K-NN Regression- SKNNR (see section 5.3) , Randomized Spherical K-NN Regression- RSKNNR (see section 5.4) and Multivariate Multiple Regression- MMR (see section 5.5) as learners along with a sliding window with exponentially fading strategy applied on the incoming data stream (to be called Damped window model in section 5.1) to develop four different incremental learning algorithms. We also build an ensemble method (see Section 5.6). It may be noted that the data stream consists of a sequence of pickup and drop-off location coordinates and their timestamps.

5.1 Damped Window Model

A window can be defined as a snapshot of data - either observation count-based or time based (Gama, 2010). This is a very useful technique in streaming data context since at no point in time we will have the "entire data" available with us. Further, windowing is a very powerful tool when it comes to dealing with *concept drift* which is a major challenge in streaming data analysis as discussed earlier. There are various types of windowing techniques such as damped window model, landmark window and sliding windows discussed in Zhu and Sasha (2002), Chang and Lee (2004) and Li et al. (2009).

In case of Landmark windows, there is no discarding of older data points and all the data points are accumulated in the window. We don 't consider this suitable in a streaming data context since this is computationally expensive particularly with fast streaming data. The other windowing technique is the Sliding windows. *Sliding windows* are one of the popular ways of discarding the older data points and considering only the recent data points for analysis (Aggarwal, 2007). In case of a *Damped Window Model*, the window length is time-based and an exponential fading strategy is used to discard the old data. We adopt this model in our problem since very old data has little relevance for taking a decision at the current time point.

A data window is defined to contain all the data collected during a specified unit of time interval. The latest data window is numbered 1, the one before the latest is numbered 2 and so on. We assign weights to the data windows and these decrease exponentially over time as discussed below. The t^{th} window is assigned the weight $h(t)$ where

$$h(t) = 1/2^{\lambda t},\ where\ \lambda > 0, t = 1, 2, \ldots \tag{7}$$

All data windows which have weights less than a preassigned value c are discarded in the sense that these data points are not used by the learners. In this paper we have used $c = 0.09$. It may be noted that by changing the value of λ, the decision regarding the discarding of the older data can be influenced. The more the value of λ, the less importance is given to the older data compared to more recent data (Cao et al., 2006). Once the length of the window is determined using the damping function mentioned in Equation 7, we treat this as a mini batch and proceed.Therefore, this may also be called a "Mini-batch Window Model" as it has been done in Putatunda (2017).

5.2 Spherical Spherical Regression (Rigid Rotation) with Damped Window Model

The SSR method is an extension of the Spherical-Spherical regression method discussed in Section 2.3 where the model parameters are estimated by using only the retained data as discussed in Section 5.1 above. The estimated parameters are then used to predict the pickup/ drop-off locations of the new data arriving in the prediction horizon. The model parameters are updated whenever a new prediction horizon is chosen thus enabling the model to handle the *concept drift*.

5.3 Spherical K-NN Regression with Damped Window Model

The SKNNR method uses the *Spherical distance* as the distance metric of the K-NN regression as discussed in Section 2.4. The spherical distance between two points u and v on the surface of a sphere is the shortest route along the surface from u to v. Mathematically, it is defined as

$$d = \arccos(u \cdot v) \tag{8}$$

where $u \cdot v$ represents the dot product (Ratcliffe, 2006). We have implemented the K-NN regression with the above spherical distance metric and the damped window model.

The KNNRSD algorithm is described below in Algorithm 1. Since the value of k is chosen from an array given to the user in some cases especially for very sparse datasets we may have training windows, the size of which is less than our chosen value of k. In such cases the we need to modify the choice of k for that window to ensure that the algorithm doesn't terminate. For this window, we take the value of k to be an integer lower than the training window size n. For example, suppose we want to run the k-NN regrssion with spherical distance with $k = 50$ but the number of observations in the training window is say 49, then the algorithm chooses the next value of k from the array which should be lower than 49 i.e. $k = 25$.

Algorithm 1 SKNNR Pseudocode

```
Require: $n > 0,\ t > 0,\ k \in [5, 10, 15, 20, 25, 50, 100]\ where, n \rightarrow Training\ Window\ Size,\ t \rightarrow Test\ Window\ Size,\ k \rightarrow no.\ of\ nearest\ neighbors\ (chosen\ from\ the\ array),\ k_{[i]} \rightarrow the\ value\ of\ k\ at\ the\ i^{th}\ position\ in\ the\ array,\ i \rightarrow 1\ to\ length(array\ k)$
  Choose $k^*$ from array $k$ {Let $k^* = k_{[i]}$}
  if $k_{[i]} < n$ then
    Set $k = k_{[i]}$
    Run k-NN Regression {with Spherical distance metric}
  else
    Initialize $j = 1$
    While $k_{[i-j]} > n$
    $j \rightarrow j + 1$
    End While
    Set $k = k_{[i-j]}$
    Run k-NN Regression {with Spherical distance metric}
  end if
```

5.4 Randomized Spherical K-NN Regression with Damped Window Model

In this paper we propose a new method RSKNNR where we first perform a simple random sampling of the training dataset. Since this sample is representative of the whole training dataset we replace the training dataset with this randomly sampled subset and proceed to use the SKNNR algorithm as discussed in Section 5.3. The proportion of the original dataset that needs to be sampled (*sampling rate*) is an important decision variable and later in this paper we do extensive sensitivity analysis to suggest a thumb rule for the same. We observe that this method greatly increases the computation speed without sacrificing the accuracy of the procedure.

It may be noted that sometimes especially for sparse training datasets, the sample size obtained with a sampling rate of $r\%$ may be less than the value of the k nearest neighbors where k is the user defined input. In such cases we run the SKNNR on the whole training data.

Algorithm 2 RSKNNR Pseudocode

```
Require: $n > 0,\ t > 0,\ 0 < r < 1,\ k \in [5, 10, 15, 20, 25, 50, 100],\ s = r \times n,$ $where,\ n \rightarrow Training\ Window\ Size,\ t \rightarrow Test\ Window\ Size,\ r \rightarrow Sampling\ Rate,\ s \rightarrow sample\ size\ after\ performing\ SRS\ with\ sampling\ rate\ r,\ k \rightarrow no.\ of\ nearest\ neighbors\ (chosen\ from\ the\ array)$
  if $s < k$ then
    Run SKNNR (with $training\ window\ size = n$)
  else
    Run SKNNR (with $training\ window\ size = s$)
  end if
```

5.5 Multivariate Multiple Regression with Damped Window Model

An alternative approach for analyzing spherical data is to first flatten the sphere and then apply linear models. Since in our applications we are operating in small regions of sphere and hence we propose the MMR method which proceeds in the manner discussed above. The points on the surface of a unit sphere centered at the origin can be represented either as an unit vector (X, Y, Z) or as angles (φ_1, φ_2) where $(\varphi_1, \varphi_2) \in [0, \pi) \times [0, 2\pi)$. φ_1 and φ_2 are called latitude and longitude respectively. Now,

$$(X, Y, Z) = (\cos\varphi_1, \sin\varphi_1 \cos\varphi_2, \sin\varphi_1 \sin\varphi_2) \tag{9}$$

as defined in Jammalamadaka and Sengupta (2001).

In the MMR method first, both the directional predictor and response variables i.e. the pickup and dropoff latitude and longitude respectively, are converted into Euclidean data using equation (9). Then we build the multivariate multiple regression model as discussed in section 2.5. This model is then used to predict the euclidean dropoff coordinates when the euclidean pickup coordinates of a new pickup is given. Finally, the predicted euclidean dropoff coordinates are converted back to polar coordinates by using equation (9) again.

5.6 Multi-Model Ensemble

We build a multi-model ensemble (discussed in Section 2.6) by combining the predictions of the RSKNNR (with the selected hyperparameters), SSR and MMR. We have taken the drop-off locations (longitude, latitude) predicted by these three methods and converted each of them to euclidean data using equation (9). Then these euclidean dropoff coordinates are averaged coordinatewise to obtain $\mathbf{m} = (\bar{X}, \bar{Y}, \bar{Z})$. Then we convert (X', Y', Z') into a unit vector by dividing it by its norm $||m|| = \sqrt{\bar{X}^2 + \bar{Y}^2 + \bar{Z}^2}$ to ensure that it lies on the unit sphere. Finally we convert $m/||m||$ into polar coordinates (Fisher et al., 1993). We use the output of this ensemble model for predicting the drop-off location. We will refer to this technique as "Ensemble" in the rest of this paper.

6 Data

We use the New York City (NYC) yellow taxi data from 1st January, 2013 to 5th January, 2013 as our primary dataset. This is a publicly available data which can be found at the NYC Taxi and Limousine Commission website (NYC-TLC, 2014). The dataset description and other details related to its pre-processing are given in Section 6.1. Apart from this, we have also tested our methods on other real world datasets which are discussed in Section 6.2.

6.1 NYC Yellow Taxi GPS Data

The NYC Yellow Taxi GPS dataset consists of various attributes related to the taxi pickup and drop-off coordinates, their corresponding timestamps, information related to trip travel time and distance travelled and payment related information. Here, we would be using some of the attributes as given in Table 1. The dataset has been cleaned of missing values and some other anomalies like erroneous GPS coordinate values. The total number of observations in the cleaned dataset for the period 1st to 5th January, 2013 is 824,799. This dataset is used as training dataset for model building. We use this dataset for determining the value of K to be used in SKNNR and also to determine the the sampling rate for the RSKNNR. Further, dataset is used for comparing the performance of the various methods discussed above. We will refer to this dataset as NYC1

Another slice of this dataset for the time period 13th - 16th January, 2013 containing 680,865 observations after cleaning, is also used for comparison of performances of the different methods. We will refer to this dataset as NYC2

Table 1: NYC Yellow Taxi GPS Data - Attribute Description

Attributes	Description
medallion	an unique id of the taxi - vehicle bound
hack_license	an unique id for the taxi license
pickup_datetime	time when the passenger(s) were picked up
dropoff_datetime	time when the passenger(s) were dropped off
pickup_longitude	pickup location's longitude coordinate
pickup_latitude	pickup location's latitude coordinate
dropoff_longitude	drop-off location's longitude coordinate
dropoff_latitude	drop-off location's latitude coordinate
trip_time_in_secs	time taken for the trip
trip_distance	distance travelled in miles

6.2 Other datasets

This section gives a brief description of the various datasets that have been used in this paper for testing purposes.

6.2.1 NYC Boro Taxi GPS Data

The NYC Boro Taxi GPS Data is a publicly available dataset comprising of trip details of Boro taxis or Green taxis (NYC-TLC, 2014). This dataset also contains information related to the taxi pickup and drop off coordinates, their corresponding timestamps, trip distance travelled and payment related information. We use the attributes given in Table 2. For our analysis, we have used have used data from 1st- 7th June, 2014. We will refer to this dataset as BORO.

Table 2: NYC Boro Taxi GPS Data- Attribute Description

Attributes	Description
lpep_pickup_datetime	time when the passenger(s) were picked up
lpep_dropoff_datetime	time when the passenger(s) were dropped off
pickup_longitude	pickup location's longitude coordinate
pickup_latitude	pickup location's latitude coordinate
dropoff_longitude	drop-off location's longitude coordinate
dropoff_latitude	drop-off location's latitude coordinate
passenger_count	no. of passengers that boarded the taxi
trip_distance	distance travelled in miles

6.2.2 Porto GPS Taxi Data

The Porto GPS taxi dataset is publicly available at the UCI Machine Learning Repository (Lichman, 2013) and was first used in Moreira-Matias et al. (2013b). This dataset contains for each trip information regarding taxi stand, call origin details, unique taxi id, unique trip id and the Polyline. The Polyline is a string of GPS coordinates with each coordinate being obtained after every 15 seconds of the trip beginning with the pickup and ending with the dropoff. For our analysis we have taken the trip details for 1st- 7th July, 2013 which consists of 34,768 observations. We obtained the pickup coordinates (start_latitude, start_longitude) and drop-off coordinates (dest_latitude, dest_longitude) from this dataset. We will refer to this dataset as PORTO.

6.2.3 San Francisco black cars GPS traces

This publicly available dataset consists of anonymized GPS traces of Uber black cars in San Francisco for one week (Henry, 2011). For each trip location coordinates are recorded every 4 seconds starting with pickup and ending with drop-off. The first record for each trip gives the pickup time, pickup latitude and longitude and the last records gives the drop-off time, dropoff latitude and longitude. The dataset has 24,552 observations after data cleaning. We will refer to this dataset as SFBLACK.

7 Experimental Results

In this section, we report the findings of the different experiments done on the NYC GPS taxi dataset with the four methods discussed in Section 5 to assess their performance. The performance evaluation metrics are discussed in Section 7.1. The entire data stream flow, model building and data analysis is implemented using the software R version 3.3.1. (R Core Team, 2016). The R packages Directional (Tsagris and Athineou, 2016), lubridate (Grolemund and Wickham, 2011), Imap (Wallace, 2012), nnet (Venables and Ripley, 2002), e1071 (Meyer et al., 2015), randomForest (Liaw and Wiener, 2002) and ggplot2 (Wickham, 2009) has been used for doing our experiments. A sensitivity analysis on the choice of λ, window size and sampling rate (for RSKNNR method) has been carried out and the results are reported in Section 7.3.

7.1 Evaluation Metrics

We use two evaluation metrics for comparing the performance of the four different methods. The geodesic distance between predicted and actual drop off points is calculated using the Vincenty inverse formula (Vincenty, 1975) and is called the Geodesic Distance Error GDE. Geodesic distance can be defined as the shortest path between two points on the surface of the earth (Economou et al., 2004). If there are two points say x and y on the surface of the earth (i.e. a sphere with radius R) then the geosedic distance between these two points is represented by G_{xy}. The radius of earth is assumed to be 6378137 meters and (ϕ_x, μ_x) and (ϕ_y, μ_y) denotes the latitude and longitude of the points x and y. Then the geodesic distance between these two points can be obtained by using the following equation (i.e. equation (10)) as described in Gade (2010).

$$G_{xy} = R \cdot \arccos(\sin\phi_x \cdot \sin\phi_y + \cos\phi_x \cdot \cos\phi_y \cdot \cos(\mu_x - \mu_y)) \qquad (10)$$

In this paper, we use the Vincenty inverse formula for calculating the geodesic distances between points on earth's surface. The Vincenty inverse formula assumes earth to be an ellipsoid (WGS84 coordinates) to calculate the geodesic distance. This is more accurate than the great circle distance methods (Vincenty, 1975). Chang et al. (2010) discusses the Vincenty formula for the computation of the geodesic distance G_{xy} and it is represented in equation (11) below.

$$G_{xy} = R \cdot \arctan\left(\frac{\sqrt{(\cos\phi_y \sin(\mu_x - \mu_y))^2 + (\cos\phi_x \cos\phi_y - \sin\phi_x \cos\phi_y \cos(\mu_x - \mu_y))^2}}{\sin\phi_x \cdot \sin\phi_y + \cos\phi_x \cdot \cos\phi_y \cdot \cos(\mu_x - \mu_y)}\right) \tag{11}$$

The GDE between the actual and the predicted coordinates are calculated using the Imap package (Wallace, 2012). Now for each prediction horizon $\mathcal{H}$, both the mean GDE and median GDE are recorded and are referred to as MGDE and MedGDE respectively.

If $MGDE_1, MGDE_2, \ldots, MGDE_n$ are MGDE recorded for n prediction horizons (assumed to be pairwise disjoint) the Aggregated Mean Geodesic Distance Error
(AMGDE) can be defined as follows.

$$AMGDE = \frac{MGDE_1 + MGDE_2 + \cdots + MGDE_n}{n} \tag{12}$$

Similarly, if $MedGDE_1, MedGDE_2, \ldots, MedGDE_n$ are the MedGDE recorded for n pairwise disjoint prediction horizons then the Aggregated Median Geodesic Distance Error (AMedGDE) can be defined as follows.

$$AMedGDE = \frac{MedGDE_1 + MedGDE_2 + \cdots + MedGDE_n}{n} \tag{13}$$

The $AMGDE$ and $AMedGDE$ are the two evaluation metrics used in this paper for comparing the performance of the different methods. The method with the least value of the chosen evaluation metric is considered the “best” method as per that metric. The units of both of these evaluation metrics are in “miles”.

7.2 Concept Drift

As mentioned in Section 2.1, concept drift refers to the change in the distribution of the streaming data over time. A consequence of the presence of concept drift is that an off-line learning model's predictive performance changes unpredictably (and often seen to deteriote) as we move forward in time. This induces a need to update the model by taking into consideration the recent data and information. In this section, we show that concept drift is present in the NYC1 dataset. We consider a part of the NYC1 dataset (time interval $00:00-\ 01:00$ of 3^{rd} January, 2013) and use it as training data for the off-line learners. We then use these models for drop-off location prediction when the pickup location coordinates are known for the next four 1 hour prediction horizons.

We consider the off-line learners to be Random Forests and Support Vector Regression (see Breiman (2001) and Vapnik (1995) for more details) which we will refer to as *RF_batch* and *SVR_batch* respectively. The reason for choosing these two offline learning methods is that they were used in Lam et al. (2015) and showed good performance for a related problem. We build Random forests with 2000 trees (as done in Lam et al. (2015)) and implement an ϵ-SVR method with a Radial basis function (RBF) kernel. We then plot the mean geodesic distance error (MGDE) for both these methods with respect to the prediction horizons as shown in figure 1. We can see an increase in the MGDE over time for both these methods which is indicative of presence of concept drift. In a similar way we observe the presence of concept drift in the other datasets used in this paper. In this context, it has been noted by Moreira-Matias et al. (2016b) that GPS trace data are inherently subject to concept drift and some scenarios where this concept drift can be clearly visualized are car accidents on a busy road, unexpected weather event such as rain or snow etc.

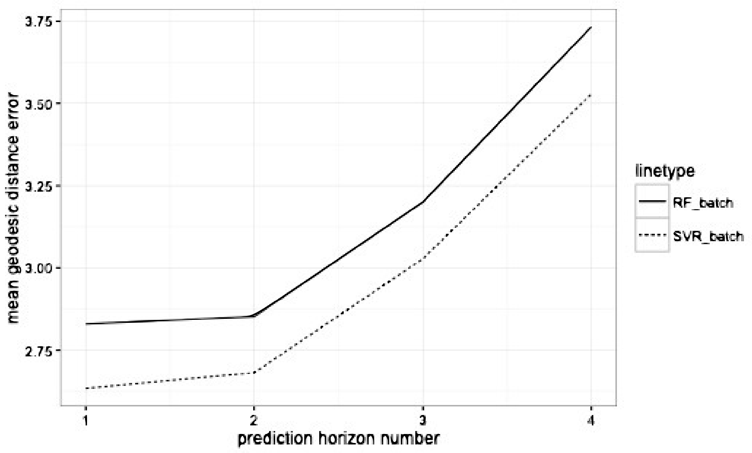


Figure 1: Demonstration of presence of concept drift in taxi GPS data streams on the NYC1 dataset

7.3 Results

In this section we describe the results of the various experiments and sensitivity analyses that were performed. The methods were first applied on the NYC1 dataset and the performance of the different methods are evaluated. Then these methods were applied on the other datasets discussed in Section 6.2 and the findings are discussed in Section 7.4.

In Table 3 below we give the values of AMGDE and AMEDGDE when the SSR method discussed in Section 5.2 is applied on NYC1 dataset. It can be seen that the SSR method does not show dependence on the window size as well as the chosen value of λ. We confirm the above observation by carrying out two way ANOVA with AMGDE as the response variable and the window size and λ value as two factors. We find that both these factors are not significant at 5% level of significance. Thus we take window size as 1 hour and λ as 0.5 for further study. When $\lambda = 0.5$ we have $h(t) < c$ for all $t \geq 7$ and hence only the data contained in the data windows numbered 1 to 6 are used by the learners for prediction purposes. We obtained similar results when we considered AMedGDE as the response variable.

Table 3: Performance of SSR across various λ and Window sizes

Window	λ	AMGDE	AMedGDE
15 mins	0.25	2.7307	1.8897
	0.5	2.7250	1.8888
	0.75	2.7240	1.8916
30 mins	0.25	2.7327	1.8840
	0.5	2.7266	1.8868
	0.75	2.7228	1.8862
1 hour	0.25	2.7274	1.8616
	0.5	2.7293	1.8808
	0.75	2.7286	1.8855

In Table 4 we give the AMGDE and AMedGDE when the MMR method discussed in Section 5.5 is applied to the NYC1 dataset. As with SSR we see that the AMGDE and AMedGDE does not depend on the choice of window size and the value of λ. We confirm this by carrying out two separate ANOVA procedures once with AMGDE and the other with AMedGDE as the response variable. As in the earlier case the factor variables window size and λ did not turn out to be significant at 5% level. Thus we take window size as 1 hour and λ as 0.5 for further study. Further we compare the AMGDE values obtained from the SSR method with those of the MMR method using a two sample paired t-test. The test turns out to be significant at 5% level indicating that the AMGDE from the MMR method is significantly lower than that of the SSR method. We arrive at the same conclusion when considering AMedGDE. Thus we can say that MMR performs better than SSR for this dataset.

Table 4: Performance of MMR method across various λ and Window sizes

Window	λ	AMGDE	AMedGDE
15 mins	0.25	2.2621	1.7258
	0.5	2.2523	1.7154
	0.75	2.2481	1.7106
30 mins	0.25	2.2620	1.7178
	0.5	2.2650	1.7302
	0.75	2.2606	1.7256
1 hour	0.25	2.2458	1.6960
	0.5	2.2609	1.7176
	0.75	2.2671	1.7296

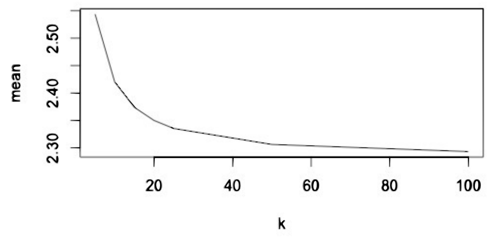
In table 5 we show the performance of the SKNNR method for different values of K, window sizes and λ. We vary K= 5, 10, 15, 20, 25, 50 and 100. It may be noted that while a larger K may increase prediction accuracy a little but it simultaneously increases the computational cost. Thus there is a accuracy-computational cost trade off that needs to be taken into consideration. We perform an ANOVA with AMGDE as the response variable and the window size, K and λ values as three factors. We find that λ is not significant but window size and k are significant at 5% level of significance. We observe that while the mean AMGDE for $k = 100$ is slightly lower than that for $k = 50$ but computationally it is far more expensive. Hence, we choose $k = 50$. For the window size, we can see in Figure 2(b) that there isn't much difference in the mean AMGDE values for three window sizes considered in this experiment. We therefore proceed with a window size of 1 hour and $\lambda = 0.5$. We obtained similar results when we considered AMedGDE as the response variable.

The performance of RSKNNR is given for different sampling rates viz. 5%, 10% and 20%, K and window sizes with λ held fixed at 0.5. As before we vary K= 5, 10, 15, 20, 25, 50 and 100. We perform an ANOVA with AMGDE as the response variable and the window size, K and Sampling rate as three factors. We find that all the three factors are significant at 5% level of significance. In Figure 3, we have shown the main effects plot for each of these factors. For similar reasons ad discussed in the case of SKNNR we choose $k = 50$. For both the window size and sampling rate, we can see that that there is not much difference in the mean AMGDE values for the different levels of these two factors. Thus we take window size as 1 hour and sampling rate as 10 % for further study. We obtained similar results when we considered AMedGDE as the response variable.

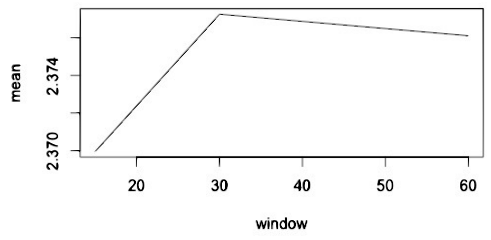
In Table 7 we show the dependence of the time taken by the RSKNNR for different sampling rates. Note that SKNNR can be thought of as a special case of RSKNNR with sampling rate = 100%. The experiments were carried out on a system with 24 GB RAM and Intel Xeon processor 2.67 GHZ with a 64 bit Windows Server 2012 OS. We have recorded the mean time taken in seconds for each of these methods with $K = 50$, $\lambda = 0.5$ and window size as 1 hour. As expected we see that RSKNNR is much faster than SKNNR. Also, the execution speed increases with lowering of sampling rate as expected.

Table 5: Performance of SKNNR method across various λ and window sizes ($\mathcal{W}$)

K	Window	λ					
		0.25		0.50		0.75	
		AMGDE	AMedGDE	AMGDE	AMedGDE	AMGDE	AMedGDE
5		2.5436	2.0311	2.5376	2.027	2.534	2.0276
10		2.422	1.9209	2.4133	1.9146	2.408	1.9088
15		2.3751	1.8675	2.3667	1.8627	2.3625	1.8585
20	15 min	2.3514	1.84	2.3433	1.8348	2.3396	1.8311
25		2.3365	1.8219	2.3283	1.8172	2.326	1.8125
50		2.3074	1.7835	2.3009	1.7795	2.2999	1.7759
100		2.2938	1.7641	2.2901	1.7618	2.2889	1.7573
5		2.5478	2.0296	2.5472	2.037	2.544	2.0375
10		2.4246	1.9137	2.4238	1.9256	2.4207	1.923
15		2.3787	1.8619	2.3779	1.8726	2.3735	1.8702
20	30 min	2.355	1.8338	2.3536	1.8416	2.3499	1.8423
25		2.3404	1.8146	2.339	1.8261	2.3352	1.8239
50		2.3098	1.7748	2.3097	1.7868	2.3057	1.7851
100		2.2954	1.7552	2.2962	1.7665	2.2942	1.7652
5		2.5389	2.0152	2.5487	2.0316	2.5519	2.0416
10		2.4136	1.8906	2.426	1.9163	2.4273	1.9259
15		2.3668	1.8382	2.3796	1.8639	2.3822	1.8743
20	1 hour	2.3419	1.8096	2.3558	1.8356	2.3574	1.8451
25		2.3271	1.7926	2.3407	1.8169	2.3427	1.8265
50		2.297	1.7545	2.3101	1.7776	2.3129	1.7877
100		2.2827	1.7357	2.2957	1.7568	2.299	1.7676



(a) Main Effects plot for AMGDE with the value of k as factor



(b) Main Effects plot for AMGDE with the value of Window Size as factor

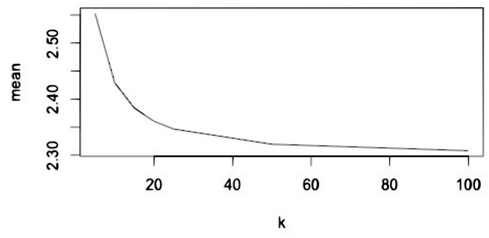
Figure 2: Main effects plots for AMGDE as response variable and the chosen value of k and Window Sizes as the factor variables for the SKNNR method

Table 6: Performance of RSKNNR with different sampling rates across three different window sizes (with $\lambda = 0.5$)

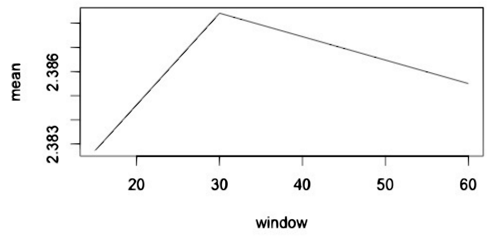
K	Window	Sampling Rate					
		5%		10%		20%	
		AMGDE	AMedGDE	AMGDE	AMedGDE	AMGDE	AMedGDE
5	15 min	2.5532	2.0483	2.5469	2.0414	2.5441	2.037
10		2.4293	1.9279	2.4241	1.9256	2.42	1.9241
15		2.3851	1.8745	2.3803	1.8748	2.3754	1.8726
20		2.3616	1.8465	2.3574	1.8439	2.3528	1.8444
25		2.3484	1.8276	2.3435	1.8276	2.3396	1.8274
50		2.3233	1.7848	2.3162	1.7878	2.312	1.7874
100		2.3185	1.6822	2.3062	1.7625	2.2995	1.7636
5	30 min	2.5595	2.0452	2.5548	2.0447	2.5546	2.0445
10		2.4367	1.9332	2.4299	1.9294	2.4283	1.9253
15		2.3904	1.8826	2.3849	1.8762	2.3825	1.8743
20		2.3667	1.8524	2.3631	1.8521	2.3589	1.8448
25		2.3528	1.8346	2.3495	1.8357	2.3447	1.8251
50		2.3271	1.765	2.3223	1.7711	2.3183	1.7711
100		2.3153	1.765	2.3105	1.7711	2.3058	1.7711
5	1 hour	2.5506	2.028	2.5564	2.0339	2.5542	2.0304
10		2.4318	1.9186	2.4305	1.9166	2.428	1.9175
15		2.3867	1.8641	2.3846	1.8693	2.3816	1.8647
20		2.3637	1.8369	2.3616	1.8402	2.3578	1.8369
25		2.3497	1.8201	2.347	1.8237	2.3431	1.8193
50		2.3208	1.7806	2.3189	1.7837	2.3148	1.7815
100		2.3065	1.7564	2.3058	1.7624	2.3013	1.759

Table 7: Comparison of the average time taken (in secs) for S50NNR and RS50NNR with different sampling rates across a period of 5 days with a 1 hour window (with $\lambda = 0.5$)

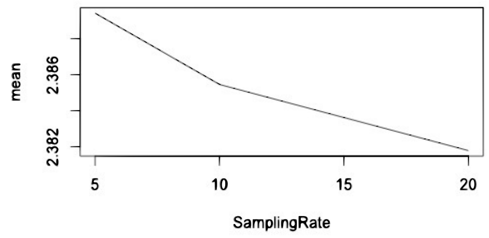
Method Name	Average Time Taken (in secs)
S50NNR	111.31
RS50NNR20P	21.91
RS50NNR10P	8.52
RS50NNR5P	4.50



(a) Main Effects plot for AMGDE with the value of k as factor



(b) Main Effects plot for AMGDE with the value of Window Size as factor



(c) Main Effects plot for AMGDE with the value of Sampling Rate as factor

Figure 3: Main effects plots for AMGDE as response variable and the chosen value of k, Window Size and the Sampling Rate as the factor variable for the RSKNNR method

As discussed in Section 2.6, Ensemble methods sometimes work better than the individual methods. In view of this we create an ensembles comprising of SSR, MMR and RS50NNR10P. Table 8 gives the performance of this ensemble with $\lambda = 0.5$ and different window sizes. We see that the AMGDE and AMedGDE does not depend on the choice of window size. Thus we take window size as 1 hour for further study.

Table 8: Performance of the ensemble model across different window sizes with $\lambda =$ 0.5

Method	Window	AMGDE	AMedGDE
	15 mins	2.2980	1.7150
Ensemble	30 mins	2.3092	1.7275
	1 hour	2.3089	1.7210

Table 9 gives a summary of the AMGDE and AMedGDE for each of the four methods viz. SSR, RS50NNR10P, MMR and the ensemble of these three methods on NYC1 dataset with prediction horizon of 1 hour and $\lambda = 0.5$. We see that the MMR method performs the best for both of the evaluation metrics. It is closely followed by the ensemble of the three methods.

An alternative way to compare the performances of these five methods is by using the Stochastic dominance approach discussed in Section 2.7. We compute the MGDE for each prediction horizon (i.e. for 119 prediction horizons with 1 hour duration in this case) and use the same to compute the ECDF of the MGDE. The ECDF of the MGDE is defined as

$$F_n(x) = \frac{\#\{MGDE_i \leq x\}}{n} \tag{14}$$

where n is the number of prediction horizons, $MGDE_i$ is the MGDE in the i^{th} prediction horizon and $\#\{MGDE_i \leq x\}$ is the number of $MGDE_i$ that are less than or equal to x.

Figure 4 shows the ECDF plots for the four methods on the NYC1 dataset. We find that the ECDF curve of the MMR method is to the left of that of the SSR and the RS50NNR10P methods. So, in terms of prediction accuracy, the MMR method is the best performer. But since the ECDF curves of the MMR and the Ensemble methods overlap, so we cannot say which one of them is stochastically dominant over the other. Hence, both the MMR and the Ensemble can be said to be comparable.

Table 9: Performance of the four methods using a Damped window model with $\lambda =$ 0.5 and window size= 1 hour

Method Name	AMGDE	AMedGDE
SSR	2.7293	1.8808
RS50NNR10P	2.3189	1.7837
MMR	2.2609	1.7176
Ensemble	2.3089	1.7210

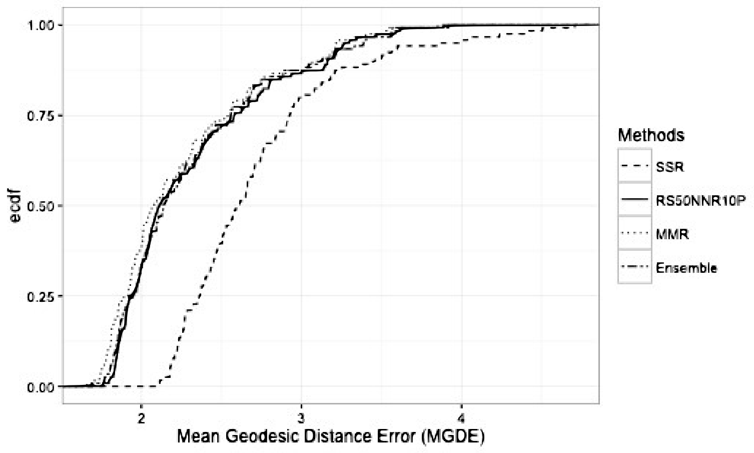


Figure 4: ECDF plots for the Mean Geodesic Distance Errors (MGDE) by prediction horizons for SSR, RS50NNR10P, MMR and Ensemble on the NYC1 dataset

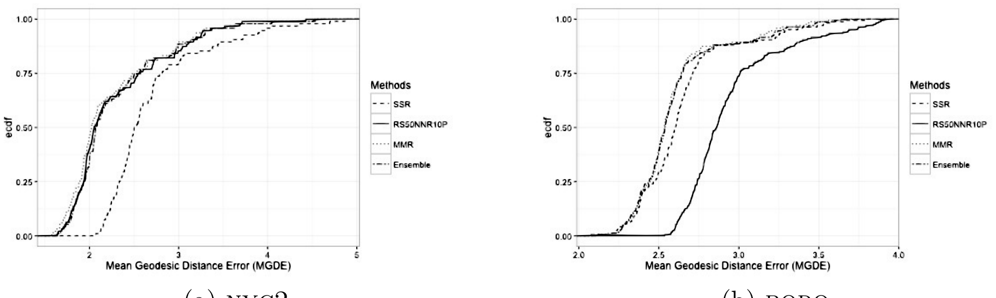
7.4 Performance on other datasets

In this section, we have have tried the four methods viz. SSR, RS50NNR10P, MMR and Ensemble method for predicting the drop off coordinates i.e. latitude and longitude using the pickup latitude and longitude as independent variabes on different datasets. These four methods are applied on each of the following datasets namely, NYC2, PORTO, SFBLACK and BORO (see Section 6.2 for details). We keep the window size as 1 hour and $\lambda = 0.5$ fixed all through this section.

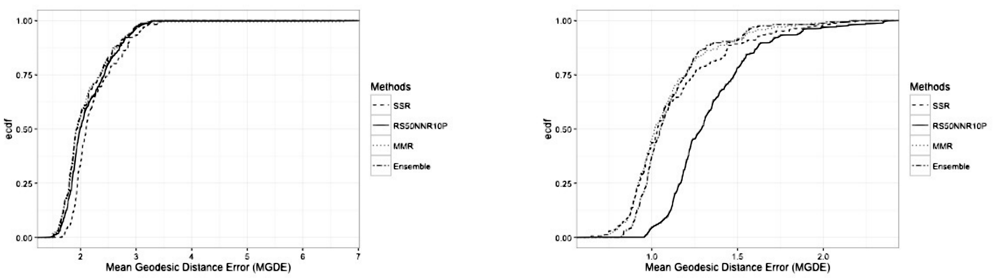
In the NYC2 dataset we again find that the MMR method performs the best both in terms of having the lowest AMGDE and AMedGDE. In the PORTO dataset, we find that AMGDE and AMedGDE values of MMR and Ensemble method are very close and these values are smaller than those of SSR and RS50NNR10P. While by AMGDE criterion the Ensemble method performs the best, the MMR is the best performer by AMedGDE criterion. In the SFBLACK dataset, we find that the MMR method is the best performer while considering the AMGDE criterion but SSR is the best performer by AMedGDE criterion. In the BORO dataset, we find that the MMR method performs the best in terms of having the lowest AMGDE while the SSR method is the best performer as per having the lowest AMedGDE. The Ensemble method comes second as per both of these criteria. The AMedGDE value of the MMR method is very close to that of the Ensemble method. A summary of the above results is given in Table 10.

In Figure 5 we show the ECDF plots of the MGDE for the four methods on the four datasets discussed above. For the NYC2 dataset in Figure 5 (a), we get similar results as it was for the NYC1 dataset i.e. both the MMR and the Ensemble are the best performing methods in terms of prediction accuracy. We get similar results for the BORO and PORTO datasets as well (see Figures 5 (b) and (c)). In case of the SFBLACK dataset, we find MMR and Ensemble methods have the best prediction accuracy and the SSR method has a better prediction accuracy than that of the RS50NNR10P method (see Figure 5 (d)).

In figure 6 we examine whether the performance of the four methods show any marked dependence on the average training window size of the five datasets for the destination prediction problem. We note that the relative performance of the four methods are not varying much with the average training window size. However, it may be noted that performance of



(a) NYC2 (b) BORO



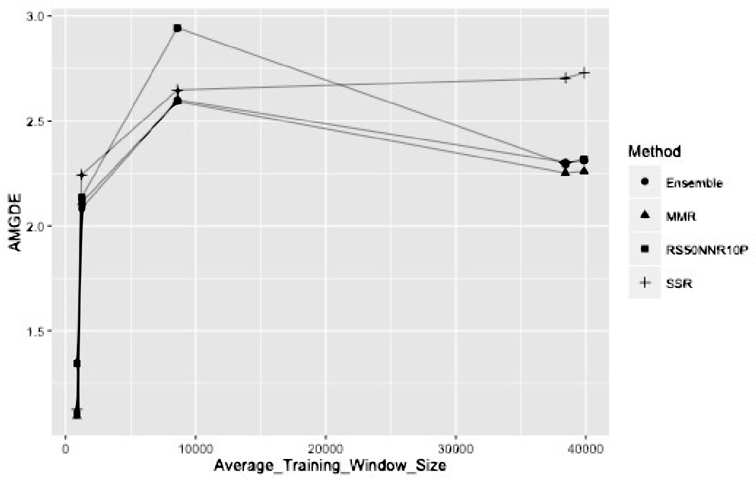
(c) PORTO (d) SFBLACK

Figure 5: ECDF plots for the Mean Geodesic Distance Errors (MGDE) by prediction horizons for SSR, RS50NNR10P, MMR and Ensemble on each of the four datasets

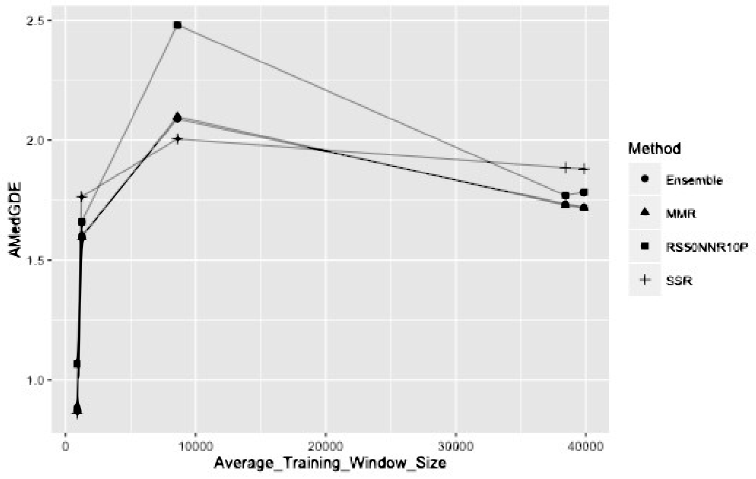
Table 10: Performance of the four methods on NYC2, PORTO, SFBLACK and BORO datasets using a 1 hour Damped window model with $\lambda = 0.5$

Dataset Name	Method Name	AMGDE	AMedGDE
NYC2	SSR	2.7043	1.8855
	RS50NNR10P	2.2954	1.77
	MMR	2.2522	1.7291
	Ensemble	2.3024	1.735
PORTO	SSR	2.2438	1.7649
	RS50NNR10P	2.1365	1.66
	MMR	2.1115	1.5957
	Ensemble	2.0837	1.602
SFBLACK	SSR	1.1256	0.8609
	RS50NNR10P	1.3437	1.0672
	MMR	1.0932	0.8716
	Ensemble	1.1104	0.8849
BORO	SSR	2.6469	2.0052
	RS50NNR10P	2.9434	2.4807
	MMR	2.5933	2.0984
	Ensemble	2.6003	2.0888

RS50NNR10P is better when the average training window size is high.



(a) AMGDE vs. Average Training Window Size



(b) AMedGDE vs. Average Training Window Size

Figure 6: Plot of AMGDE/ AMedGDE vs. Average Training Window Size for all the four methods for the Destination Prediction Problem

8 Discussion

The choice of the suitable method for use needs to take into account the time-accuracy trade-off. Since, in a streaming data context, the paradigm is to provide good results fast rather than giving the best possible solution we need to take into account the the total time taken (i.e. sum of training and prediction time) for the method to execute along with the prediction accuracy of the method. In this context we compare the MMR method with the Random Forests (RF) and Support Vector Regression (SVR) methods which are two standard methods which have been used earlier in destination prediction problem (Lam et al., 2015). In section 8.1 we investigate the variation of the total time taken and the prediction accuracy of these three different methods for different data sizes.

8.1 Static data experiment

We first perform an experiment to demonstrate how the total time taken and accuracy varies with respect to the training data size for each of the three methods viz. MMR, SVR and RF for the drop-off location prediction when the pickup location coordinates are known. We take random samples of different sizes for training and test data from the NYC1 dataset. The data size for different training data windows varies from 262 to $70,000$ for the five datasets used in this paper viz. NYC1, NYC2, BORO, porto and SFBLACK. So for this experiment we consider training datasizes of 250(250)1000(1000)10000(2000)20000(5000)70000. We take 20% of the training data size as the test data size. The test data is also randomly sampled fromNYC1. We then apply each of the three methods and plot the MGDE and the total time taken with respect to the training data size for MMR and RF in figure 7. We also do the same for MMR and SVR as shown in figure 8.

We see in figures 7 and 8, that compared to MMR, both RF and SVR have a better prediction accuracy but the time taken is very high especially for larger datasets. Hence RF and SVR are not suitable in situations where the velocity of the data stream is high. But

We notice from figures 7 and 8 that for smaller training data sizes, say less than 10000 the total time taken by both RF and SVR are comparable to that of MMR. Hence it indicates the possibility that for slow data streams which

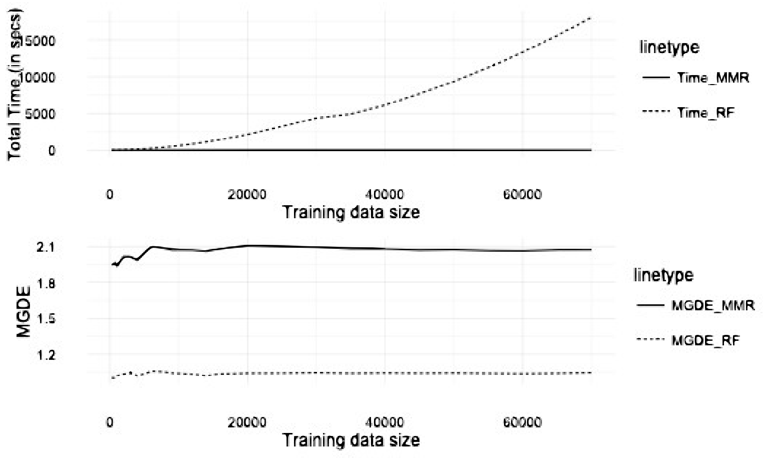


Figure 7: Time and MGDE vs. different training data size for static experiment with MMR and RF for drop-off location coordinate prediction given the pickup location coordinates are known

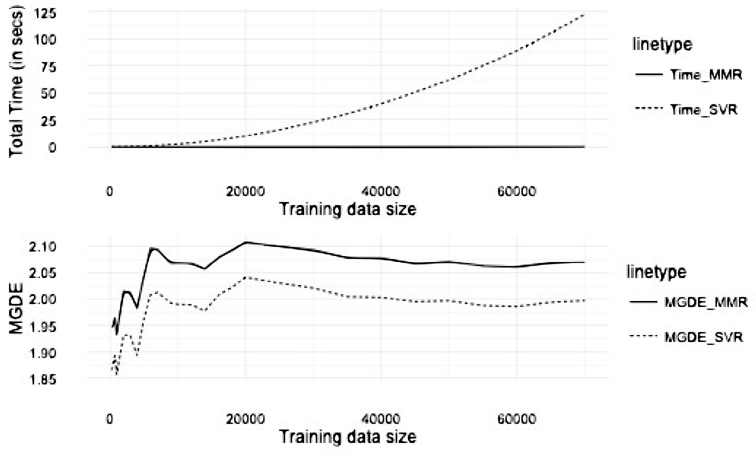


Figure 8: Time and MGDE vs. different training data size for static experiment with MMR and SVR for drop-off location coordinate prediction given the pickup location coordinates are known

leads to training data size less than 10000 these methods may be useful. Thus it is of interest to investigate how the incremental variations for RF and SVR performs with respect to MMR for such datasets. Since the PORTO and SFBLACK are of this kind we compare these three methods on these two datasets the details of which are given in section 8.2.

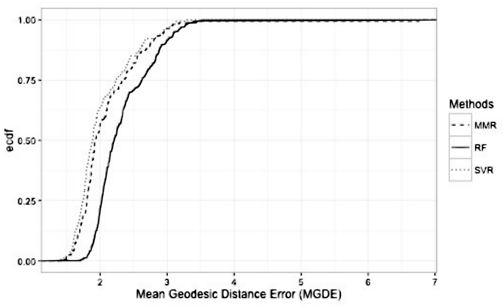
8.2 Performance Comparison of Incremental RF and SVR with MMR

As before we use a Damped window model with $\lambda = 0.5$ for comparison purposes. In table 11, we report the results of the comparison of performances of MMR, RF and SVR on the PORTO and SFBLACK datasets. We find that MMR is a better performer in terms of prediction accuracy than RF but the prediction accuracy of SVR is slightly better than MMR across both these datasets. In figure 9, we find that the ECDF of the MMR method is to the right of that of the SVR method for PORTO indicating that SVR method is better in terms of prediction accuracy for this dataset. However for the SFBLACK dataset, the ECDFs of the SVR method and MMR method overlaps and the ECDF of the RF method is to the right of both. Thus we can conclude that both SVR method and MMR method have better prediction accuracy than the RF method. Since no stochastic dominance is indicated between the SVR and MMR methods we cannot say which of these two have greater predictive accuracy.

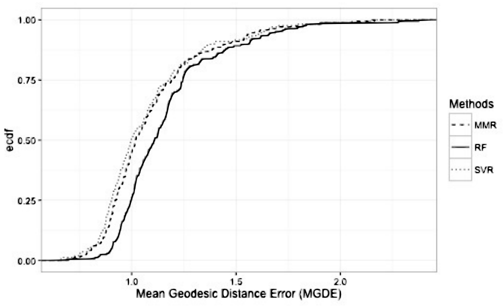
From sections 8.1 and 8.2, it can be concluded that for smaller datasets i.e. data size less than 10000, both SVR and MMR methods can be used but for larger datasets, the MMR is the method of choice.

Table 11: Performance of the MMR, RF and SVR methods on PORTOand SFBLACK datasets using a 1 hour Damped window model with $\lambda =$ 0.5

Dataset Name	Method Name	AMGDE	AMedGDE
PORTO	MMR	2.1115	1.5957
	RF	2.3352	1.8520
	SVR	2.0241	1.4852
SFBLACK	MMR	1.0932	0.8716
	RF	1.1661	0.9292
	SVR	1.0761	0.8373



(a) PORTO



(b) SFBLACK

Figure 9: ECDF plots for the Mean Geodesic Distance Errors (MGDE) by prediction horizons for MMR, RF and SVR on the PORTO and SFBLACK datasets

8.3 Choice of Paramters for Damped Window Model

In this section, we give an illustration for the method that may be adopted for selecting the parameters λ and c for the Damped window model. In Table 12, we give the results obtained when the MMR method is applied on the NYC1 dataset for different values of c (i.e. $0.05, 0.09, 0.15$ *and* 0.20) and λ (i.e. 0.25, 0.5 and 0.75). We choose the MMR method for this illustration since it is the best method among the new methods proposed in this paper. The window size is held constant at 1 hour. We find that as we increase the value of c, the predictive accuracy of the model drops. In Table 12 we see that the method is not very sensitive to the choice of the values of λ and c. In this paper, we have mostly worked with $\lambda = 0.5$ and $c = 0.09$. Any other choice of λ and c is expected to yield similar results.

Table 12: AMGDE of MMR method using a Damped window model (with window size= 1 hour) across different values of λ and the cut-off c for the NYC1 dataset

λ	Damped window model			
	c=0.05	c=0.09	c=0.15	c=0.2
0.25	2.2444	2.2458	2.2496	2.2516
0.5	2.2544	2.2609	2.2642	2.2671
0.75	2.2642	2.2671	2.2682	2.2682

In this section, we carry out a performance comparison study for the Damped window model and the sliding windows techniques. In Table 13, we compare compare a 1 hour sliding windows with a 1 hour Damped window model (with $\lambda = 0.5$ and $c = 0.09$) using a dense dataset i.e. NYC1 and a sparse dataset i.e. PORTO. We find that for the NYC1 dataset, the predictive accuracy for the MMR method using a Damped window model and a sliding windows are comparable whereas, in case of the PORTO dataset, the MMR method using the Damped window model clearly performs better than the one using sliding windows.

Table 13: Performance of MMR method using a 1 hour Damped window model (with $\lambda = 0.5$ and $c = 0.09$) and a 1 hour Sliding windows for the NYC1 and the PORTO datasets

Dataset	Damped window model		Sliding windows	
	AMGDE	AMedGDE	AMGDE	AMedGDE
NYC1	2.2609	1.7176	2.2579	1.7234
Porto	2.1115	1.5957	2.3805	1.637

9 Predicting Next Pickup Location

In this section we have explored the dynamic next pickup prediction problem where the dropoff latitude and longitude are used as the predictor variables. The four methods discussed earlier namely, SSR, RS50NNR10P, MMR and the Ensemble method are used for solving this problem and their performance are compared using the evaluation metrics discussed in Section 7.1. We have used the NYC1, NYC2 and PORTO datasets as described in section 6. We have derived the next pickup latitude and next pickup longitude variables of a vehicle after the completion of every trip from the information existing in these datasets.

In the NYC1 dataset we find that the Ensemble method performs the best but it is closely followed by the MMR method if we take the AMGDE as the evaluation metric. But if we take the AMedGDE as the evaluation metric then both the Ensemble method and the SSR method perform almost equally well and both are better than the other two methods.

In the NYC2 dataset we find that the Ensemble method performs the best in terms of having the lowest AMGDE and also AMedGDE. The MMR method comes second when AMGDE is considered while the SSR method is the second best when the AMedGDE is considered.

In the PORTO dataset, the MMR method performs the best followed closely by the RS50NNR10P method and the Ensemble method if one considers the AMGDE as the evaluation metric. But if we take the AMedGDE as the evaluation metric, then the ensemble method turns out to be the best followed closely by MMR and RS50NNR10P. A summary of the above results is given in Table 14.

In Figure 10, we compare the four methods in each of the three datasets by using stochastic dominance approach as we did in sections 7.3 and 7.4 for the destination prediction problem. Here also we find that the ECDF curves

Table 14: Performance of the four methods on the modified NYC1, NYC2 and PORTO datasets for next pickup prediction using a 1 hour Damped window model with $\lambda = 0.5$

Dataset Name	Method Name	AMGDE	AMedGDE
NYC1	SSR	1.6817	0.9917
	RS50NNR10P	1.7114	1.1887
	MMR	1.5555	1.0647
	Ensemble	1.5288	0.984
NYC2	SSR	1.6635	1.0223
	RS50NNR10P	1.6464	1.5533
	MMR	1.5274	1.0565
	Ensemble	1.4905	0.9796
PORTO	SSR	1.8326	1.2711
	RS50NNR10P	1.2589	1.0176
	MMR	1.2495	1.0023
	Ensemble	1.2605	0.9858

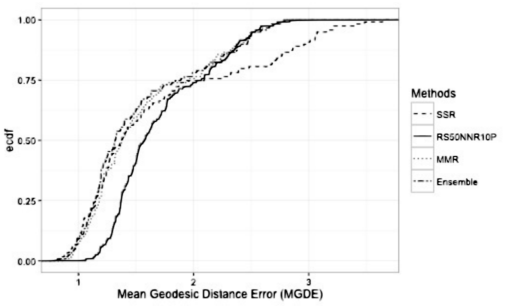
of both the MMR and the Ensemble methods overlap each other and both of these are to the left of the other methods. So we can say that the MMR and the Ensemble methods are the best performers in terms of prediction accuracy.

We have mentioned in Section 3 that Gandhi (2015) worked on the next pickup location problem where the author applied Artificial Neural Networks (ANN) in a batch setting. The author uses a simple 3-layer feed-forward neural network with an input layer, a hidden layer and an output layer. In this paper, the number of hidden nodes is taken in accordance with the Geometric pyramid rule (Masters, 1993) which states that for a three layer feed neural network with m inputs and n outputs the number of nodes H in the hidden layer is

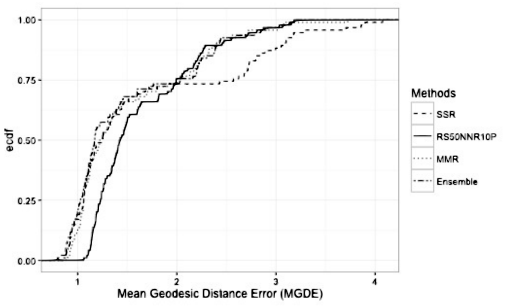
$$H = \sqrt{m * n} \tag{15}$$

Since the next pickup latitude and longitude are predicted separately using two inputs viz. drop-off latitude and longitude, the number of nodes in the hidden layer is 1.

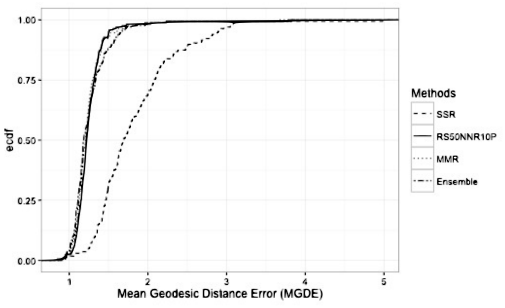
We apply an incremental ANN method using a Damped window model (with $\lambda = 0.5$ and window size= 1 hour) on the NYC1, NYC2 and PORTO datasets. In Table 15, we compare the MMR method with the ANN method. We find that for the NYC1 dataset, the ANN method performs comparably with the MMR in terms of prediction accuracy but for both the NYC2 and



(a) NYC1



(b) NYC2



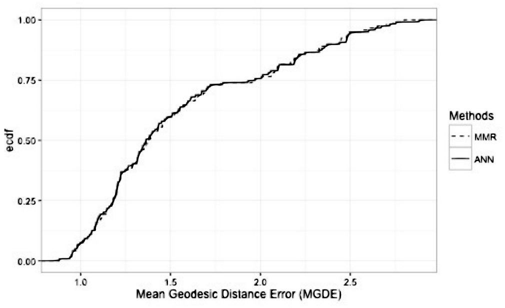
(c) PORTO

Figure 10: ECDF plots for the Mean Geodesic Distance Errors (MGDE) by prediction horizons for SSR, RS50NNR10P, MMR and Ensemble on each of the three datasets for the next pickup prediction problem

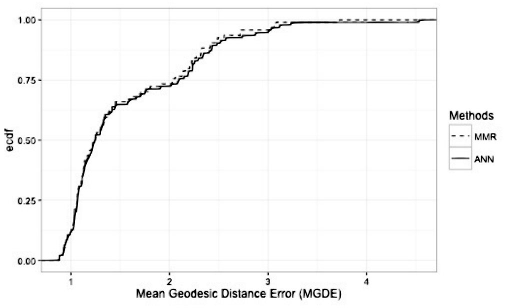
the PORTO datasets, the MMR method has a better prediction accuracy than that of the ANN method in terms of the AMGDE / AMedGDE. In Figure 11 we give the plot of the ECDF curves of the MGDE for both the ANN and MMR methods. We find that both the ECDF curves of ANN and the MMR methods overlap each other for all the three datasets and hence we cannot comment on which one of these two methods is better.

Table 15: Performance comparison of the MMR method with the ANN method on the modified NYC1, NYC2 and PORTO datasets for next pickup prediction using a 1 hour Damped window model with $\lambda = 0.5$

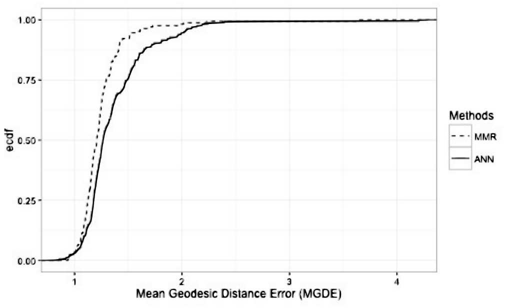
Dataset Name	Method Name	AMGDE	AMedGDE
NYC1	ANN	1.5538	1.0605
	MMR	1.5555	1.0647
NYC2	ANN	1.5555	1.0817
	MMR	1.5274	1.0565
PORTO	ANN	1.3696	1.0854
	MMR	1.2495	1.0023



(a) NYC1



(b) NYC2



(c) PORTO

Figure 11: ECDF plots for the Mean Geodesic Distance Errors (MGDE) by prediction horizons for MMR and ANN on each of the three datasets for the next pickup prediction problem

10 Conclusion

In Table 16 we give a summary of performance of the four methods viz. SSR, RS50NNR10P, MMR and the Ensemble method (using an one hour Damped window model with $\lambda = 0.5$) for the destination prediction problem on the five different datasets considered in this paper. We find that on the whole, the MMR and the Ensemble method performs better than the other two methods. Thus if a single method is desired to be implemented for such a problem we recommend the MMR method while if enough resources are available then the Ensemble method can be implemented.

In Table 17 we give a summary of performance of the four methods for the next pickup prediction problem on three different datasets using an one hour Damped window model with $\lambda = 0.5$. We find that the Ensemble method performs better than the other three methods in this case. However, if resource constraints dictate implementation of a single method then we recommend the MMR method as on the whole it has the next best performance after the Ensemble method.

In the datasets considered in this paper the travel distances were generally short. Thus the true potential of the SSR method may not have been fully exploited. It would be interesting to see whether the accuracy of the SSR method improves when the travel distances are larger, say in the case of inter-city travel.

In this paper, we have examined the destination / next pickup location prediction problem and have proposed four new incremental learning methods. We find from our studies that the MMR is the best performing method in terms of prediction accuracy when the training data sizes are large. When the training data sizes are small to moderate then both the RF and SVR methods are good choices considering both prediction accuracy and total computation time. When the next pickup prediction problem is considered the MMR method and the Ensemble (combination of SSR, MMR and RS50NNR10P) are both good performers in terms of prediction accuracy. The MMR method is also seen to be generally better than incremental ANN method in terms of prediction accuracy.

Table 16: Summary of performance of the four methods for destination prediction on different datasets using a 1 hour Damped window model with $\lambda = 0.5$

Dataset	No. of Pred Horizons	Avg. Pred Horizon Size*	Avg. Training Window Size*	Method Name	AMGDE	AMedGDE
NYC1	119	6825	39879	SSR	2.7293	1.8808
				RS50NNR10P	2.3189	1.7837
				MMR	2.2609	1.7176
				Ensemble	2.3089	1.721
NYC2	95	6806	38433	SSR	2.7043	1.8855
				RS50NNR10P	2.2954	1.77
				MMR	2.2522	1.7291
				Ensemble	2.3024	1.735
PORTO	167	208	1233	SSR	2.2438	1.7649
				RS50NNR10P	2.1365	1.66
				MMR	2.1115	1.5957
				Ensemble	2.0837	1.602
SFBLACK	167	146	869	SSR	1.1256	0.8609
				RS50NNR10P	1.3437	1.0672
				MMR	1.0932	0.8716
				Ensemble	1.1104	0.8849
BORO	167	1481	8585	SSR	2.6469	2.0052
				RS50NNR10P	2.9434	2.4807
				MMR	2.5933	2.0984
				Ensemble	2.6003	2.0888

* rounded up to the nearest integer

Table 17: Summary of performance of the four methods for the next pickup location prediction on different datasets using a 1 hour Damped window model with $\lambda = 0.5$

Dataset	No. of Pred Horizons	Avg. Pred Horizon Size*	Avg. Training Window Size*	Method Name	AMGDE	AMedGDE
NYC1	119	6335	37057	SSR	1.6817	0.9917
				RS50NNR10P	1.7114	1.1887
				MMR	1.5555	1.0647
				Ensemble	1.5288	0.984
NYC2	95	6320	35803	SSR	1.6635	1.0223
				RS50NNR10P	1.6464	1.5533
				MMR	1.5274	1.0565
				Ensemble	1.4905	0.9796
PORTO	167	191	1139	SSR	1.8326	1.2711
				RS50NNR10P	1.2589	1.0176
				MMR	1.2495	1.0023
				Ensemble	1.2605	0.9858

* rounded up to the nearest integer

References

Aggarwal, C. C. (2007). *Data Streams - Models and Algorithms*, volume 31 of *Advances in Database Systems*. Springer.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 1–16, New York, NY, USA. ACM.

Barceló, J., Montero, L., Marqués, L., and Carmona, C. (2010). Travel time forecasting and dynamic origin-destination estimation for freeways based on bluetooth traffic monitoring. *Transportation Research Record: Journal of the Transportation Research Board*, 2175:19–27.

Bosnic, Z., Rodrigues, P. P., Kononenko, I., and Gama, J. (2011). Correcting streaming predictions of an electricity load forecast system using a prediction reliability estimate. In *Man-Machine Interactions 2, Proceedings of the 2nd International Conference on Man-Machine Interactions, ICMMI 2011, The Beskids, Poland, October 6-9, 2011*, pages 343–350.

Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(421):123–140. Kluwer Academic Publishers, Hingham, MA, USA.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. Kluwer Academic Publishers, Hingham, MA, USA.

Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010). *Information Retrieval: Implementing and Evaluating Search Engines*. The MIT Press. isbn:0262026511, 9780262026512.

Cao, F., Ester, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *2006 SIAM Conference on Data Mining*, pages 328–339.

Castro, P. S., Zhang, D., Chen, C., Li, S., and Pan, G. (2013). From taxi gps traces to social and community dynamics: A survey. *ACM Comput. Surv.*, 46(2):17:1–17:34.

Chang, H., Tai, Y., and Hsu, J. (2010). Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18.

Chang, H.-W., Tai, Y.-C., Chen, H. W., and Hsu, J. Y.-J. (2008). iTaxi: Context-aware taxi demand hotspots prediction using ontology and data mining approaches. In *Proceedings of the 13th Conference on Artificial Intelligence and Applications*, TAAI'08, Berlin, Heidelberg. Springer-Verlag.

Chang, J. and Lee, W. (2004). Decaying obsolete information in finding recent frequent itemsets over data stream. *IEICE Transaction on Information and Systems*, E87-D(6).

Chang, T. (1986). Spherical regression. *Annals of Statistics*, 14(3):907–924.

Chen, C. (2014). *Understanding social and community dynamics from taxi GPS data*. PhD thesis, Institut National des T el ecommunications, Evry, France.

Chen, L., Lv, M., Ye, Q., Chen, G., and Woodward, J. (2011). A personal route prediction system based on trajectory data mining. *Information Sciences*, 181:1264–1284.

Cover, T. and Hart, P. (1967). Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, IT-13:21–27.

Davidson, R. (2008). *Stochastic dominance*. Palgrave Macmillan, UK, 2nd edition.

Downs, T. (2003). Spherical regression. *Biometrika*, 90(3):655–668.

Eberlein, X.-J., Wilson, N., and Bernstein, D. (1999). Modeling real-time control strategies in public transit operations. In Wilson, N., editor, *Computer-Aided Transit Scheduling*, volume 471 of *Lecture Notes in Economics and Mathematical Systems*, pages 325–346. Springer-Verlag, Heidelberg.

Economou, G., Pothos, V., and Ifantis, A. (2004). Geodesic distance and mst based image segmentation. In *In XII European Signal Processing Conference (EUSIPCO 2004)*, pages 941–944.

Ellis, B. (2014). *Real - Time Analytics: Techniques to analyze and visualize streaming data*. Wiley Publishing, 1st edition. isbn: 1118837916, 9781118837917.

Fisher, N. I., Lewis, T., and Embleton, B. J. J. (1993). *Statistical Analysis of Spherical Data*. Cambridge university press.

Förster, K., Monteleone, S., Calatroni, A., Roggen, D., and Tröster, G. (2010). Incremental knn classifier exploiting correct-error teacher for activity recognition. In Draghici, S., Khoshgoftaar, T. M., Palade, V., Pedrycz, W., Wani, M. A., and Zhu, X., editors, *ICMLA*, pages 445–450. IEEE Computer Society.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *journal of computer and system sciences*, 55(1):119–139.

Friedman, J. H., Baskett, F., and Shustek, L. J. (1975). An algorithm for finding nearest neighbours. *IEEE Trans on Computers*, C-24(10):1000–1006.

Fukunaga, K. and Narendra, P. M. (1975). A branch and bound algorithm for computing k nearest neighbours. *IEEE Trans. on Computers*, pages 750–753.

Gade, K. (2010). A non-singular horizontal position representation. *The Journal of Navigation*, 63:395–417.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition.

Gama, J. a., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37.

Gambs and Killijian (2012). Next place prediction using mobility markov chains. Available at: `http://homepages.laas.fr/mkilliji/docs/workshops/MPM12.pdf`.

Gandhi, S. (2015). Analysing and Predicting Driver's Next Location. Technical report, University of Warwick Erasmus Mundus Msc in Complex Systems Science. [Online, Accessed: 15-Dec-2015].

Grolemund, G. and Wickham, H. (2011). Dates and times made easy with lubridate. *Journal of Statistical Software*, 40(3):1–25.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer series in statistics. Springer, 2nd edition. New York.

Hazelton, M. (2008). Statistical inference for time varying origin–destination matrices. *Transportation Research Part B: Methodological*, 42(6):542–552.

Henry (2011). Uber gps traces. Github, Retrieved from: `https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata`. [Online, Accessed on 1-Sep-2015].

Herring, R., Hofleitner, A., Abbeel, P., and Bayen, A. (2010). Estimating arterial traffic conditions using sparse probe data. In *Proceedings of the International IEEE Conference on Intelligent Transportation Systems*, pages 929–936.

Hjort, N. L. and Claesken, G. (2003). Frequentist model average estimators. *Journal of the American Statistical Association*, 98(464):879–899.

Hu, H., Wu, Z., Mao, B., Zhuang, Y., Cao, J., and Pan, J. (2012). Pick-up tree based route recommendation. In *Proceedings of the International Conference on Web-Age Information Management*, pages 471–483.

Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2000). Diversion issues in real-time vehicle dispatching. *Transport Sci*, 34:426–438.

Jammalamadaka, S. R. and Sengupta, A. (2001). *Topics in circular statistics.* World Scientific Publishing Co. Pte. Ltd. isbn: 9810237782.

Jin, Y., Jiang, D., Yuan, S., Cao, J., Wang, L., and Zhou, G. (2008). OD *Count Estimation Based on Link Count Data*, pages 217–226. Springer Berlin Heidelberg, Berlin, Heidelberg.

Johnson, R. and Wichern, D. (2007). *Applied Multivariate Statistical Analysis.* Pearson Prentice Hall, 6th edition. Upper Saddle River, NJ, USA.

Kim, B. S. and Park, S. B. (1986). A fast nearest neighbour finding algorithm based on the ordered partition. *IEEE Trans on PAMI*, PAMI-8(6):761–766.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intell. Data Anal.*, 8(3):281–300.

Kotsiantis, S. B. and Pintelas, P. (2005). Selective averaging of regression models. *Annals of Mathematics, Computing & Teleinformatics*, 1(3):65–74.

Koychev, I. (2000). Gradual forgetting for adaptation to concept drift. In *In Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning*, pages 101–106.

Krumm, J. (2008). A markov model for driver turn prediction. In *SAE Technical Paper*. SAE International.

Krumm, J. and Horvitz, E. (2006). Predestination :inferring destinations from partial trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Computing*, UbiComp'06, pages 243–260, Berlin, Heidelberg. Springer-Verlag.

Kusner, M., Tyree, S., Weinberger, K. Q., and Agrawal, K. (2014). Stochastic neighbor compression. In Jebara, T. and Xing, E. P., editors, *Proceedings of* *the 31st International Conference on Machine Learning (ICML-14)*, pages 622–630. JMLR Workshop and Conference Proceedings.

Lam, H. T., Diaz-Aviles, E., Pascale, A., Gkoufas, Y., and Chen, B. (2015). (Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories. In *arXiv:1509.05257v1 [stat.ML]*.

Li, B., Zhang, D., Sun, L., Chen, C., Li, S., Qi, G., and Yang, Q. (2011). Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2011 IEEE International Conference on, pages 63–68. IEEE.

Li, H., Ho, C., and Lee, S. (2009). Incremental updates of closed frequent itemsets over continuous data streams. *Expert Systems with Applications*, 36(2):2451–2458.

Li, X., Pan, G., Wu, Z., Qi, G., Li, S., Zhang, D., Zhang, W., and Wang, Z. (2012). Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science*, 6(1):111–121.

Liao, L., Fox, D., and Kautz, H. (2004). Learning and inferring transportation routines. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI 2004)*.

Liaw, A. and Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22.

Lichman, M. (2013). UCI machine learning repository. Available: `https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015`. [Online, Accessed: 05-Dec-2015].

Liu, L., Andris, C., and Ratti, C. (2009). Uncovering cabdrivers' behavior patterns from their digital traces. *Computers, Environment and Urban Systems*, 34:541–548.

Liu, S., Liu, Y., Ni, L. M., Fan, J., and Li, M. (2010). Towards mobility-based clustering. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 919–928, New York, NY, USA. ACM.

Liu, X., Zhu, Y., Wang, Y., Forman, G., Ni, L. M., Fang, Y., and Li, M. (2012). Road recognition using coarse-grained vehicular traces. Technical report, HP Lab.

Lunga, D. and Ersoy, O. (2010). Spherical nearest neighbor classification: Application to hyperspectral data. Technical report, Purdue University.

Mackenzie, J. (1957). The estimation of an orientation relationship. *Acta Crystallographica*, 20:61–62.

Mardia, K. V. and Jupp, P. E. (2000). *Directional Statistics*. Wiley series in probability and statistics. Wiley, Chichester. Previous ed. published as: Statistics of directional data. London : Academic Press, 1972.

Marmasse, N. and Schmandt, C. (2002). A user-centered location model. *Personal and Ubiquitous Computing*, 6:318–321.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. Academic Press Professional, Inc., San Diego, CA, USA.

Mendes-Moreira, J. a., Jorge, A. M., de Sousa, J. F., and Soares, C. (2012). Comparing state-of-the-art regression methods for long term travel time prediction. *Intell. Data Anal.*, 16(3):427–449.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., and Leisch, F. (2015). *e1071: Misc Functions of the Department of Statistics Probability Theory Group (Formerly: E1071) TU Wien*.

Miao, F., Lin, S., Munir, S., Stankovic, J., Huang, H., Zhang, D., He, T., and Pappas, G. J. (2015). Taxi dispatch with real-time sensing data in metropolitan areas: a receding horizon control approach. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems*, ICCPS '15, pages 100–109, New York, NY, USA. ACM.

Miclet, L. and M.Dabouz (1983). Approximative fast nearest neighbour recognition. *Pattern Recognition Letters*, 1:277–285.

Moreira-Matias, L., Cats, O., Gama, J. a., Mendes-Moreira, J. a., and de Sousa, J. F. (2016a). An online learning approach to eliminate bus bunching in real-time. *Appl. Soft Comput.*, 47(C):460–482.

Moreira-Matias, L., Gama, J., Ferreira, M., and Mendes-Moreira, J. (2013a). Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393 – 1402.

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., and Damas, L. (2013b). Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402.

Moreira-Matias, L., Gama, J., Mendes-Moreira, J., and de Sousa, J. (2014). An incremental probabilistic model to predict bus bunching in real-time. *In Advances in intelligent data analysis*, XIII:227–238.

Moreira-Matias, L., Gama, J., Mendes-Moreira, J., Ferreira, M., and Damas, L. (2016b). Time-evolving O-D matrix estimation using high-speed gps data streams. *Expert systems with Applications*, 44(C):275–288.

Murty, M. N. and Devi, V. S. (2011). *Pattern Recognition: An Algorithmic Approach*. Undergraduate topics in computer science. Springer, New York.

Muthukrishnan, S. (2003). Data streams: Algorithms and applications. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 413–413, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Navot, A., Shpigelman, L., Tishby, N., and Vaadia, E. (2006). Nearest neighbor based feature selection for regression and its application to neural activity. In Weiss, Y., Schölkopf, B., and Platt, J. C., editors, *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada*, pages 996–1002. MIT Press.

NYC-TLC (2014). TLC trip record data. Available: `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`. [Online, Accessed on 17-Nov-2015].

Oldfield, T. and Hubbard, R. (1994). Analysis of c_α geometry in protein structures. *Proteins*, 18:324–337.

Palma, A. T., Bogorny, V., Kuijpers, B., and Alvares, L. O. (2008). A clustering-based approach for discovering interesting places in trajectories.

In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 863–868, New York, NY, USA. ACM.

Papadopoulos, A. and Manolopoulos, Y. (2005). *Nearest Neighbor Search: A Database Perspective*. Series in Computer Science. Springer.

Park, J., Murphey, Y., McGee, R., Kristinsson, J., Kuang, M., and Phillips, A. (2014). Intelligent trip modeling for the prediction of an origin–destination traveling speed profile. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1039– 1053.

Parry, K. and Hazelton, M. L. (2012). Estimation of origin–destination matrices from link counts and sporadic routing data. *Transportation Research Part B: Methodological*, 46(1):175–188.

Persad-Maharaj, N., Barbeau, S., Labrador, M., Winters, P., Pérez, R., and Georggi, N. (2008). Real-time travel path prediction using GPS-enabled mobile phones. In *15th World Congress on Intelligent Transportation Systems, New York*.

Phithakkitnukoon, S., Veloso, M., Bento, C., Biderman, A., and Ratti, C. (2010). Taxi-aware map: identifying and predicting vacant taxis in the city. *Ambient Intelligence*, 6439:86–95.

Powell, W. B. (1986). A stochastic model of the dynamic vehicle allocation problem. *Transp. Sci.*, 20:117–129.

Powell, W. B. (1987). An operational planning model for the dynamic vehicle allocation problem with uncertain demands. *Transportation Research Part B: Methodological*, 21(3):217–232.

Putatunda, S. (2017). *Streaming Data: New Models and Methods with Applications in the Transportation Industry*. PhD thesis, Indian Institute of Management Ahmedabad.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ratcliffe, J. G. (2006). *Foundations of hyperbolic manifolds*, volume 149 of *Graduate texts in mathematics*. Springer, New York, 2nd edition.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Shakhnarovich, G., Darrell, T., and Indyk, P. (2005). *Nearest-Neighbor Methods in Learning and Vision*. Neural Information Processing Series. The MIT Press, Cambridge, Massachusetts, USA.

Simmons, R., Browning, B., Zhang, Y., and Sadekar, V. (2006). Learning to predict driver route and destination intent. In *IEEE Intelligent Transportation Systems Conference, 2006*, ITSC '06, pages 127–132. IEEE.

Stephens, M. A. (1979). Vector correlation. *Biometrika*, 66:41–48.

Sun, L., Zhang, D., Chen, C., Castro, P., Li, S., and Wang, Z. (2012). Real time anomalous trajectory detection and analysis. *Mobile Networks and Applications*, pages 1–16.

Tiesyte, D. and Jensen, C. S. (2008). Similarity-based prediction of travel times for vehicles traveling on known routes. In *GIS*, pages 14:1–14:10.

Tran, V. (2015). Optimization of scheduling and dispatching cars on demand. Master's thesis, San Jose State University.

Tsagris, M. and Athineou, G. (2016). *Directional: Directional Statistics*. R package version 1.9.

Urbano, F. and Cagnacci, F., editors (2014). *Spatial Database for GPS Wildlife Tracking Data*. Springer International Publishing.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Veloso, M., Phithakkitnukoon, S., and Bento, C. (2011). Urban mobility study using taxi traces. In *Proceedings of the International Workshop on Trajectory Data Mining and Analysis*, TDMA '11, pages 23–30, New York, NY, USA. ACM.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review. XXIII*, 176:88–93.

Wallace, J. R. (2012). *Imap: Interactive Mapping*. R package version 1.32.

Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated.

Weinberger, K. Q. and Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classication. *Journal of Machine Learning Research*, 10:207–244.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5:241–259.

Xue, A., Zhang, R., Zheng, Y., Xie, X., Huang, J., and Xu, Z. (2013). Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE 2013)*, ICDE '13, pages 254–265, Washington, DC, USA. IEEE Computer Society.

Yue, Y., Zhuang, Y., Li, Q., and Mao, Q. (2009). Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In *Proceedings of the 17th international conference on geoinformatics*, pages 1–6.

Yunck, T. P. (1976). A technique to identify nearest neighbours. *IEEE Trans. SMC*, SMC-6(10):678–683.

Zhang, B. and Srihari, S. N. (2004). Fast k-nearest neighbour classification using cluster-based trees. *IEEE Trans. PAMI*, 26(4):525–528.

Zhang, D., Li, N., Zhou, Z.-H., Chen, C., Sun, L., and Li, S. (2011). iBAT: detecting anomalous taxi trajectories from gps traces. In *Proceedings of the 13th international conference on ubiquitous computing*, UbiComp '11, pages 99–108, New York, NY, USA. ACM.

Zhu, Y. and Sasha, D. (2002). Statstream: Statistical monitoring of thousands of data streams in real time. In *Proceedings of the 28th International Conference on Very Large Data Bases*, number TR2002-827 in VLDB '02, pages 358–369. VLDB Endowment.