Soft Comput (2016) 20:4205–4216
DOI 10.1007/s00500-016-2288-6

FOCUS

A novel methodology to predict urban traffic congestion with ensemble learning

G. Asencio-Cortés[1] · E. Florido[1] · A. Troncoso[1] · F. Martínez-Álvarez[1]

Published online: 3 August 2016

Abstract Urban traffic congestion prediction is a very hot topic due to the environmental and economical impacts that currently implies. In this sense, to be able to predict bottlenecks and to provide alternatives to the circulation of vehicles becomes an essential task for traffic management. A novel methodology, based on ensembles of machine learning algorithms, is proposed to predict traffic congestion in this paper. In particular, a set of seven algorithms of machine learning has been selected to prove their effectiveness in the traffic congestion prediction. Since all the seven algorithms are able to address supervised classification, the methodology has been developed to be used as a binary classification problem. Thus, collected data from sensors located at the Spanish city of Seville are analyzed and models reaching up to 83 % are generated.

Keywords Traffic congestion · Prediction · Machine learning · Ensembles · Time series

1 Introduction

There are many factors contributing to an increasing bottleneck. One is the very composition of traffic (Wei Qu et al. 2012). A lane along with four trucks in a row does not have the same capacity as one that is only occupied by cars. In general, heavy vehicles travel slower and generate longer lines of vehicles. Another factor that negatively affects traffic is the interaction between cars. When traffic density is above 1700 vehicles per hour any overtaking or maneuvering causes a reduction in the speed of the road (Hernández et al. 2002). Branch road or slip road may also complicate matters in jam, as they bring an extra supply of vehicles to a road which, in itself, is already saturated.

Decreasing speed helps to prevent traffic jams. It allows the flow to move more evenly. If all vehicles driving down a road are at a low speed, the capacity of track circulation would not saturate so quickly. The mass of cars would move slowly but evenly. It is a very complex problem, and possibly, the most feasible solution would be to create more and better infrastructures. The problem is that as more roads and highways are built, the number of cars also increases, with eventually the same traffic issues arising. Different solutions have been attempted in cities around the world, with varying degrees of success (Carlson et al. 2010).

Some highways have added new lanes to extend them in a general way, using the hard shoulder or decreasing the size of the existing lanes if feasible. These adjustments are both expensive and time consuming. As it depends on the population growth and on the sales of vehicles, this is a medium-term solution (Ahmane et al. 2013).

A less costly solution would be to study how traffic behaves using information obtained from roadside sensors (Dunkel et al. 2011) based at particular segments of the road network, referencing these variables: (Lee et al. 2010) congestion, intensity, occupancy, average speed of vehicles, etc. Obtaining a model for traffic control varies depending on the area of study.

Traffic congestion in the Spanish city of Seville, in particular, on the bridge of the Fifth Centenary, is a latent problem. The huge volume of vehicles transiting in either direction may vary for different slots of time. Northbound traffic and southbound traffic are separated by a moveable barrier, which

Communicated by A. Herrero.

✉ F. Martínez-Álvarez
fmaralv@upo.es

[1] Division of Computer Science, Universidad Pablo de Olavide, 41013 Seville, Spain

permits the number of lanes allocated to northbound traffic or southbound traffic to be changed to accommodate heavy demand in either direction, to respond to unusual traffic conditions, and to balance lane availability with real-time traffic demand. It is impossible to leave the reversible lane from its beginning to its end by any alternative path. Therefore, the proper management of this lane and even the possibility of providing alternatives to traffic at peak times become an essential task.

This paper presents an approach of general-purpose methodology devoted to predict congestion of vehicles on roads, with application to the spot before mentioned. In particular, ensembles of well-known machine learning algorithms have been proposed. Four different scenarios of possible retention have been analyzed, those in which the anticipation of traffic retention was of 15, 30, 45, and 60 min. In these scenarios, on average, the results achieved an accuracy greater than 85 %. It is note worthy that this paper is an extension of that previously published in Florido et al. (2015), in which single soft computing methods were applied.

The rest of the paper is divided as follows. Section 2 reviews the most recent and relevant works related to traffic congestion prediction. Section 3 describes the proposed methodology to predict traffic congestion. The achieved results are presented and discussed in Sect. 4. Finally, the conclusions of this paper are drawn in Sect. 5.

2 Related works

Urban traffic-flow prediction is a task of utmost relevance to decrease traffic congestion in large-scale urban areas. For this reason, many prediction methods have been proposed and measured according to different metrics. In this sense, an interesting review that explores different methods and different metrics, with a short critique of measurement criteria, can be found in Rao and Rao (2012).

The prediction of traffic flow on the highway in St. Louis metropolitan area was addressed in Amin et al. (1998). The authors developed models based on radial basis function neural networks based on the historical data retrieved from nine different sensors. The results achieved outperformed those of ARIMA.

The work in Pescaru (2013) proposed a technique for congestion prediction in urban traffic. In particular, it used event-based routes selection and relied on information collected by a sensor network. Simulation experiments with more than 50 traffic patterns over eight crowded intersections demonstrated promising results.

Two urban traffic prediction models using different modeling approaches were introduced in Liang and Wakahara (2014). The first model was based on the traffic-flow propagation in the network, while the second one on the time-varied spare flow capacity on the concerned road links. Predictive errors were reduced up to 22 % when compared with the existing shift model.

The application of neuro-fuzzy techniques to predict traffic volume of vehicles on a busy traffic corridor was established in Ogunwolu et al. (2014). Using as case of study the city of Lagos, a traffic prediction system was designed based on acquired real-time traffic data. The network was trained and the fuzzifier module categorized the numerical output of the model. Information on estimated time and fuel consumption cost are also provided.

A deep architecture that consists of two parts, i.e., a deep belief network (DBN) at the bottom and a multitask regression layer at the top were presented in Huang et al. (2014). A DBN was employed for unsupervised feature learning, showing that it can learn effective features for traffic-flow prediction in an unsupervised fashion. Abundant experiments showed that the proposed model achieved close to 5 % improvements over other approaches found in the literature.

An study to attempt to extend deep learning theory into large-scale transportation network analysis can be found in Ma et al. (2015). A deep restricted Boltzmann machine and recurrent neural network architecture are utilized to model and predict traffic congestion evolution based on global positioning system (GPS) data from taxi. Results from a Chinese city showed that the prediction accuracy can reach 88 %.

An approach to predict the urban traffic congestion using floating car trajectory data efficiently was proposed in Kong et al. (2016). The authors used a new fuzzy comprehensive evaluation method, in which the weights of multi-indexes are assigned according to the traffic flows. The prediction was made using a particle swarm optimization algorithm, which calculates some traffic-flow parameters. The results achieved outperformed other methods in terms of accuracy, instantaneity, and stability.

An approach based on chaos theory for predicting motorized traffic flow on urban road networks was introduced in Adewumi et al. (2016). Nonlinear time series modeling techniques were used with emphasis on the largest Lyapunov exponent to aid in the prediction of traffic flow. Despite the results are promising, the authors did not perform any comparative study to evaluate its performance.

The paper in Li et al. (2016) proposed an adaptive data-driven real-time congestion prediction method. This framework included a traffic pattern recognition algorithm based on the adaptive K-means clustering, a two-dimensional speed prediction model, and an adaptive threshold calibration method. After the principal component analysis, the adaptive K-means cluster algorithm is conducted to obtain different traffic patterns. Congestion recognition is realized with an adaptive threshold calibration method and congestion prediction is then raised according to different traffic patterns. Parameter calibration and model evaluation were carried out on the proposed method using floating car travel speed data. The results showed better real-time performance, accuracy, and robustness than ARIMA model and Kalman filtering method.

Finally, a very recent survey of current urban traffic management schemes for priority-based signalling, and reducing congestion and the average waiting time of vehicles is accessible in Nellore and Hancke (2016). The main objective of this survey is to provide a taxonomy of different traffic management schemes used for avoiding congestion. Existing urban traffic management schemes for the avoidance of congestion and providing priority to emergency vehicles are considered and set the foundation for further research.

3 Methodology

This section describes the proposed methodology. First, the information provided by the stations of data collection is described in Sect. 3.1. Later, in Sect. 3.2, the base machine learning classifiers are listed along with a brief description of them. The procedure to create ensembles is detailed in Sect. 3.3. Finally, the validation process followed to assess the methodology performance is summarized in Sect. 3.4.

3.1 Information provided by the stations of data collection

The stations of data collection use a set of sensors installed in the roadway to detect the passage of vehicles. These data refer to the following traffic variables obtained from all vehicles registered during the integration period (IP) which, in this case, is set to 1 min, are summarized in Table 1.

Finally, information about future occurrence of traffic congestion is included in the class. This class, which is binary and only accepts 0 and 1 as possible values, is generated in a similar way as in Asencio-Cortés et al. (2015, 2016). That is, if congestion is to happen within the following preset minutes (in this study, those minutes are 15, 30, 45, or 60, as described in Sect. 4.3), then an 1 is assigned to the class. If not, 0 is assigned. The assignment of such values is determined by the HIOCC algorithm, developed by the Transport and Road Research Lab (TRRL) (Collins 1993) and which comes implemented in the station of data collection itself in the period of integration. It is expressed as YES = 1 or NO = 0. This variable will be the class label and, therefore, the one wanted to be correctly predicted.

Feature selection in the field of urban traffic congestion prediction is a relevant issue (Yang 2013). For this reason, several variables described in the previous section might not be considered. In particular, the variables Kamikaze alarm, reverse direction, average length, composition length, composition speed, and error have been discarded due of its null

Table 1 Description of the variables retrieved from the stations of data collection

Variables	Description
Intensity	Number of vehicles registered by the detector during the IP
Occupation	Arithmetic mean of the occupation time of the vehicles in the IP
Average speed	Arithmetic mean of the speed of the vehicles in the IP
Average distance	Arithmetic mean of the distance between two consecutive vehicles in the IP
Average length	Arithmetic mean of the length of the vehicles in the IP
Composition length	Number of vehicles classified by length in the IP
Composition speed	Number of vehicles classified by speed in the IP
Kamikaze alarm	Boolean variable indicating the existence of any vehicle circulating in wrong way during the IP
Reverse direction	Boolean variable indicating that the circulation in the IP has suffered a modification from the usual sense
Error	Time within the IP, in which if vehicles have passed the detector, but it did not registered them
Congestion	Boolean variable indicating the existence of activation of an alarm of congestion

relevance for this analysis. Therefore, four input variables were used to train models: intensity, occupation, average speed, and average distance.

3.2 Base machine learning algorithms

A set of seven algorithms of machine learning have been selected to evaluate their effectiveness in the traffic congestion prediction: nearest neighbors (K-NN), C4.5 decision trees (C4.5), artificial neural networks (ANN), stochastic gradient descent optimization (SGD), a fuzzy unordered rule induction algorithm (FURIA), Bayesian networks (BN), and support vector machines (SVM). All the seven algorithms are able to address a classification problem; therefore, they are called classifiers in this work. Specifically, the data sets used to validate the performance of traffic congestion prediction have two classes (binary classification).

Since results easily interpretable are desired, the capability of a machine learning algorithm of producing a viewable knowledge model is also a relevant property to select the most suitable for this problem. Three out of the seven classifiers generate an interpretable model: C4.5 (trees), FURIA (rules), and BN (graphs).

Below, there is a short description of each of the seven algorithms studied as base classifiers. In first place, K-NN algorithm (Aha et al. 1991) belongs to the case-base reasoning group of machine learning and it is based on the proximity between the sample to be classified and the samples that form the training set. Therefore, it is a deterministic algorithm whose critical point is to select the number of neighbors to use. The most common strategies simply choose the closest instance (the sample belonging to the training set that gives the shortest distance from the sample that you want to be labeled) or take an odd number of neighbors to ensure no tie between their classes, whether it is a binary classification.

The C4.5 algorithm (Quinlan 1993; Salzberg 1994) is a method for generating a decision tree, which consists in selecting an attribute as root of the tree and creates a branch for each of the possible values for that attribute. With each resulting branch (new node of the tree), the same process is done, another attribute is selected and a new branch for each possible attribute value is generated. This procedure continues until the samples are classified through one of the paths of the tree. The end node of each path is a leaf node, to which the corresponding class is assigned.

Therefore, the objective of the decision trees is obtaining rules or relationships that allow classification based on the attributes. This algorithm allows the use of the concept of information gain, build decision trees when some of the samples show unknown values for some of the attributes, working with attributes that have continuous values, decision tree pruning, and obtaining classification rules.

ANNs (McCulloch and Pitts 1943) are a family of statistical learning algorithms inspired by biological neural networks that are used to estimate or approximate functions that may depend on a large number of inputs and are generally unknown. ANNs are generally presented as systems of interconnected neurons that can calculate values from inputs, and are able of machine learning and pattern recognition, thanks to its adaptive nature.

SGD (Duchi et al. 2011) is an optimization technique that can be used for classification purposes by learning various linear models (SVM with linear kernel was used according to the binary class of the data sets). For SVM, the error function (to be minimized) is continuous and it is called the hinge loss. The SGD procedure is computationally simple and converges fast, allowing models, such as linear SVM or logistic regression, to be learned from high-dimensional data sets.

FURIA (Hühn and Hüllermeier 2009) is based on the rule learner RIPPER algorithm (Cohen 1995), but it obtains fuzzy rules. The rules are put into unordered sets and it makes use of an efficient rule stretching to deal with uncovered instances. To obtain fuzzy rules, FURIA generates the conventional rules from a modified RIPPER, and then, it searches for the best fuzzy extension of each rule, where a fuzzy extension is a rule of the same structure, but with intervals replaced by fuzzy intervals. The fuzzification is then realized for the antecedent with the largest purity. This process is repeated until all antecedents have been fuzzified.

BN (Friedman et al. 1997) are probabilistic machine learning algorithms that construct networks based on the Bayes' theorem. The algorithm is based on two components: a function for evaluating a given network based on the data and a method for searching through the space of possible networks. The quality of a network is measured by the probability of the data given such network. The probability that the network accords to each instance is calculated and these probabilities are multiplied together over all the instances. The BN algorithm is based on the independency of the attributes and it was designed for classification tasks.

SVM (Schölkopf and Smola 2002) is a binary classifier for numeric data that use separating hyperplanes as the decision boundary between the two classes. The optimization problem of determining these hyperplanes is set up with the notion of margin. A maximum margin hyperplane is one that cleanly separates the two classes, and for which a large region exists on each side of the boundary with no training data points in it. In linearly separable data, it is possible to construct a linear hyperplane which cleanly separates data points belonging to the two classes.

The implementation of the algorithms of this study, as well as their parameters setup, are those included into the free data mining tool Weka (Hall et al. 2009).

Once the algorithms have been described, the particular configuration used in the experimentation is now described. For K-NN, the number of neighbors (K) was set to 1 and the Euclidean distance was used without any type of distance weighting. In the C4.5 algorithm, the confidence factor for prunning was set to 0.25 and the minimum number of instances covered by the leaf nodes was 2. A minimum description length correction was used when finding splits on the numeric attributes to avoid large and, therefore, less interpretable tree models.

The ANN algorithm was configured as follows. A multilayer perceptron using the backpropagation technique wa,s used. The network has three layers: an input layer, a hidden layer and an output layer. According to the number of input attributes, there were four neurons at the input layer. There was one hidden layer with three neurons. Finally, the output layer has two neurons, according to the two possible classes (traffic congestion or non-congestion). The learning rate, that determines the step size when searching for a solution and hence how quickly the search converges, was set to 0.3.

In the SGD algorithm, the loss function was set to the hinge loss (linear SVM), the number of iterations was set to 500, the regularization constant to 0.0001, and the learning rate was set to 0.01.

In the FURIA algorithm, the T-norm of the fuzzy part was established as the product T-norm, threefolds were used for pruning the rules, the minimum total weight of instances in rules was 2 and the number of optimization runs was set to 2.

For BN, a genetic search with tournament selection was used as search algorithm and a simple estimator is used for estimating the conditional probability tables of the Bayes' network (with $\alpha = 0.5$).

The SVM implementation used is the included in the LibSVM (Chang and Lin 2011) free package and it was configured as follows. The SVM type was the C-SVC for classification tasks, the kernel was the polynomial of degree 1, and the cost parameter C was set to 1.

3.3 Ensemble algorithms

The ensemble learning is a paradigm of machine learning motivated by the fact that different classifiers may make different predictions due to the specific characteristics of the classifier, or their sensitivity to the random issues in the training data. An ensemble algorithm is an approach to increase the prediction accuracy by combining the results from multiple classifiers. The basic approach of ensemble analysis is to apply the aggregated classifiers (internal classifiers) multiple times using either different models, or using the same model on different subsets of the training data. The results from different classifiers are then combined into a single-robust prediction.

Ensemble algorithms intend to improve the accuracy of a classifier by reducing their bias and variance. Some ensemble methods, such as bagging and Random Forests (RF), are designed only to reduce the variance, whereas other ensemble methods, such as boosting and stacking, can help reduce both the bias and variance. In some cases, such as boosting, it is possible for the ensemble to overfit the noise in the data and thereby lead to lower accuracy.

To select a set of representative algorithms of ensemble learning, three classic approaches were considered to perform a comparative study: bagging, boosting, and stacking. Moreover, a meta-learning-based algorithm that selects best probability thresholds to classify binary classes, named probability threshold selector (PTS), was also included in the ensemble algorithms group of study in this work. The objective is to check whether some base classifiers might be improved using ensemble techniques in the traffic congestion prediction problem.

The bagging technique (Breiman 1996; Kuncheva 2004) is based on a simple approach to vote the predictions of a set of classifiers. The procedure, in general terms, is the following. The training set is divided in several training subsets using a bootstrap re-sampling, each one with the same number of examples. Next, from each subset, a classifier is trained, and given a test set, a prediction for each classifier is requested. The final class is computed by the aggregation of the predictions of all classifiers. Such aggregation generally consists of a simple voting assembly (e.g., the majority predicted class).

Although the bagging technique achieves generally best results than the base classifiers, those have been aggregated independently (each one is trained with random samples). For this reason, the performance of bagging strongly depends on the ability of the classifiers to complement among them; this means that the individual errors of a classifier can be corrected by the others. Therefore, combining multiple classifiers only could achieve an improvement in effectiveness when their models are significantly different from one another and each one classifies correctly a reasonable percentage of the data set.

The baseline configuration for the bagging algorithm assumes the same internal classifiers to perform the voting assembly. In this work, the bagging technique was applied using both the same internal classifier and different ones. Each of the seven base classifiers described in Sect. 3.2 was the internal classifier for the bagging algorithm. In addition, the Random Forests (Breiman 2001) algorithm, which is a bagging of decision trees, was also been included in the comparison study.

The boosting approach iteratively seeks for models that complement well among them covering with correct predictions the maximum amount of data. Unlike the bagging technique, in the boosting procedure, each new model depends on the performance of those built previously. The instances incorrectly predicted in each iteration are assigned with a greater weight and importance to be predicted correctly in the further iterations.

The selected boosting algorithm was the AdaBoost M1 (Freund and Schapire 1996). This algorithm begins by assigning equal weight to all instances in the training data. Next, the internal classifier is trained with these instances, and those are reweighted according to the classifier output. The weight of correctly classified instances is decreased, and the weight of those that are misclassified is increased. In the further iterations, classifiers are trained with the reweighted data, which consequently focuses on classifying the hard instances correctly.

In addition to the algorithm AdaBoost M1, a Logistic Model Trees (LMT) learning algorithm (Landwehr et al. 2005) was analyzed and included in the comparison study, providing a method that produces an interpretable knowledge model based on trees. LMT is based on the LogitBoost algorithm (Friedman et al. 2000), and it induces trees with linear-logistic regression models at the leaves.

The stacking class of ensembles arises from the idea of non-homogeneous boosting that takes the outputs of a group of classifiers and produces the final classification by the weighted majority voting. In this procedure, two layers are

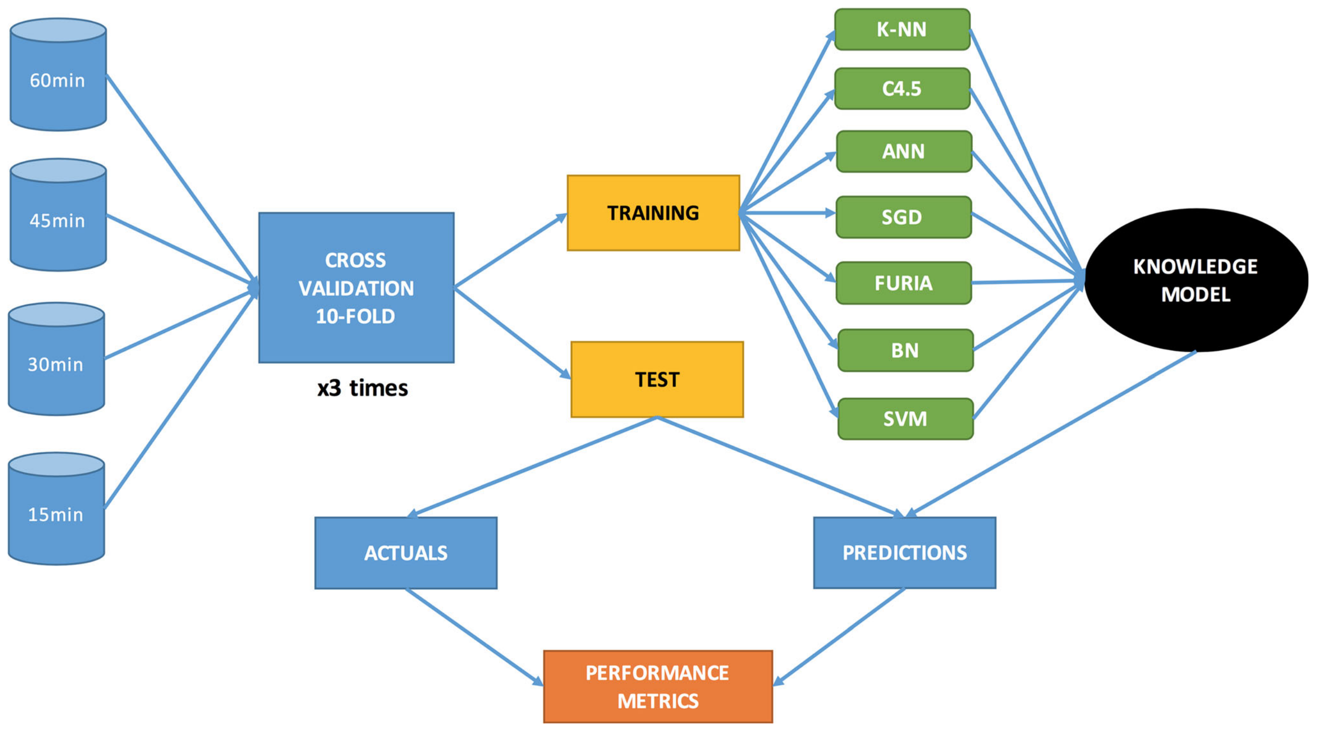

Fig. 1 Methodology applied to validate the seven base classifiers

involved: at the lower level are the base classifiers, and at the upper level is a master classifier combining their outputs. In non-homogeneous boosting, a linear classifier is used at the upper level. This architecture is the base of the stacking approach.

The stacking technique takes a set of diverse classifiers used at the lower level. At the upper level, instead of using only a linear classifier, any general classifier can be used. In fact, both the base classifiers and the master may come from the most diverse paradigms.

The stacking algorithm used in this study was the published in Wolpert (1992). Each of the seven base classifiers described in Sect. 3.2 was probed to be the master classifier for the stacking algorithm. For each master classifier, the six remaining classifiers act as lower level classifiers in the stacking.

Finally, a meta-learning based algorithm named PTS was also included in the study. PTS selects a mid-point threshold on the probability output by a base classifier. The mid-point threshold is set, so that a given performance measure is optimized. The F-measure was used to provide such a measure. Performance is measured using a tenfold cross-validation over the training data. In addition, the probabilities returned by the base learner can have their range expanded, so that the output probabilities will reside between 0 and 1. Each of the seven base classifiers described in Sect. 3.2 was probed as the base classifier for the PTS algorithm. The PTS implementation used was the included in the Weka external package named ThresholdSelector.

3.4 Validation process

To validate the effectiveness of the classifiers, a tenfold cross-validation was performed three times for each of the four data sets of this study (60, 45, 30, and 15 min). This is a classic procedure to validate classifiers in machine learning, and it consists of the following steps. First, each data set is divided in both training and test subsets. Second, the training set was used to train the classifier and to produce its knowledge model. Then, such model was used to predict the instances of the test set, and finally, the resulting predictions were compared to the actual test classes producing the performance metrics.

The process of training the classifier may imply different steps depending on the type of the classifier. Thus, the base classifiers studied in this work follow the simple training process described before, as it can be seen in Fig. 1. However, ensemble methods generally imply an iterative subprocess to train the internal classifiers, in which their predictions are based on. Specifically, the algorithms that belong to the bagging and boosting ensemble approaches may need to perform a determined number of iterations over their internal classifiers, as shown in Fig. 2. In particular, the bagging and boosting algorithms used were trained with all the seven base

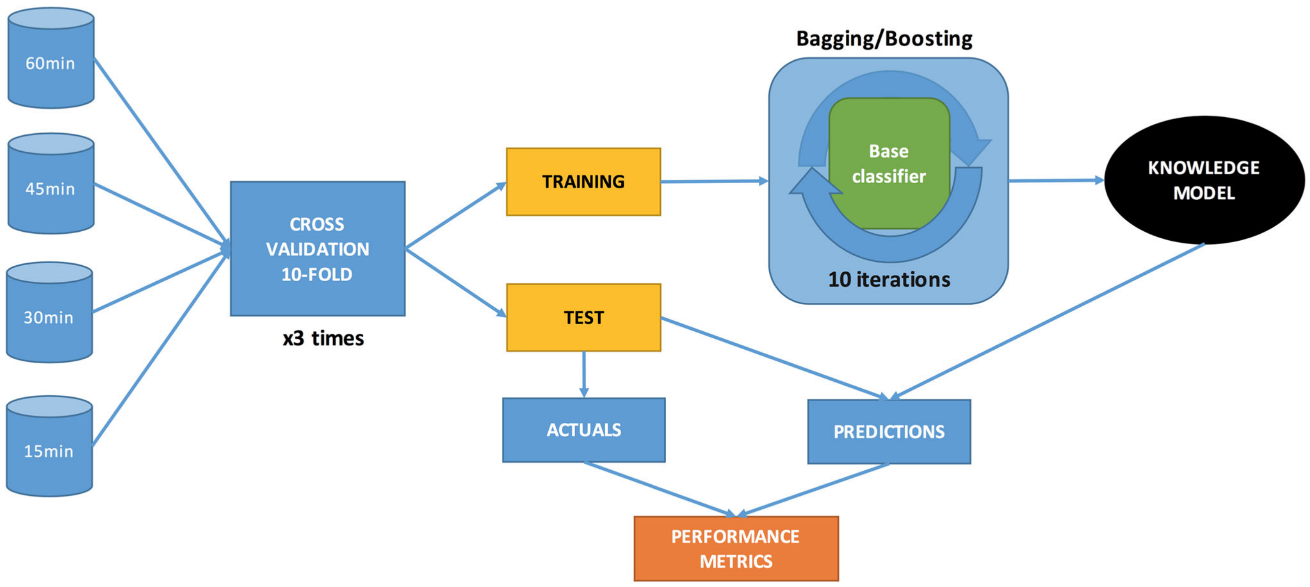

Fig. 2 Methodology applied to validate the bagging and boosting algorithms

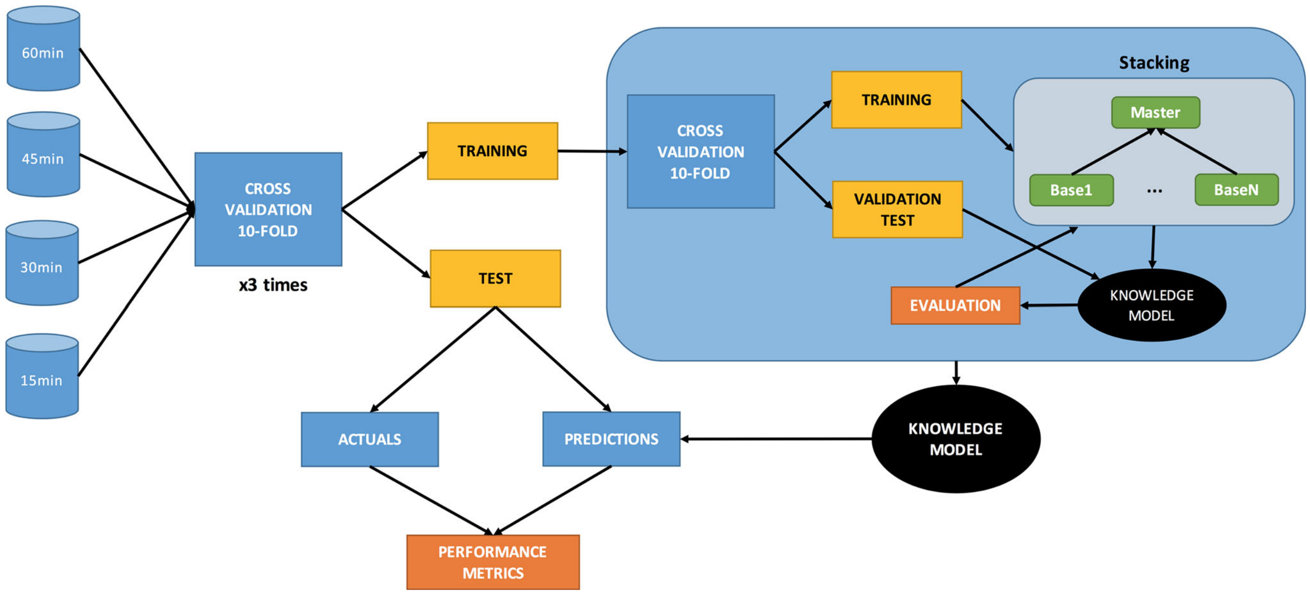

Fig. 3 Methodology applied to validate the stacking algorithm

classifiers described in Sect. 3.2, that is, one experiment for each one of them. During the ensemble training process, ten iterations were performed.

The stacking and PTS algorithms perform an internal cross-validation (using tenfold) to conduct the training of the base classifiers and to tune the consensus of their predictions. This process is shown in the Figs. 3 and 4.

4 Experiments

In this section, the data sets used to train and test the classifiers are explained. Next, the parameters used to evaluate the quality of the results are presented. Finally, the results produced in the experimentation are shown and discussed.

4.1 Data sets description

For this study, data were obtained from the detector during the last quarter of 2011 using a total of 8926 records for testing, and 27,321 for training. In addition, the *Congestion* variable has been taken as class label attribute, since this attribute takes values YES = 1 or NO = 0. As data mining techniques have been applied to various intervals of time, the class has been moved to analyze the behavior at 15, 30, 45, and 60 min.

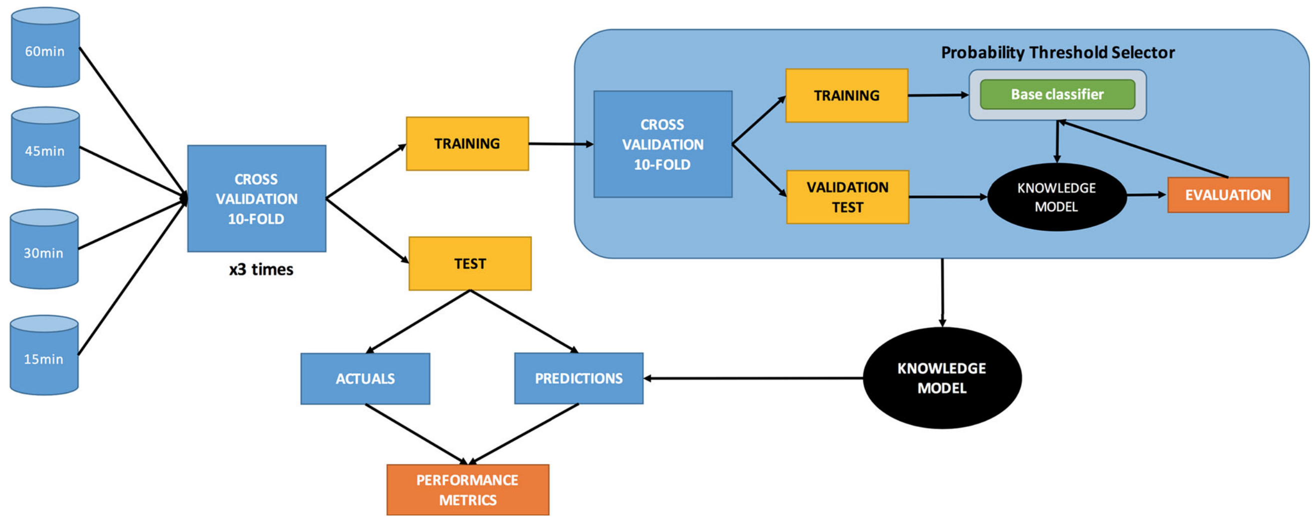

Fig. 4 Methodology applied to validate the PTS algorithm

That is, for each of the four scenarios evaluated, the class label corresponds to whether there will be or not congestion in the next 15, 30, 45, or 60 min.

4.2 Quality parameters

The quality parameters used to evaluate the different methods are now presented. It was decided to use the most common ones in the literature. On the one hand, a set of indicators that quantify the successes and mistakes of classifiers is calculated:

1. True positive, TP. It is defined as the number of times the classifier assigns a 1 to the instance that is being classified (predicts the occurrence of a retention), and this, indeed, happens during the next few minutes.
2. True negative, TN. It is the number of times which has been predicted that retention does not occur during the next minutes, and, in fact, it does not.
3. False positive, FP. The number of times has been erroneously detected a retention in the next minutes, that is, the number of times the classifier assigns a label with value 1 when really ought to assign a 0.
4. False negative, FN. The number of times a retention has not been detected, but, indeed, there was a retention within the next several minutes.

On the other hand, from these four indicators, the quality parameters are properly calculated. In particular:

1. Recall (also known as sensitivity). It is the proportion of correctly identified retentions, regardless the value of the FP. Mathematically, it is expressed as:

$$S = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{1}$$

2. Area under the curve, AUC. It is defined as the definite integral of the receiver-operating characteristic (ROC), on the interval [0, 1], where ROC identifies the graphic presentation of the relationship between both sensitivity and specificity. Formally:

$$\mathrm{AUC} = \int_0^1 \mathrm{ROC}(t)\mathrm{d}t. \tag{2}$$

3. Precision (also known as positive predictive value, PPV). This value measures the reliability of the TP, that is, the certainty associated with each TP. In other words, the relationship between TP and FP is, mathematically, formulated as:

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}. \tag{3}$$

4. Matthew's correlation coefficient, MCC. It was proposed by Matthews in 1975 (Matthews 1975). It gives the balanced measure of TP, FP, TN, and FN. Mathematically,

$$\mathrm{MCC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TN} + \mathrm{FN})(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})}}. \tag{4}$$

4.3 Results

After the validation process explained in the methodology section, the results for each type of experimentation are discussed in the following subsections. For each experimentation, a non-parametric statistical test was performed to

Table 2 Validation results for the base classifiers

Horizon (min)	Parameter	K-NN	C4.5	ANN	SGD	FURIA	BN	SVM
60	MCC	0.75	0.77	**0.82**	0.78	0.77	0.76	0.79
	AUC	0.89	0.95	**0.99**	0.88	0.92	0.98	0.88
	Precision	0.97	0.97	0.98	0.97	0.97	0.97	0.97
	Recall	0.97	0.97	0.98	0.98	0.98	0.97	0.98
45	MCC	0.26	0.26	0.30	0.11	**0.32**	0.27	0.02
	AUC	0.66	0.86	**0.89**	0.51	0.67	0.88	0.50
	Precision	0.91	0.89	0.90	0.88	0.91	0.89	0.88
	Recall	0.92	0.99	0.97	1.00	0.97	0.99	1.00
30	MCC	0.37	0.43	0.44	0.45	**0.48**	0.44	0.45
	AUC	0.71	0.89	**0.92**	0.69	0.76	0.91	0.68
	Precision	0.92	0.92	0.92	0.92	0.93	0.92	0.92
	Recall	0.93	0.96	0.97	0.97	0.95	0.96	0.97
15	MCC	0.42	0.49	0.53	0.54	**0.55**	0.51	0.54
	AUC	0.74	0.91	**0.94**	0.74	0.80	0.94	0.74
	Precision	0.93	0.92	0.94	0.94	0.94	0.93	0.94
	Recall	0.94	0.98	0.96	0.96	0.95	0.97	0.97

determine whether the differences in effectiveness among the classifiers are statistically significant. The MCC was selected as the metric to compare the effectiveness in the statistical tests.

Specifically, the Kruskal–Wallis and Friedman tests have been performed, as well as a post-hoc test using Mann–Whitney tests with Bonferroni correction. Such analyses have been carried out using the free on-line tool STATService (Parejo Maestre et al. 2012). To have enough data to carry out the statistical tests, 30 values of the MCC were taken for each classifier and data set (one value per fold and execution; tenfolds $\times$ three executions).

4.3.1 Validation results for base classifiers

All the results for the base classifiers and different horizon of predictions are shown in Table 2. ANN achieves effectiveness significantly better than the other base classifiers in terms of the AUC measure for all data sets. With respect to the hardest data sets (45, 30 and 15 min), the FURIA algorithm performed significantly better in terms of MCC values. Moreover, FURIA is able to produce an interpretable model based on fuzzy rules.

Therefore, regarding the results achieved by the base classifiers, the best election, in terms of effectiveness and interpretability, has been found to be the FURIA algorithm for all the data sets. It is also important to highlight that the choice of this algorithm, since an easy interpretation of the results is highly desired. In this sense, the structure shown by the rules is of immediate interpretation for any professional who is not related to data mining.

4.3.2 Validation results for ensemble classifiers

The bagging, boosting, stacking, and PTS approaches were applied to the different data sets of study, producing significant improvements with respect to the performance achieved by the base classifiers. However, these improvements are only in the hardest data sets (45, 30, and 15 min). For the 60 min data set, no significant differences with the base classifiers were found.

The results for the bagging ensembles are shown in Table 3. Note that the bagging technique was applied to the seven base classifiers. In addition, the random forests (RF) algorithm is placed in the rightmost column of the table. The results for the boosting ensembles are shown in Table 4. The boosting technique was also applied to the seven base classifiers. Moreover, the logistic model trees (LMT) algorithm is placed at the end of the table. Finally, the results for the stacking and PTS ensembles are shown in Tables 5 and 6.

Specifically, for the 45 min data set, the MCC was significantly improved from 0.32 (achieved by FURIA) to 0.46 (achieved by a bagging over BN). Regarding the 30 min data set, the MCC was significantly improved from 0.48 (achieved by FURIA) to 0.52 (achieved by the PTS technique over BN). For the 15 min data set, the MCC was improved, with statistically significance, from 0.55 (achieved by FURIA) to 0.58 (achieved by the PTS technique over ANN).

It can be also noticed that the classifier BN is the algorithm that improves the more its accuracy when it was wrapped by an ensemble technique, specially when automatically selecting a threshold for its classification probability.

Moreover, regarding the precision and recall values using the ensemble techniques, although recall values were gener-

Table 3 Validation results for the bagging-based ensembles

Horizon (min)	Parameter	K-NN	C4.5	ANN	SGD	FURIA	BN	SVM	RF
60	MCC	0.76	0.80	**0.83**	0.78	0.78	0.72	0.78	0.79
	AUC	0.95	0.99	**0.99**	0.90	0.97	0.98	0.89	0.98
	Precision	0.97	0.97	0.98	0.97	0.97	0.99	0.97	0.97
	Recall	0.97	0.98	0.98	0.98	0.98	0.92	0.98	0.98
45	MCC	0.28	0.28	0.29	0.10	0.35	**0.46**	0.01	0.28
	AUC	0.76	0.88	0.89	0.55	0.76	**0.89**	0.53	0.85
	Precision	0.91	0.90	0.90	0.88	0.91	0.98	0.88	0.91
	Recall	0.92	0.97	0.98	1.00	0.95	0.78	1.00	0.94
30	MCC	0.38	0.42	0.46	0.45	**0.49**	0.48	0.44	0.39
	AUC	0.79	0.91	**0.92**	0.71	0.78	0.91	0.70	0.88
	Precision	0.92	0.92	0.92	0.92	0.94	0.98	0.92	0.92
	Recall	0.93	0.96	0.96	0.97	0.95	0.79	0.97	0.94
15	MCC	0.43	0.48	0.52	0.54	0.54	**0.56**	0.54	0.46
	AUC	0.84	0.93	0.94	0.76	0.85	**0.94**	0.75	0.91
	Precision	0.93	0.93	0.93	0.94	0.94	0.98	0.93	0.93
	Recall	0.94	0.96	0.97	0.97	0.95	0.84	0.97	0.95

Table 4 Validation results for the boosting-based ensembles

Horizon (min)	Parameter	K-NN	C4.5	ANN	SGD	FURIA	BN	SVM	LMT
60	MCC	0.75	0.79	0.82	0.78	0.77	0.77	0.78	**0.82**
	AUC	0.87	0.98	0.97	0.96	0.89	0.98	0.96	**0.99**
	Precision	0.97	0.97	0.98	0.97	0.97	0.97	0.97	0.98
	Recall	0.97	0.98	0.98	0.98	0.97	0.98	0.98	0.98
45	MCC	0.28	0.29	0.24	0.28	**0.32**	0.27	0.25	0.29
	AUC	0.64	0.86	0.87	0.87	0.63	0.88	0.87	**0.89**
	Precision	0.91	0.91	0.89	0.90	0.91	0.90	0.90	0.90
	Recall	0.91	0.93	0.98	0.96	0.97	0.98	0.97	0.98
30	MCC	0.38	0.38	0.45	0.46	**0.49**	0.42	0.45	0.43
	AUC	0.69	0.88	0.90	0.90	0.74	0.91	0.90	**0.92**
	Precision	0.93	0.92	0.92	0.93	0.94	0.92	0.92	0.92
	Recall	0.92	0.93	0.96	0.96	0.94	0.96	0.97	0.97
15	MCC	0.42	0.44	**0.55**	0.54	0.55	0.50	0.53	0.52
	AUC	0.71	0.91	**0.93**	0.93	0.76	0.93	0.93	0.93
	Precision	0.93	0.93	0.94	0.94	0.94	0.93	0.93	0.93
	Recall	0.93	0.94	0.95	0.96	0.96	0.96	0.97	0.97

ally lower, the precision was increased. Such result is relevant for traffic congestion prediction, where it is desirable to achieve more precise forecasts even though the coverage was lower, due to the high cost derived from false alarms.

5 Conclusions

The issue of predicting urban traffic congestion has been addressed in this paper. A novel methodology has been proposed, which turns the prediction problem into a binary classification one. Thanks to this transformation, seven well-known base classifiers have been used. In addition, ensembles have been proposed, not only by combining the seven aforementioned methods, but also including methods that are ensembles themselves. To assess the performance of the approach, four different scenarios have been created: prediction of traffic congestion 15, 30, 45, and 60 min ahead. In particular, data from Seville (Spain) have been analyzed and used for predictive purposes. Several ensembles have achieved successful results, with accuracy (MCC) up to 83 % for the 60 min horizon of predictions considered. Neverthe-

Table 5 Validation results for the stacking-based ensembles

Horizon (min)	Parameter	K-NN	C4.5	ANN	SGD	FURIA	BN	SVM
60	MCC	0.75	0.79	0.78	0.77	**0.83**	0.78	0.83
	AUC	0.88	0.94	0.98	0.95	**0.93**	0.95	0.92
	Precision	0.97	0.98	0.97	0.97	0.98	0.97	0.98
	Recall	0.97	0.97	0.98	0.98	0.98	0.97	0.98
45	MCC	0.28	0.25	0.24	0.27	**0.37**	0.25	0.13
	AUC	0.64	0.83	0.83	**0.84**	0.73	0.83	0.53
	Precision	0.91	0.89	0.89	0.90	0.92	0.90	0.89
	Recall	0.91	0.98	0.97	0.96	0.94	0.97	0.99
30	MCC	0.31	0.45	0.43	0.44	**0.46**	0.44	0.39
	AUC	0.65	0.75	0.76	0.80	**0.84**	0.81	0.78
	Precision	0.92	0.92	0.93	0.94	0.94	0.92	0.90
	Recall	0.92	0.97	0.96	0.94	0.93	0.97	0.92
15	MCC	0.43	0.53	0.52	0.51	**0.56**	0.52	0.52
	AUC	0.71	0.80	0.79	0.80	**0.83**	0.78	0.79
	Precision	0.93	0.93	0.92	0.92	0.95	0.93	0.92
	Recall	0.93	0.97	0.96	0.96	0.95	0.97	0.95

Table 6 Validation results for the PTS ensembles

Horizon (min)	Parameter	K-NN	C4.5	ANN	SGD	FURIA	BN	SVM
60	MCC	0.75	0.79	0.82	**0.83**	0.79	0.78	0.76
	AUC	0.89	0.95	**0.99**	0.91	0.92	0.98	0.89
	Precision	0.97	0.98	0.98	0.98	0.98	0.98	0.97
	Recall	0.97	0.97	0.97	0.98	0.97	0.97	0.96
45	MCC	0.28	0.42	**0.44**	0.20	0.37	0.43	0.29
	AUC	0.67	0.86	**0.89**	0.58	0.68	0.88	0.71
	Precision	0.91	0.94	0.96	0.90	0.92	0.95	0.91
	Recall	0.90	0.89	0.83	0.98	0.94	0.88	0.89
30	MCC	0.38	0.50	0.50	0.44	0.46	**0.52**	0.41
	AUC	0.71	0.89	0.92	0.68	0.76	**0.92**	0.69
	Precision	0.93	0.94	0.94	0.92	0.94	0.94	0.92
	Recall	0.92	0.94	0.94	0.97	0.94	0.94	0.95
15	MCC	0.42	0.56	**0.58**	0.54	0.58	0.57	0.49
	AUC	0.74	0.91	**0.94**	0.74	0.80	0.94	0.79
	Precision	0.93	0.95	0.97	0.94	0.95	0.96	0.96
	Recall	0.94	0.93	0.91	0.97	0.94	0.92	0.93

less, all ensembles reached satisfactory results, leading to conclude that the methodology designed in, in general, robust for this problem. Future work is directed towards the inclusion of new attributes as well as for discovering how urban traffic congestion may affect to different spots, that is, how congestion is propagated through different routes.

Acknowledgments This study was funded by the Spanish Ministry of Economy and Competitiveness and by the Junta de Andalucía under projects TIN2014-55894-C2-R and P12-TIC-1728, respectively.

Compliance with ethical standards

Conflict of interest G. Asencio-Cortés declares that he has no conflict of interest. E. Florido declares that he has no conflict of interest. A. Troncoso declares that she has no conflict of interest. F. Martínez-Álvarez declares that he has no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Adewumi A, Kagamba J, Alochukwu A(2016) Application of chaos theory in the prediction of motorised traffic flows on urban networks. Mathematical Problems in Engineering, ID5656734:1–15

Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. Machine learning 6(1):37–66

Ahmane M, Abbas-Turki A, Perronnet F, Wu J, El-Moudni A, Buisson J, Zeo R (2013) Modeling and controlling an isolated urban intersection based on cooperative vehicles. Transportation Research Part C: Emerging Technologies 28:44–62

Amin SM, Liu A-P, Rodin EY, Rink K, García-Ortiz A (1998) Traffic prediction and management via RBF neural nets and semantic control. Computer-Aided Civil and Infrastructure Engineering 13(5):315–327

Asencio-Cortés G, Martínez-Álvarez F, Morales-Esteban A, Reyes J, Troncoso A (2015) Improving earthquake prediction with principal component analysis: Application to chile. Lecture Notes in Artificial Intelligence 9121:393–404

Asencio-Cortés G, Martínez-Álvarez F, Reyes J, Morales-Esteban A, Reyes J (2016) A sensitivity study of seismicity indicators in supervised learning to improve earthquake prediction. Knowledge-Based Systems 101:15–30

Breiman L (1996) Bagging predictors. Machine learning 24(2):123–140

Breiman L (2001) Random forests. Machine learning 45(1):5–32

Carlson RC, Papamichail I, Papageorgiou M, Messmer A (2010) Optimal mainstream traffic flow control of large scale motorway networks. Transportation Research Part C: Emerging Technologies 18:193–212

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

Cohen WW (1995) Fast effective rule induction. In Proceedings of the International Conference on Machine Learning, pages 115–123

Collins JF (1993) Automatic incident detection: Experience with TRRL algorithm HIOCC. TRRL Supplementary Report 775:1–6

Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research 12:2121–2159

Dunkel J, Fernandez A, Ortiz R, Ossowski S (2011) Event-driven architecture for decision support in traffic management systems. Expert Systems with Applications 38:6530–6539

Florido E, Castaño O, Troncoso A, Martínez-Álvarez F(2015) Data mining for predicting traffic congestion and its application to spanish data. In *Proceedings of the International Conference of Soft Computing Models in Industrial and Environmental Applications*, pages 341–352

Freund Y, Schapire RE et al (1996) Experiments with a new boosting algorithm. In Proceedings of the International Conference on Machine Learning 96:148–156

Friedman J, Hastie T, Tibshirani R et al (2000) Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics 28(2):337–407

Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Machine learning 29(2–3):131–163

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1):10–18

Hernández JZ, Ossowski S, García-Serrano A (2002) Multiagent architectures for intelligent traffic management system. Transportation Research Part C: Emerging Technologies 10(5):473–506

Huang W, Song G, Hong H, Xie K (2014) Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. IEEE Transactions on Intelligent Transportation Systems 15(5):2191–2201

Hühn J, Hüllermeier E (2009) FURIA: an algorithm for unordered fuzzy rule induction. Data Mining and Knowledge Discovery 19(3):293–319

Kong X, Xu Z, Shen G, Wang J, Yang Q, Zhang B (2016) Urban traffic congestion estimation and prediction based on floating car trajectory data. Future Generation Computer Systems 61:97–107

Kuncheva LI (2004) Combining pattern classifiers: methods and algorithms. John Wiley & Sons,

Landwehr N, Hall M, Frank E (2005) Logistic model trees. Machine Learning 59(1–2):161–205

Lee WH, Tseng SS, Shieh WY (2010) Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system. Information Sciences 180:62–702

Li F, Gong J, Liang Y, Zhou J (2016) Real-time congestion prediction for urban arterials using adaptive data-driven methods. Multimedia Tools and Applications 13:1–20

Liang Z, Wakahara Y (2014) Real-time urban traffic amount prediction models for dynamic route guidance systems. EURASIP Journal on Wireless Communications and Networking 85:1–13

Ma X, Yu H, Wang Y, Wang Y (2015) Large-scale transportation network congestion evolution prediction using deep learning theory. PLoS ONE 10(3):1–17

Matthews BW (1975) Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochimica et Biophysica Acta - Protein Structure 405:442–451

McCulloch WS, Pitts W (1943) A logical calulus of ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5:115–133

Nellore K, Hancke GP (2016) A survey on urban traffic management system using wireless sensor networks. Sensors 16:1–25

Ogunwolu L, Adedokun O, Orimoloye O, Oke SA (2014) A neuro-fuzzy approach to vehicular traffic flow prediction for a metropolis in a developing country. Journal of Industrial Engineering International 7(13):52–66

Parejo Maestre JA, García J, Ruiz-Cortés A, Riquelme JC (2012) Statservice: Herramienta de análisis estadístico como soporte para la investigación con metaheurísticas. In Actas del VIII Congreso Expañol sobre Metaheurísticas, Algoritmos Evolutivos y Bio-inspirados, pp 1–8

Pescaru D (2013) Urban traffic congestion prediction based on routes information. In: Proceedings of the IEEE International Symposium on Applied Computational Intelligence and Informatics, pages 121–126

Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers,

Rao AM, Rao KM (2012) Measuring urban traffic congestion - a review. International Journal for Traffic and Transport Engineering 2(4):286–305

Salzberg SL (1994) C4. 5: Programs for machine learning. Machine Learning 16(3):235–240

Schölkopf B, Smola AJ (2002) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press,

Wei Qu Z, Xing Y, Song XM, Duan YZ, Wei F (2012) A study on the coordination of urban traffic control and traffic assignment. Discrete Dynamics in Nature and Society 2012(12):367–368

Wolpert DH (1992) Stacked generalization. Neural networks 5(2):241–259

Yang S (2013) On feature selection for traffic congestion prediction. Transportation Research Part C: Emerging Technologies 26:160–169