Generating Artificial Data to Evaluate Machine Learning Predictive Algorithms for Bus Travel Time

Leandro S. Ribeiro, Thiago P. Faleiros, Maristela Holanda

Computer Science Department – University of Brasília (UNB), Brasília, Brazil
leandro.santos.r@gmail.com,{thiagodepaulo,mholanda}@unb.br

Abstract. This paper proposes a simulator called Simulator for Bus Information Analysis (S-BIA). The proposed simulator is capable of quickly generating a large amount of data that may be used to train bus travel time predictive algorithms in an urban transport network. To validate the proposal, a case study was carried out on a bus line in the city of Brasília/DF, Brazil. In the case study, S-BIA generated data for several scenarios that differ in distinct levels of variability and these data were used to evaluate the performance of machine learning predictors in each of the scenarios. Moreover, we provided experimental evaluation of several machine learning algorithms and we compared their performance to our proposed method based in Linear Combination of predictors. The mean absolute error was adopted in these experiments to evaluate the quality of the predictor's results, and our proposed linear combiner approach was able to improve the performance of the prediction in both real and artificial data.

Categories and Subject Descriptors: I.2.6 [**Learning**]: Knowledge acquisition; I.6 [**Simulation and modeling**]: Simulation Output Analysis; H.2.8 [**Database Applications**]: Spatial databases and GIS

Keywords: machine learning, traffic simulation, bus travel time prediction.

1. INTRODUCTION

Urban residents rely on different modes of public transportation for their daily commute. Based on that, many works have been developed aiming to improve the efficiency and the accessibility of urban transportation services [Lima and Campos 2017; Monteiro et al. 2017]. In this context, accurate and real-time travel time information for buses has an important role because, among other reasons, it can help passengers better plan their trips and minimize waiting times.

Many factors such as weather conditions, day of week, time of day, and current traffic conditions may influence bus travel times. However, the exact nature of such relationships between travel times and predictor variables is usually not known and somehow these factors need to be incorporated into prediction algorithms either indirectly through binned analyses or through direct modeling [Kormáksson et al. 2014]. Moreover, travel times in urban areas are prone to high degrees of variability due to the presence of traffic lights, congestion, geometric conditions of roads and weather conditions [Reddy et al. 2016].

One way to analyze the impact of urban traffic conditions on the task of bus travel time prediction is to collect data from the traffic network in a variety of situations. These situations are related to high and low degrees of variability in traffic conditions. Thus, a predictor can be used to predict bus travel times for each of these situations, and the prediction results can be compared for each scenario. However, when using real traffic data, it is difficult to analyze the impact of different traffic conditions on predictor performance. Therefore, one way to overcome this challenge would be to use traffic simulation tools. Using simulation tools, it is possible to vary only one specific traffic parameter while all others remain constant and analyze its impact on predictor performance. For example, you can keep all other parameters constant while varying only the number of buses in the transport network, the probability of events, how much events impact traffic or the length and timing of the peak time window. Besides that, Wen et al. in [Wen 2018] highlight safety, convenience and low cost as advantages of using a simulation.

The results obtained by various predictors deteriorate under different traffic conditions, and it is hard to find real datasets with different traffic conditions with explicit parameters that may influence bus travel times. Thus, considering the aforementioned assumptions, a bus traffic simulator was built with several parameters related to urban traffic conditions. This study is designed to assess the hypothesis that linear combiner predictor can obtain lower mean square error when compared to machine learning-based predictors in an environment with different levels of urban traffic.

Here, we extend the results obtained by [dos Santos Ribeiro et al. 2018], from which we can highlight the following contributions: i) the development of a simulator capable of quickly generating a large amount of data that may be used to train the predictor; ii) the evaluation performance of a simple K-Nearest Neighbors (K-NN) predictor algorithm for many scenarios with several degree of traffic variability. In addition to the contributions from the previous publication, the current paper contributions are: i) the proposal of a linear combiner that combines the results of K-Nearest Neighbors and Support Vector Machine predictions; ii) the experimental comparison of machine learning-based regression predictors in simulated and real data. We perform an experimental analysis with the following algorithms: K-Nearest Neighbors (K-NN) [Aha et al. 1991], an Artificial Neural Network (ANN) [Shalev-Shwartz and Ben-David 2014], a Support Vector Machine (SVM) [Cortes and Vapnik 1995], a Random Forest (RF) [Ho 1995], a gradient boosting algorithm [Friedman 1999] XGBoost, a meta-ensemble $META_2$ that combines a K-NN predictor with an SVM predictor using a linear regression and the Linear Combiner. As long as simulation generated data could be biased, prediction results using simulated data were compared with prediction results using real traffic data. The results showed that the proposed Linear Combiner (LC) predictor outperformed the other algorithms. Moreover, the rank of predictors was consistent for both real and simulated data, which indicates that the LC predictor is a viable option for predicting bus travel with diverse traffic conditions.

The current paper is organized as follows: in Section 2 the related work is discussed; in Section 3 the architecture of the proposed tool and its components is described; in Section 4 the proposed LC predictor is presented; in Section 5 a case study with its respective results is presented; and finally, in Section 6 there are the conclusions.

2. RELATED WORK

Many simulation models can be classified into two types: macroscopic and microscopic [Helbing et al. 2002]. The macroscopic traffic models are restricted to the description of the collective vehicle dynamics in terms of the spatial vehicle density $\rho(x, t)$ and the average velocity $V(x, t)$ as a function of the location x and time t. In contrast, the microscopic traffic models delineate the positions $x_{\mathrm{a}}(t)$ and velocities $v_{\mathrm{a}}(t)$ of all interacting vehicles as a function of time t.

The macroscopic models are used for large scale traffic, treating traffic as a liquid, frequently applying hydrodynamic flow theory to vehicle behavior. Macroscopic models such as Cellular Automaton (CA), car following model and IDM/MOBIL are widespread models used within the traffic science community [Sommer et al. 2011].

In [Wen 2018], traffic simulations of vehicles in a connected environment, traveling through a commercial area, were carried out with the PARAMICS tool to test the collection of traffic data and the prediction of travel times. Four types of models were constructed for the analysis of travel times: linear regression, multivariate adaptive regression spline, stepwise regression and elastic net. The results showed that the approaches had similar performance in terms of Root Mean Square Error (RMSE).

A comprehensive list of traffic simulation software is presented in [Barceló et al. 2010], describing their approaches to model building and implementation. The microscopic approach is represented by the following types of software VISSIM, AVENUE, Paramics, Aimsun, MITSIM, SUMO and DRACULA. The macroscopic approach is represented by METANET. Although most of these tools are complete in terms of functionality, a simpler model may be implemented to generate a large volume of data with less computational effort.

When there is a concern about the exact positions of simulated nodes, macroscopic and mesoscopic models cannot offer this level of detail, then only microscopic simulations are considered. Microscopic simulations model the behavior of single vehicles and interactions between them [Sommer et al. 2011]. However, there are different disadvantages, such as a high computational time and the requirement for detailed information, which might limit the applicability of a microscopic model to medium networks and those that do not operate in real time [Adacher and Tiriolo 2018].

Considering the trade-off between macroscopic and microscopic models, one of the challenges of this work is to propose a simulator that can provide the exact position of each simulated node, but with a good applicability in real-time applications and without the need for high computational time, merging characteristics of the macroscopic and microscopic models. Considering the high computational cost required in the general simulator software, and the lack of a specific simulator to generate training data for bus travel time predictive algorithms, we propose a dedicated simulator able to model several external events that may occur in daily traffic and to generate data useful for the experimental evaluation of predictive algorithms. The proposed simulator seeks to achieve these objectives by reducing the complexity of the model, while the results maintain the necessary coherence for the analysis of the performance of travel time predictors.

As for the travel time prediction problem, [Vanajakshi and Rilett 2007; 2004] used SVM and ANN for short-term traffic parameters prediction and reported that SVM is a viable alternative to ANN. The studies concluded that SVM would be a better choice for travel time predictions when only a small dataset is available for training. In addition, they compared different methods of travel time prediction including historical method, time series analysis, ANN and SVM. Comparisons showed that the performance of both ANN and SVM was better than the other models.

Wu et al. shows that SVM performs better than historical and real-time methods to predict travel times [Wu et al. 2003]. Besides that, SVM avoids the over-fitting problem and can be trained through a linear optimization process which is not observed in traditional ANN [Bin et al. 2006].

Yu et al. developed an SVM based model to predict bus travel times. The results showed that the SVM model performed better than prediction models based on historical averages, Autoregressive Integrated Moving Average (ARIMA), and ANN. They compared four models: ANN, SVM, K-NN and Linear Regression. The results suggested that the performance of the SVM model was the most accurate among the four models for predicting arrival time of buses at bus stops [Yu et al. 2010].

In [Mendes-Moreira et al. 2015] three different algorithms RF, Projection Pursuit Regression (PPR) and SVM were dynamically combined in a heterogeneous configuration to improve the prediction accuracy of long-term travel times [Mendes-Moreira et al. 2015].

High variability conditions may affect the performance of the travel time predictor. In [Reddy et al. 2016] a bus travel time predictor was proposed based on SVM, and promising results were observed under high variability conditions. These results motivated the choice of SVM as part of our travel time prediction method based on Linear Combination. Similarly, a K-NN predictor was also chosen to be part of the LC predictor. K-NN is one of the simplest and oldest methods used for pattern classification and prediction, and it often yields efficient performance, in certain cases, its accuracy is greater than state-of-art predictors [Hassanat et al. 2014; Hamamoto et al. 1997]. In addition, a K-NN predictor can be used to predict travel times using real time data or near real time data while an SVM predictor predicts travel times using historical data.

Given the successful SVM results predicting travel times and the characteristics of good generalization ability, ability to find global minimums in training sets, ability to avoid over-fitting problems, among others of this model, we chose to use an SVM predictor in the LC. Moreover, due to the K-NN speed and simplicity of implementation and the potential accuracy gain combining different predictors, a K-NN predictor was chosen to be combined with an SVM predictor in the proposed LC.

3. SIMULATOR FOR BUS INFORMATION ANALYSIS (S-BIA) ARCHITECTURE

The S-BIA consists of four software components and a Geographic Database. The software components are: Time Controller, Line Simulator, Trip Simulator, and Location REST Service. The diagram with the components is shown in Figure 1. Each software component will be described in the following sections.

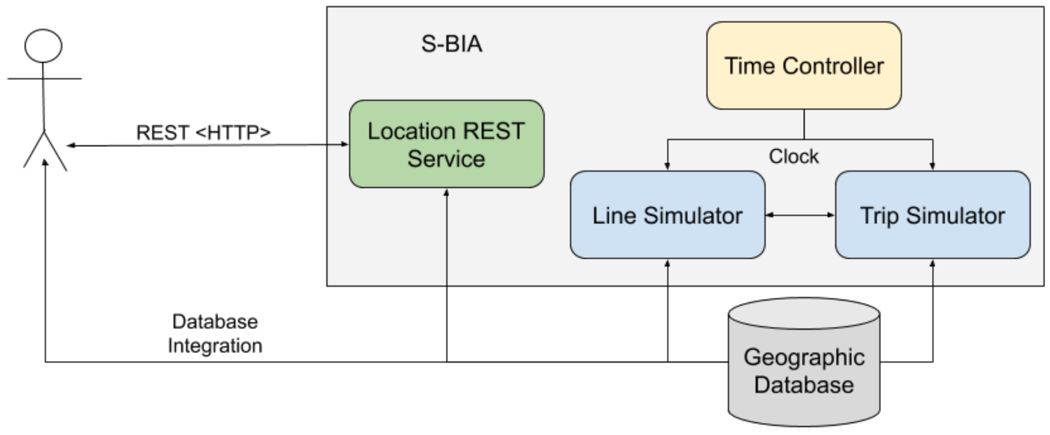


Fig. 1. S-BIA Architecture.

3.1 Bus Line Graph Structure

A bus line can be represented as a graph, where the bus stops are the graph nodes, while the paths between bus stops are the graph edges, as shown in Figure 2.

The edges have different status (normal, light event, moderate event, and severe event), may be under the influence of other edges (severe, moderate, light and absent) and they have associated average velocities. On the other hand, nodes have associated delays. Node delays, edge status and influence will be better explained in Section 3.2.

3.2 Line Simulator

The Line Simulator performs scheduled tasks according to a parameterized period. The main activities performed by this component are: updating edge status, updating neighboring edges influence, updating edges average velocities, updating node delays, and updating edges in geographic database. The responsibility of this component is to update attributes related to the behavior of roads and bus stops.

To simulate the occurrence of events, such as accidents or bad weather, each edge is classified as: normal, light event, moderate event, and severe event. The normal status represents a situation without events. In other words, it is not under the influence of external events that may negatively

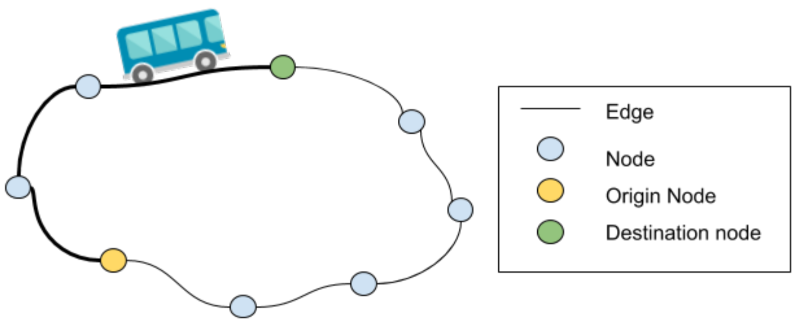


Fig. 2. Bus Line Graph.

impact the traffic flow, whereas the other statuses represent the occurrence of events that impact the traffic flow with increasing degrees of severity from light to severe.

During the edge's status updating process, if an edge has normal status, a pseudo-random probabilistic event determines an increasing probability of severe, moderate and light events. This means the occurring probability of a light event is greater than the occurring probability of a severe event. If no event occurs, the edge retains normal status.

If an edge is affected by some event during the edge's status updating process, the event regression probability is evaluated. A severe event regresses to a moderate event, while a moderate event regresses to a light event and a light event regresses to the normal status. Again, the regression probability of a light event is greater than the regression probability of a moderate event, which is greater than the regression probability of a severe event. Figure 3 illustrates the edge status state machine.

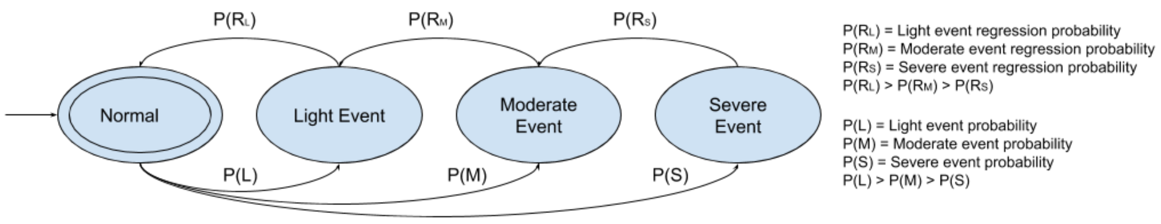

P(RL)
P(RM)
P(RS)
Normal
Light Event
Moderate Event
Severe Event
P(L)
P(M)
P(S)
P(RL) = Light event regression probability
P(RM) = Moderate event regression probability
P(RS) = Severe event regression probability
P(RL) > P(RM) > P(RS)
P(L) = Light event probability
P(M) = Moderate event probability
P(S) = Severe event probability
P(L) > P(M) > P(S)

Fig. 3. Edge Status State Machine.

To represent the influence of neighboring edges, four influence classifications were created: severe, moderate, light and absent. An edge is under severe influence when the edge immediately downstream has a severe event, on the other hand, an edge is under moderate influence when the edge that is after the edge immediately downstream has a severe event. Finally, an edge is under light influence when the edge immediately upstream has a severe event. If the edge does not fit into any of the above rules it will be under absent influence. Figure 4 illustrates the influence of an edge under severe event on its neighboring edges.

After updating edge status and edge influence, edge velocity is updated. Each edge has an average velocity that is initially represented by the maximum velocity allowed in that edge. In the edge updating average velocity first step, the path maximum velocity is multiplied by a correction factor that varies according to the edge status. This factor is always a number less than or equal to one. The resulting velocity will always be less than or equal to the maximum edge velocity allowed. The higher the severity of the event, the lower the correction factor. This correction factor has a normal distri-

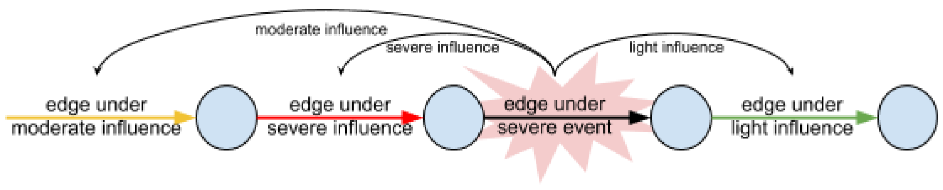

moderate influence
severe influence
light influence
edge under moderate influence
edge under severe influence
edge under severe event
edge under light influence

Fig. 4. The Neighboring Edges' Influence.

bution so that one simulation parameter determines the mean and another determines the standard deviation.

In the second update step, the result calculated in the first step is multiplied by a peak time correction factor. This factor is not applied if the current time is outside the interval of the peak hour. Similar to the status correction factor, the peak time correction factor has a parameter that determines the Gaussian mean and a parameter that determines the Gaussian standard deviation, but in this case, the Gaussian mean is not a constant value. Rather, it is determined by a linear discontinuous function that decreases in the first half of the peak hour window and increases in the second half of this window, so that the velocity variation becomes a little smoother. The discontinuous linear function can be replaced by any other mathematical function to better model this phenomenon.

Finally, in the third update step, the result of the second step is multiplied by the influence correction factor – the greater the influence of the neighboring edge, the lower the influence correction factor and the lower the final average velocity. As with all other correction factors, this factor has a normal distribution parameterized by the mean and the standard deviation.

Node delays represent the amount of time buses spend at bus stops. These delays also have a normal distribution with mean and standard deviation parametrized. Presently, to simplify the process, the peak time window does not affect node delays. However, it is possible to apply a peak time correction factor to these delays, increasing the delay at peak hours.

The last activity of the Line Simulator is to update the Geographic Database with edge average velocities. This allows any graphical tool that can integrate with a geographic database to monitor edge status and edge average velocity at simulation time.

3.3 Trip Simulator

Similar to the Line Simulator, the Trip Simulator executes scheduled tasks according to a parameterized period. The main activities of this component are: updating bus position, updating bus velocity, and updating Geographic Database with bus positions. The responsibility of the Trip Simulator is to update the behavior attributes of the buses.

Each graph element behaves differently, whether it is an edge or a node. When a fraction of time is available to a bus that is on a node, this time is consumed while the bus stands at the bus stop. However, when the bus is on an edge and a fraction of time is available, it moves over the edge at a distance that is a function of its instantaneous velocity and the available time. The travel times spent by all the buses at each of the stops or crossing each of the edges are recorded in the Geographic Database at the exact moment that the vehicle leaves the respective graph element.

The instantaneous bus velocity is a function of the graph element average velocity. Graph nodes always have zero average velocity, while the average velocities of edges are calculated by the Line Simulator. The instantaneous velocity of the buses is the multiplication of the edge average velocity by a velocity oscillation factor. Consequently, two buses on the same edge do not necessarily have the same instantaneous velocity. The buses' instantaneous velocities have a normal distribution, with mean equal to edge average velocity and parameterizable standard deviation, which is the velocity oscillation factor.

Node delays are multiplied by a delay oscillation factor, so two buses do not necessarily remain the same time period at a certain bus stop. The delay oscillation factor also has a normal distribution with mean equal to the node average delay and parameterizable standard deviation – the delay oscillation factor.

Finally, the Geographic Database is updated with the positions of the buses, allowing other tools to monitor these positions at runtime through database integration.

3.4 Location REST Service

The Location REST Service allows information to be retrieved from the entire bus fleet through a Hypertext Transfer Protocol (HTTP) call. The information is made available through a JavaScript Object Notation (JSON) document and includes name, line, velocity, and location (latitude and longitude) of all buses.

This service, in addition to allowing system integration independent of the platform, the operational system, or programming language, also makes it possible to obtain bus fleet simulated data in real time.

3.5 Time Controller

The Time Controller is nothing more than a clock that controls the simulation time. All other system components use the Time Controller clock instead of the operational system clock. The Time Controller clock speed is set by the "Time Multiplier" parameter. If the parameter is set to two, then the time passage in the simulation will be twice as fast as the real time passage. Therefore, this component allows the simulation to be executed both in real time and in an accelerated time, so that weeks of traffic simulation can be executed in a few hours.

3.6 Geographic Database

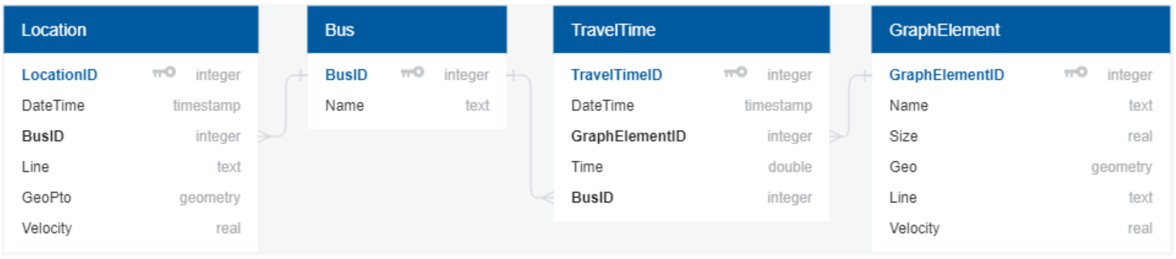
The Geographic Database, illustrated in Figure 5, stores information from nodes and edges, bus locations over time, and travel time information of all edges and nodes. In addition to maintaining the entire simulation history the database also works as an integration point between S-BIA and other systems.

The "Location" table, in Figure 5, stores the buses' geographic location and velocity at a given time. The "TravelTime" table records the time a bus has remained in a given graph element. The "GraphElement" table has all graph nodes and edges including each element name, size, maximum speed allowed , and the bus line name. When the element is a node, the "Graph Element" table has the geolocated point with the node location. However, when the element is an edge this table stores the set of geolocated points and lines that represent the edge trace and position.

3.7 S-BIA Parameters

Adjusting S-BIA parameters, a traffic situation with high variability can be created: constant changes in bus speeds, occurrence of many external events and many other traffic situations. Likewise, other adjustments can create a situation with low variability: with constant velocities and little or no occurrence of external events.

In addition, by adjusting the parameters it is possible to define the number of buses that will be running during the simulation, the windows of the peak hours, the Line Simulator and the Trip Simulator update periods, and the time multiplication factor. Table I lists all S-BIA parameters.



Location	
LocationID	integer
DateTime	timestamp
BusID	integer
Line	text
GeoPto	geometry
Velocity	real

Bus	
BusID	integer
Name	text

TravelTime	
TravelTimeID	integer
DateTime	timestamp
GraphElementID	integer
Time	double
BusID	integer

GraphElement	
GraphElementID	integer
Name	text
Size	real
Geo	geometry
Line	text
Velocity	real

Fig. 5. Geographic Database.

Table I. S-BIA Parameters

Name	**Description**
fleetSize	Number of buses
severeEventProb	Severe event probability
moderateEventProb	Moderate event probability
lightEventProb	Light event probability
normalCorrectionFactor	Normal status correction factor
lightCorrectionFactor	Light event correction factor
moderateCorrectionFactor	Moderate event correction factor
severeCorrectionFactor	Severe event correction factor
correctionFactorSD	Correction factor standard deviation
severeEventEndProb	Severe event ending probability
moderateEventEndProb	Moderate event ending probability
lightEventEndProb	Light event ending probability
absentInfluence	Absent influence factor
lightInfluence	Light influence factor
moderateInfluence	Moderate influence factor
severeInfluence	Severe influence factor
influenceSD	Influence factor standard deviation
morningPeakTime	Morning peak time
afternoonPeakTime	Afternoon peak time
tripSimulatorUpdate	Trip Simulator update period
lineSimulatorUpdate	Line Simulator update period
timeMultiplier	Simulation time multiplier factor
delayOscillationFactor	Node delay oscillation factor
delayOscillationFactorSD	Standard deviation of the node delay oscillation factor
velocityOscillationFactor	Edge velocity oscillation factor
velocitOscillationFactorSD	Standard deviation of the edge velocity oscillation factor
peakTimeCorrectionFactor	Peak time correction factor
peakTimeCorrectionFactorSD	Standard deviation of the peak time correction factor

4. LINEAR COMBINER

A linear combination is an expression in which terms are multiplied by constants, usually called weights, and then the results are summed [Lee et al. 2009]. Here, the concept of linear combination will be used to calculate the estimated travel time of buses that leave and have as destination predetermined bus stops. In this case, the terms that will be multiplied by the weights are the results of two preview predictions: the prediction result of a K-NN predictor (P_{KNN}) and the prediction result of an SVM predictor (P_{SVM}). Equation 1 represents in a simplified way the linear combination between the SVM prediction and the K-NN prediction.

$$\alpha \cdot P_{KNN} + \beta \cdot P_{SVM}, \tag{1}$$

where $\alpha + \beta = 1$.

The weights in Equation 1 are represented by the Greek letters α, which is the weight of the K-NN prediction, and β, which is the weight of the SVM prediction. The sum of the weights is equal to one. The linear combination is done for each edge of the graph $\{A_1, A_2, ..., A_{82}\}$. A 10-fold cross validation with Mean Absolute Error (MAE) cost function and a grid search for hyperparameter optimization was performed. Table II shows the set of values for α and β hyperparameters.

Table II. LC hyperparemeters values.

Hyperparameters	**Values**
Weight of K-NN prediction (α)	$\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$
Weight of SVM prediction (β)	$\{\beta \in \mathbb{R} \mid \beta = 1 - \alpha\}$

The LC predictor result is the weighted average between the best SVM prediction result and the best K-NN prediction result, such that α is a hyperparameter of the LC predictor and the weight of the K-NN prediction and β is also a hyperparameter of the LC predictor and the weight of the SVM prediction. Algorithm 1 presents the steps of the linear combination process.

Algorithm 1: Linear Combination.

```
Input: Instances of training data (I)
Output: All edges' linear combinations ($M_{CL}$)
begin
    $M_{SVR} = SVR.Fit(I)$;
    $M_{KNN} = KNN.Fit(I)$;
    Best models $M_{CL} = \emptyset$;
    $\alpha = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$;
    K-fold cross validation $k = 10$;
    Algorithm $cl = LinearCombiner()$;
    Cost Function $f_c = MAE$;
    for each edge $i$ in $i = 1, 2, ..., 82$ do
        $searcher = GridSearch(cl, \alpha, M_{SVR}[i], M_{KNN}[i], k, f_c)$;
        $result = searcher.fit(I[i])$;
        $m = result.getBestModel()$;
        $M_{CL}.add(m)$;
    end
    return $M_{CL}$;
end
```

It is important to note that the LC predictor is an ensemble, that uses two learning algorithms to obtain a predictor with better results than any other algorithm that is part of the ensemble individually [Opitz and Maclin 1999; Polikar 2006; Rokach 2010]. Empirically, ensembles tend to produce better results when there is a great diversity among the models [Kuncheva and Whitaker 2003; Sollich and Krogh 1996].

5. CASE STUDY

A case study was carried out using simulated and real data to evaluate the performance of travel time predictors using scenarios with different variability degrees. The simulations were performed on a bus line located in downtown Brasília/DF, Brazil.

Geolocation data from the Federal District urban transport network, on the right side of Figure 6, were obtained from a GeoServer[1] maintained by the Federal District Transit Department (DFTRANS). For the case study, only the "CIRCULAR-W3-SOUTH-NORTH-L2-NORTH-SOUTH" bus line was used, on the left side of Figure 6. The bus line was represented by a graph with 82 nodes named N1 to N82, and 82 edges named A1 to A82.

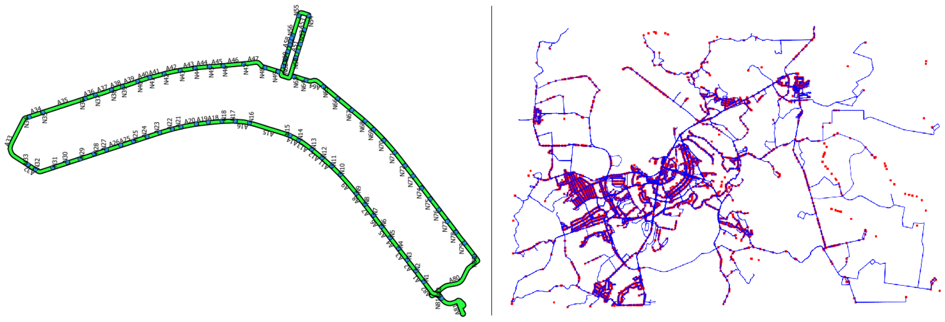


Fig. 6. Federal District Urban Transport Network.

To create different levels of variability, simulator and predictor parameters were changed. The parameters were divided into 5 sets $\mathrm{C_i}$, $\mathrm{C_j}$, $\mathrm{C_k}$, $\mathrm{C_l}$, and $\mathrm{C_m}$. Each of these sets has 3 different configurations. For instance, $\mathrm{C_i(0)}$ is the parameter set $\mathrm{C_i}$ in configuration 0 while $\mathrm{C_m(2)}$ is the parameter set $\mathrm{C_m}$ in configuration 2. The parameter sets $\mathrm{C_i}$, $\mathrm{C_j}$, $\mathrm{C_k}$, and $\mathrm{C_i}$ are simulator parameters and are listed in Table I with their respective descriptions. The parameter set $\mathrm{C_m}$ has a single predictor parameter, which determines the quantity of previous travel times used to train the prediction algorithm.

Two data sets were created. The first one was called low deviation data set, because it has lower values for the *delayOscillationFactorSD* and *velocitOscillationFactorSD* parameters. The second one was called high deviation data set, because it has higher values for those parameters. Since there are 2 data sets, 4 simulator parameter sets, 1 predictor parameter set – each parameter set with 3 possible configurations – the amount of simulated scenarios is 162, and the total amount of predicted scenarios is 486. Table III shows the 5 parameter sets and their values for each configuration and data set.

For each data set (high deviation and low deviation) S-BIA ran for 3 hours for each of the scenarios described, with a fleet of 82 buses and a simulation time multiplication factor of 60, for each 3 hours of simulation a little more than a week of traffic data were generated. Table IV shows that S-BIA ran for approximately 2 months and generated more than 10 years of traffic data.

Experiments were also carried out with real traffic data obtained from a REST service provided by the Piracicabana bus operator, which covers the bus line "CIRCULAR-W3-SOUTH-NORTH-L2-NORTH-SOUTH". Geolocation data were collected in the period from 2018-11-28 00:38 to 2018-12-19 18:21, i.e., slightly more than 3 weeks of traffic data. During this period, 6,827,229 locations were collected from 59 different buses, approximately 310,000 locations per day. Traffic data is updated every 5 seconds, as reported in DFTRANS open data website. The data provided by the bus operator REST service are described in Table V.

Since the amount of simulated data is very large, only 8 edges were selected for the experiment: $A_1, A_9, A_{33}, A_{35}, A_{51}, A_{58}, A_{80}, A_{81}$. The edge samples were taken as heterogeneously as possible, selecting edges of different sizes, geographical positions and geometries. In order to further reduce the number of data sets, only 3 scenarios were chosen: the most conservative scenario, with the least variability $\{C_i(0), C_j(0), C_k(0), C_l(0), C_m(0)\}$, the average scenario $\{C_i(1), C_j(1), C_k(1), C_l(1), C_m(1)\}$,

[1]http://geoserver.org

Table III. Simulation Scenarios

Parameter			
	Ci		
Parameter	**Ci(0)**	**Ci(1)**	**Ci(2)**
severeEventProb	0,0000	0,0005	0,0010
moderateEventProb	0,0000	0,0010	0,0020
lightEventProb	0,0000	0,0020	0,0040
	Cj		
Parameter	**Cj(0)**	**Cj(1)**	**Cj(2)**
lightCorrectionFactor	0,90	0,80	0,70
moderateCorrectionFactor	0,75	0,65	0,55
severeCorrectionFactor	0,60	0,50	0,40
peakTimeCorrectionFactor	0,80	0,70	0,60
	Ck		
Parameter	**Ck(0)**	**Ck(1)**	**Ck(2)**
lightInfluence	1,00	0,90	0,80
moderateInfluence	1,00	0,80	0,70
severeInfluence	1,00	0,70	0,60
	Cl − low deviation data set		
Parameter	**Cl(0)**	**Cl(1)**	**Cl(2)**
delayOscillationFactorSD	0,01	0,05	0,10
velocitOscillationFactorSD	0,01	0,05	0,10
	Cl − high deviation data set		
Parameter	**Cl(0)**	**Cl(1)**	**Cl(2)**
delayOscillationFactorSD	0,25	0,50	0,75
velocitOscillationFactorSD	0,25	0,50	0,75
	Cm		
Parameter	**Cm(0)**	**Cm(1)**	**Cm(2)**
qtdPreviousTrips	6	4	2

Table IV. Simulation Data.

Data Sets	2
Scenarios per Data Set	243
Hours per Scenario	3
Real Time Simulation Hours	1458
Real Time Simulation Days	60.75
Time Multiplier	60
Simulation Time Hours	87480
Simulation Time Days	3645
Simulation Time Weeks	520.71
Simulation Time Months	121.5
Simulation Time Years	10.13

Table V. Bus Fleet Data.

Name	**Description**
ID	Vehicle identification number.
Timestamp	Vehicle location timestamp.
GPS-Latitude	Latitudinal position in WGS 84 coordinate system.
GPS-Longitude	Longitudinal position in WGS 84 coordinate system.
GPS-Direction	Bus line direction.
Line	Bus line.
Velocity	Vehicle velocity.

and the more chaotic scenario, with greater variability $\{Ci(2), C_j(2), C_k(2), C_l(2), C_m(2)\}$. Considering 8 different edges, 3 scenarios for each edge and 2 deviation data sets (low deviation and high deviation), we have a total of 48 data sets to train and test the models. Due to reasons concerning reproducibility, all simulated datasets and mean absolute error values obtained in our experimental evaluation are freely available [2].

Six other predictors besides the LC predictor were trained and evaluated for performance comparison purposes. The algorithms used by the other 6 predictors were: K-NN, ANN, SVM, RF, XGBoost and a meta-ensemble that combines a K-NN predictor with an SVM predictor using a linear regression, this meta-algorithm was named $META_2$. All these predictors were trained and evaluated for later comparison with the LC predictor.

The features data used to train and test the models were: the current travel time, the period of the day represented by a real number contained in the interval $[0; 24)$, the day of the week, the average bus speed, and η previous travel times where η is determined by the predictor parameter "qtdPreviousTrips".

A 10-fold cross validation with MAE cost function was performed for all predictors. All predictors' hyperparameters except XGBoost were fit using grid search. The XGBoost hyperparameters were fit using Bayesian search due to its large number and range of hyperparameters.

In this paper we presented the regression performances considering mean absolute error, all results values are freely available. Figure 7 shows the number of data sets in which each predictor obtained the smallest error for the simulated traffic data set. The LC predictor obtained the best overall result, with the smallest error for 28 out of 48 data sets, that is, it was better in 58.33% cases. The performance is even better if the result is restricted only to high deviation data sets, in which case the LC predictor was better in 22 of 24 data sets, i.e., it was better in 91.67% cases. However, if only low deviation data sets are considered, the predictor RF is the one that obtained the best result, with smallest error in 62.5% cases. The second best overall result was the SVM predictor, which got the smallest error in 39.58% cases.

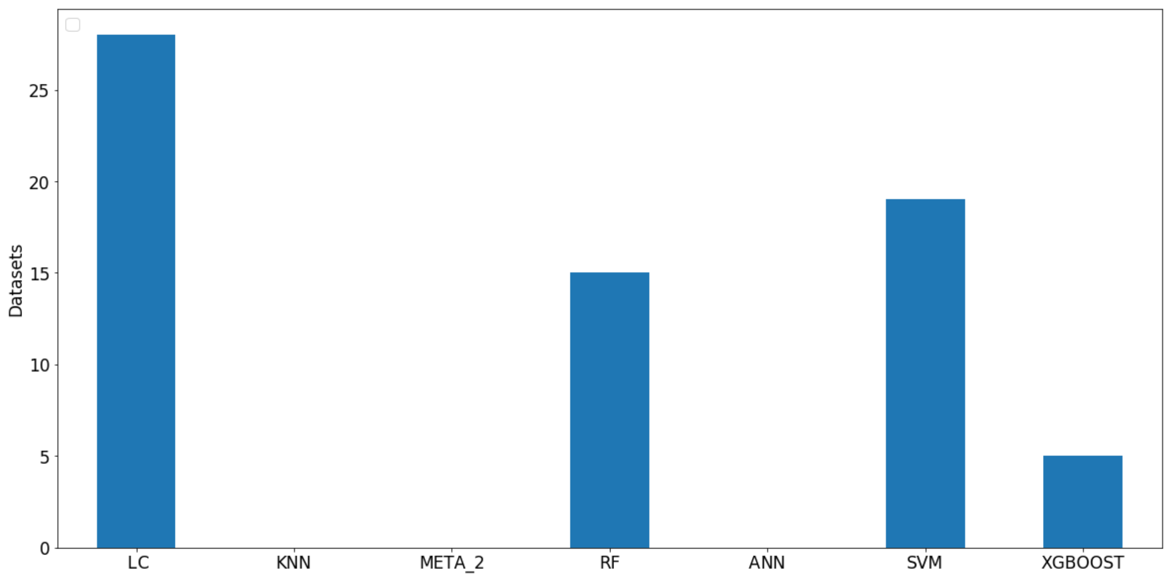


Fig. 7. Number of Data Sets with Smallest Error for Simulated Traffic Data.

A Friedman's test and Nemenyi's post-hoc test with 95% confidence interval were applied to the predictors' error data to evaluate the statistically significant differences between the algorithms. The

[2]Data and results in terms of mean absolute error available at: `https://github.com/curupiras/Generating-Artificial-Data-for-Bus-Travel-Time-Predictions.git`

Friedman test is a non-parametric test based on average ranking differences [Demšar 2006]. The algorithm with highest performance has the rank of 1, the second best performance 2, and so on. Nemenyi's post-hoc test is used to find pairs of algorithms which produce statistically significant differences [de Paulo Faleiros et al. 2017]. These are an advisable statistically significant difference test to use when there is a control algorithm (usually the proposed one) and results from multiple data sets [Trawiński et al. 2012; García et al. 2010]. The null hypothesis states that all the algorithms performed equivalently and therefore their ranks should be equal. In our case we want to determine if the proposed algorithm obtained better results with statistically significant differences in comparison to other algorithms.

Figure 8 presents the critical difference diagram to illustrate the results of the statistical significance test for the simulated data. In this diagram, the algorithms connected by a line do not present statistically significant differences among them. The LC predictor is superior with statistically significant differences to RF, $META_2$, ANN and K-NN algorithms. In this case, there are no statistically significant differences between LC, SVM and XGBoost algorithms, however the LC algorithm is first in the statistical test ranking.

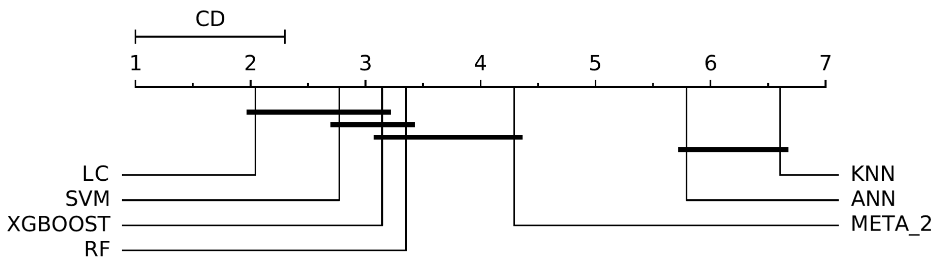


Fig. 8. Statistical Test for Simulated Data Experiment.

Figure 9 and Figure 10 show the critical difference diagrams for the high deviation simulated data set and the low deviation simulated data set respectively. In Figure 10 the LC algorithm is superior with statistically significant differences to ANN and K-NN algorithms. There are no statistically significant differences between LC algorithm and RF, SVM, XGBoost and $META_2$ algorithms. RF algorithm was first in the statistical ranking for this subset, however with no statistically significant difference to LC.

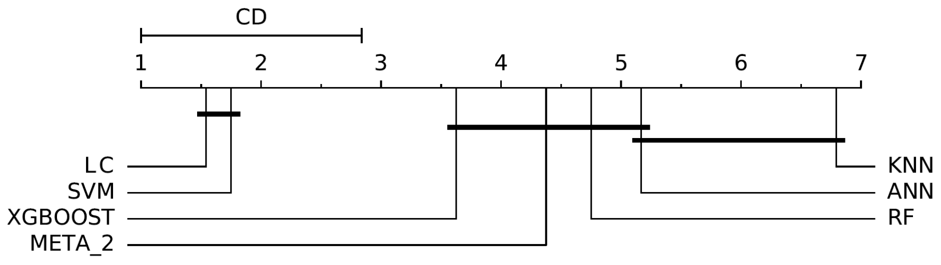


Fig. 9. Statistical Test for High Deviation Simulated Data Experiment.

Finally, in Figure 9 the LC algorithm is superior with statistically significant differences to XGBoost, $META_2$, RF, ANN and K-NN algorithms. There are no statistically significant differences between LC and SVM. LC algorithm was the first in the statistical ranking for this subset, showing that LC performs very well when the standard deviations of speed and delay are high.

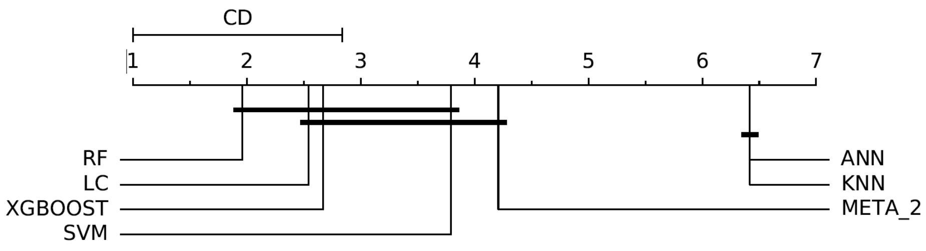


Fig. 10. Statistical Test for Low Deviation Simulated Data Experiment.

Figure 11 shows the number of data sets in which each predictor obtained the smallest error for the real traffic data set. The LC predictor had the second best result, with the smallest error for 30 of 82 data sets, that is, it was better in 36.59% cases. The algorithm with the best result was XGBoost, with the smallest error for 32 of 82 data sets, better in 39.58% cases. The RF predictor, which had the best result for low deviation simulated data set, was better in 17.07% cases, ANN predictor was better in 6.10% cases and the SVM predictor was better in 15.85% cases. The $META_2$ predictor outperformed the other algorithms in only 1 case and the K-NN predictor in no case.

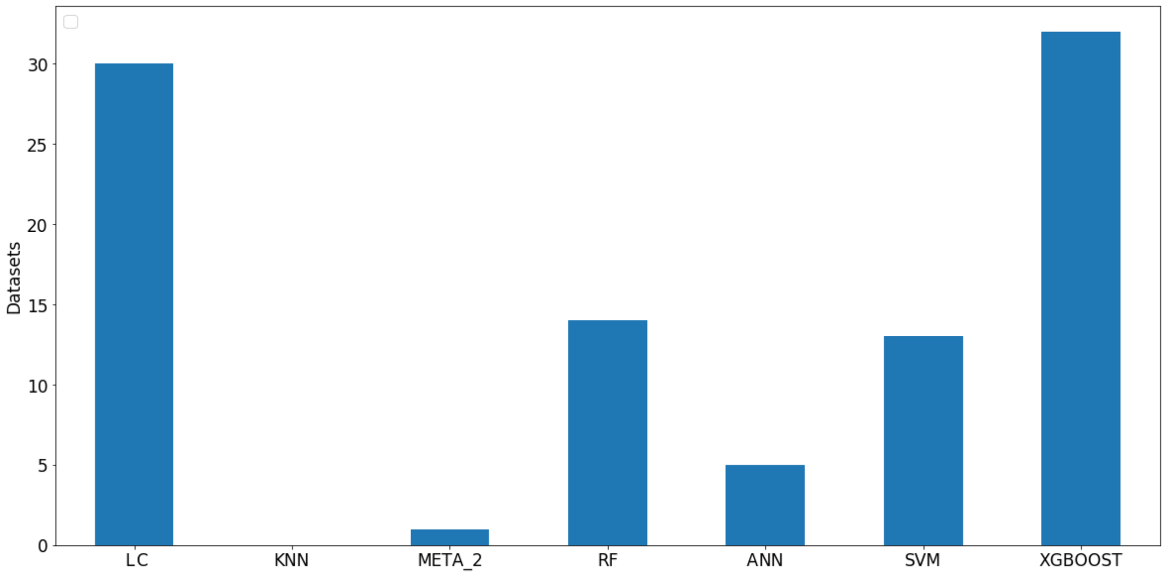


Fig. 11. Number of Data Sets with Smallest Error for Real Traffic Data.

Figure 12 presents the critical difference diagram to illustrate the statistical significance test results for the real traffic data. According to Figure 12 LC is superior with statistically significant differences to RF, $META_2$, ANN and K-NN algorithms. In this case, there are no statistically significant differences between CL, SVM and XGBoost algorithms. Although the XGBoost algorithm has the smallest error for 32 cases and LC for 30 cases, LC was the first in the statistical test ranking, because in cases where LC did not have the lowest error its error was very close to the error of the best algorithm for the respective case, that is, LC obtained more consistent results than XGBoost algorithm in most cases.

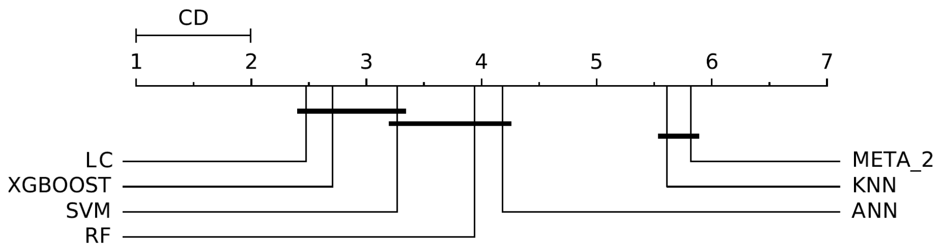


Fig. 12. Statistical Test for Real Data Experiment.

6. CONCLUSION

In this work, a simulator capable of quickly generating a large amount of data for travel time prediction algorithms was presented. The implementation was able to provide the geographic position and speed of each simulated bus, upon request, either using REST API calls or geographic database integration.

Furthermore, to validate the implementation, a case study was carried out on a bus line in the city of Brasília. Over 10 years of traffic data with different levels of variability were generated in approximately 2 months, proving S-BIA's ability to generate a large quantity of data in a short period of time for several different scenarios. The simulated data were used to train and test seven different bus travel time predictors and the LC predictor performance was evaluated in terms of mean absolute error and compared with other state of art algorithms. The results showed that the LC predictor performance when trained with real traffic data was similar to its performance when it was trained with simulated data.

With simulated data, LC predictor showed the best performance in most data sets, with the lowest error in 58% cases. In the high deviation data subset the performance was even better presenting the smallest error in 91.67% cases. In addition, LC predictor proved to be the first in the statistical test ranking, with statistically significant difference in relation to RF, $META_2$, ANN and K-NN algorithms.

With real traffic data, LC predictor presented the smallest error in 36.59% cases, losing only to the XGBoost algorithm, which presented the smallest error in 39.58% cases, but LC predictor was the first in the statistical test ranking, with statistically significant difference in relation to RF, $META_2$, ANN and K-NN algorithms.

Future work includes: testing S-BIA with other bus lines; refinements to the model to simulate traffic in multiple lanes; to add the possibility of multiple path choice; and to improve experiments training prediction algorithms with simulated data and testing them with real data.

REFERENCES

Adacher, L. and Tiriolo, M. A macroscopic model with the advantages of microscopic model: A review of cell transmission model's extensions for urban traffic networks. *Simulation Modelling Practice and Theory*, 2018.

Aha, D. W., Kibler, D., and Albert, M. K. Instance-based learning algorithms. *Machine learning* 6 (1): 37–66, 1991.

Barceló, J. et al. *Fundamentals of traffic simulation.* Vol. 145. Springer, 2010.

Bin, Y., Zhongzhen, Y., and Baozhen, Y. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems* 10 (4): 151–158, 2006.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning* 20 (3): 273–297, 1995.

de Paulo Faleiros, T., Rossi, R. G., and de Andrade Lopes, A. Optimizing the class information divergence for transductive classification of texts using propagation in bipartite graphs. *Pattern Recognition Letters* vol. 87, pp. 127–138, 2017.

Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* 7 (Jan): 1–30, 2006.

dos Santos Ribeiro, L., de Paulo Faleiros, T., and Terto de Holanda, M. Generating artificial data for bus travel time predictions. *Proceedings XIX GEOINFO*, 2018.

Friedman, J. Greedy function approximation: A gradient boosting machine http://www. salford-systems. com/doc. *GreedyFuncApproxSS. pdf*, 1999.

García, S., Fernández, A., Luengo, J., and Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 180 (10): 2044–2064, 2010.

Hamamoto, Y., Uchimura, S., and Tomita, S. A bootstrap technique for nearest neighbor classifier design. vol. 19, pp. 73 – 79, 02, 1997.

Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. Solving the problem of the k parameter in the knn classifier using an ensemble learning approach. *CoRR* vol. abs/1409.0919, 2014.

Helbing, D., Hennecke, A., Shvetsov, V., and Treiber, M. Micro-and macro-simulation of freeway traffic. *Mathematical and computer modelling* 35 (5-6): 517–547, 2002.

Ho, T. K. Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on.* Vol. 1. IEEE, pp. 278–282, 1995.

Kormáksson, M., Barbosa, L., Vieira, M. R., and Zadrozny, B. Bus travel time predictions using additive models. In *Data Mining (ICDM), 2014 IEEE International Conference on.* IEEE, pp. 875–880, 2014.

Kuncheva, L. I. and Whitaker, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning* 51 (2): 181–207, 2003.

Lee, W.-H., Tseng, S.-S., and Tsai, S.-H. A knowledge based real-time travel time prediction system for urban network. *Expert Systems with Applications* 36 (3): 4239–4247, 2009.

Lima, A. M. and Campos, J. How reliable is the traffic information gathered from web map services? In *GEOINFO.* pp. 234–245, 2017.

Mendes-Moreira, J., Jorge, A. M., de Sousa, J. F., and Soares, C. Improving the accuracy of long-term travel time prediction using heterogeneous ensembles. *Neurocomputing* vol. 150, pp. 428–439, 2015.

Monteiro, C. M., Martins, F. V. C., and Junior, C. A. D. Optimization of new pick-up and drop-off points for public transportation, 2017.

Opitz, D. and Maclin, R. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* vol. 11, pp. 169–198, 1999.

Polikar, R. Ensemble based systems in decision making. *IEEE Circuits and systems magazine* 6 (3): 21–45, 2006.

Reddy, K. K., Kumar, B. A., and Vanajakshi, L. Bus travel time prediction under high variability conditions. *Current Science (00113891)* 111 (4), 2016.

Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Review* 33 (1-2): 1–39, 2010.

Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

Sollich, P. and Krogh, A. Learning with ensembles: How overfitting can be useful. In *Advances in neural information processing systems.* pp. 190–196, 1996.

Sommer, C., German, R., and Dressler, F. Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing* 10 (1): 3–15, 2011.

Trawiński, B., Smętek, M., Telec, Z., and Lasota, T. Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. *International Journal of Applied Mathematics and Computer Science* 22 (4): 867–881, 2012.

Vanajakshi, L. and Rilett, L. R. A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed. In *Intelligent Vehicles Symposium, 2004 IEEE.* IEEE, pp. 194–199, 2004.

Vanajakshi, L. and Rilett, L. R. Support vector machine technique for the short term prediction of travel time. In *Intelligent Vehicles Symposium, 2007 IEEE.* IEEE, pp. 600–605, 2007.

Wen, X. A work zone simulation model for travel time prediction in a connected vehicle environment. *arXiv preprint arXiv:1801.07579*, 2018.

Wu, C.-H., Su, D.-C., Chang, J., Wei, C.-C., Ho, J.-M., Lin, K.-J., Lee, D., et al. An advanced traveler information system with emerging network technologies. In *Proc. 6th Asia-Pacific Conf. Intelligent Transportation Systems Forum.* pp. 230–231, 2003.

Yu, B., Yang, Z. Z., and Wang, J. Bus travel-time prediction based on bus speed. In *Proceedings of the Institution of Civil Engineers-Transport.* Vol. 163. Thomas Telford Ltd, pp. 3–7, 2010.