Intelligent Data Analysis 16 (2012) 427–449
DOI 10.3233/IDA-2012-0532
IOS Press

Comparing state-of-the-art regression methods for long term travel time prediction

João Mendes-Moreira[a,e,*], Alípio Mário Jorge[b,e], Jorge Freire de Sousa[c,f] and Carlos Soares[d,g]
[a]*Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal*
[b]*Faculdade de Ciencias, Universidade do Porto, Porto, Portugal*
[c]*Departamento de Engenharia Industrial e Gestão, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal*
[d]*Faculdade de Economia, Universidade do Porto, Porto, Portugal*
[e]*LIAAD-INESC TEC, Porto, Portugal*
[f]*UGEI-INESC TEC, Porto, Portugal*
[g]*UESP-INESC TEC, Porto, Portugal*

Abstract. Long-term travel time prediction (TTP) can be an important planning tool for both freight transport and public transport companies. In both cases it is expected that the use of long-term TTP can improve the quality of the planned services by reducing the error between the actual and the planned travel times. However, for reasons that we try to stretch out along this paper, long-term TTP is almost not mentioned in the scientific literature. In this paper we discuss the relevance of this study and compare three non-parametric state-of-the-art regression methods: Projection Pursuit Regression (PPR), Support Vector Machine (SVM) and Random Forests (RF). For each one of these methods we study the best combination of input parameters. We also study the impact of different methods for the pre-processing tasks (feature selection, example selection and domain values definition) in the accuracy of those algorithms. We use bus travel time's data from a bus dispatch system. From an off-the-shelf point-of-view, our experiments show that RF is the most promising approach from the three we have tested. However, it is possible to obtain more accurate results using PPR but with extra pre-processing work, namely on example selection and domain values definition.

Keywords: Travel time prediction, long-term, machine learning, regression

1. Introduction

Travel time prediction (TTP) for the long-term could be used in order to better plan transportation services. However, it is not. In logistics, in particular for freight transportation, the use of TTP could be done to better plan the deliveries. Typically, at the best, the existing approaches to address this problem use average times by segmenting time empirically: early morning, morning peak period, etc. This approach can be explained, at least partially, by the lack of data on actual times. Some companies that work in this business do not follow regular routes leading to a reduced number of trips per route. Consequently, in such conditions travel time cannot be accurately predicted due to data sparsity. The

*Corresponding author: João Mendes-Moreira, Departamento de Engenharia Informática, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal. E-mail: jmoreira@fe.up.pt.

only known work on long term TTP was developed for a route planner owned by ANWB, a motorist association from the Netherlands [29].

The reason why long term TTP is not used in public transportation companies is different. In the nineties, some companies began the implementation of dispatch systems that allow the monitoring of the fleet and the archive of respective actual services [46]. Nowadays this kind of systems is used by many public transport providers, specially in developed countries. The use of long term TTP for the planning of public transport companies does not depend on data availability but, instead, on the lack of knowledge on how to use this data in order to reduce costs and/or increase clients satisfaction. In order to better understand this difficulty we briefly describe the typical tasks of operational planning at a public transport company. They are usually done in the following sequential way [8,14,33]:

1. The network definition: it is, obviously, a planning task for the long/very long term. It comprises the definition of the lines, routes and bus stops. We define route as an ordered sequence of directed road stretches and bus stops. Lines are a set of routes, typically two routes that use roughly the same road stretches but in opposite directions.
2. The trips definition: it is a medium term task, with an horizon much shorter than the network definition. There are typically two different methods for trip definition: (1) headway-based, defining the time between two successive trips on the same route [52]; or (2) schedule-based, defining timetables by explicitly setting the departure time and the time of passage at the main bus stops. The supply of trips is defined by route even if they are articulated between groups of routes/ lines [9].
3. The definition of the duties of the drivers and buses: they are medium term (several months) tasks. The goal of both tasks is to define duties. A duty is the work a bus / driver must do. When a duty is defined, in both cases, we do not know which driver or bus will do it. We are just making a logic assignment. The case of bus duties is much simpler than driver duties for obvious reasons: drivers must stop for lunch, cannot drive every day of the week, etc., i.e., they have many more constraints than buses. According to [14], "each driver duty is subject to a set of rules and constraints defined by the government legislation, union agreements and some internal rules of the company". Typically, bus duties are defined before drivers duties.
4. The assignment of duties: it is the task where the driver duties are assigned to drivers and bus duties are assigned to buses. We are now making a physical assignment. Assignment for driver duties is more complex than for bus duties, for similar reasons to the ones explained above. The assignment of driver duties to drivers is called rostering. It can vary significantly from one company to another.

The sequential nature of this planning process and the complexity of these tasks are, probably, the main reasons for the lack of planning procedures using long-term TTP. All this sequential process is strongly time consuming. Additionally, once the duties are planned changes in travel times can have an effect on these planned duties that is hardly predictable. How to use TTP in the planning of public transport companies is, nowadays, the main question. Is it possible to use more flexible planning processes in order to reduce the prediction horizon? How short can this horizon be if the planning is done from the scratch? All these questions have, certainly, different answers for different companies and there are not yet answers to them. The use of long-term TTP can imply the change of some planning procedures. More study must be done in order to better understand the gains and costs from the use of long-term TTP.

In this paper we address the problem of long term TTP. We believe this is an important study which has not been done yet. While in the business of freight transport services still there is a search for data acquisition processes (this is not covered in this paper), in public transport companies the problem is to know how to use these data in order to better plan the operations. In both cases long term TTP is (or will

be) needed. We hope that this paper can give a contribution to increase the knowledge on how to perform TTP for the long term. This paper is an extended version of [40]. This previous paper of 8 pages length was more oriented towards the machine learning process. The present paper describes with much more detail the experiments done and makes the bridge between the business problem and the data mining problem.

In order to predict long term TTP we have chosen three state-of-the-art regression methods: Projection Pursuit Regression, Support Vector Machines and Random Forests. Since each induction algorithm has a set of parameters we have conducted experiments in order to choose the best set of input parameters for each algorithm. However, the performance of these algorithms also depends on the pre-processing tasks, i.e., tasks that are performed before the training of the model. These tasks change somehow the original training set in order to reduce the training time and/or increase the accuracy of the predictions. We are interested in this last goal. The three pre-processing tasks [41] are: (1) the selection of a subset from the original set of features; (2) the selection of a subset of examples from the original training set; (3) the definition of the values to be used at each feature. The first case is called feature selection, the second one is called example (or instance) selection and the last one is the domain values definition. In other words, given a table, the column selection (feature selection), the row selection (example selection) and the domain values definition are the three pre-processing tasks (Reinartz call them focusing tasks). The three experiments we have done on pre-processing tasks are organized as follows:

- First experiment: tests different feature subsets;
- Second experiment: tests different methods for example selection together with the test of different sets of input parameters;
- Third experiment: tests different domain values for the variable week day together with the test of different sets of input parameters.

In the following we firstly describe the data we have used for these experiments. Then, we present the experimental setup followed by the description of the methods used. The three experiments previously described, are presented. Each one is focused on a different pre-processing task as described in [41]. Finally we discuss the lessons learned from these experiments and point out some research directions in order to take advantage of the existence of long term TTP for the planning of transport services.

2. Description of the data

The experiments described in this paper use data provided by the Sociedade de Transportes Colectivos do Porto, SA (STCP), the largest urban public transport operator in greater Oporto, Portugal. STCP has made, in the last few years, important investments in information systems in order to support the decisions of both top and operational managers. One such effort was the SAEI system, a bus dispatching system for real time control purposes.

The main components of the SAEI include:

- Automatic vehicle location based upon dead reckoning sensors supplemented by differential GPS technology;
- Voice and data communication system using TETRA (Trans European Trunked Radio);
- On-board computer and control head displaying schedule adherence information to drivers, and detection and reporting of schedule and route adherence to dispatchers;
- Automatic passenger counters on front and rear doors of twenty vehicles;

Table 1
Variables: data type and cardinality

Variable	Data type	Cardinality†
Departure time	Numeric (in seconds)	6466
Week day	Symbolic	7
Day of the year	Numeric	237‡
Week of the year	Numeric	36
Day type	{holiday, bridge day, tolerance day}	4
School break	{break, normal}	2
Sundays unpd§	Numeric	5
Bus type	Symbolic	5
Driver	Symbolic	206
Service	Symbolic	79
Entrance flow	{we3, we4, holidays, normal}	4
Exit flow	{we3, we4, holidays, normal}	4
Wind speed	Numeric (in m/s)	13
Temperature	Numeric (in $^\circ$ Celsius)	203
Precipitation	Numeric (in mm, $1mm = 1lt/m^2$)	55
Travel time	Numeric (in seconds)	2410

† The cardinality refers to the period from January 1st to August 30th of 2004.
‡ For the said period there are 244 days, but due to some errors in the backup files of the SAEI from where the data was collected, some days are missing.
§ Unpd means until next pay day.

– Computer-aided dispatch center.

A test version of the SAEI system began to store data on a regular basis in 2003. Sometime later, the control center started to be used regularly. Nowadays, at rush hour, it has 8 human officers (dispatchers) to coordinate any problems that might occur. Each dispatcher is responsible for six or seven bus lines. Despite the fact that the SAEI was designed for control purposes, nowadays it is a major source of the data warehouse owned by STCP. The data warehouse is the main source of information for management purposes. The most impressive characteristic of this data warehouse is its very low level of granularity [23]: in fact, information is stored at the bus stop level, allowing a very detailed level of analysis. The data from the SAEI system is used in the experiments described in this paper. Since the goal is to predict travel times of trips, for this study just the information for beginning and end times of the trips (bus voyage from end-point to end-point) was used. Information about passing times at bus stops exists in the SAEI system but it has not been used. Another possibility, instead of predicting trip time, would have to predict travel time for each road stretch that composes the route (this is known as link-based prediction). However, trip time prediction (also known as route-based prediction) is reported to have less variance [10]. The loss of information (predictions per stretch) is not relevant for this particular problem because the duties definition processes at STCP do not use stretch predictions. They use instead route predictions. However, whatever the prediction problem is (route-based or link-based), the lessons learned from the experiments we describe would be the same.

The variables used in the experiments were obtained by visual inspection [39] and using the advise of traffic experts. In Table 1 the input variables for each one of the seasonalities/impact factors identified by visual inspection are presented. For the seasonality of the year, we use two variables, because, initially, it was not clear which of them would be more valuable. Despite the existence of all these 15 input variables the majority of the experiments use only 4 of them. They are: departure time, week day, day of the year and day type. The reason to do this is explained in Secttion 5.1. That section, on feature selection, is the

only one where all variables are used. The target variable (usually referred as the y variable) is the value we want to predict that is the travel time.

We have used data from January 1st to August 30th to analyse the variables (Table 1) and for the experiments on feature selection (Section 5.1). All the remaining experiments described in this paper use data from January 1st to March 31st of 2004. All data is from route 78-1-1. This route was cancelled in the end of 2006.

3. Experimental setup

The problem we address in this paper is a typical inductive learning regression problem. Inductive learning for regression consists ideally in the inference of a function

$$\hat{f} : X \rightarrow \Re, \text{such that } \hat{f}(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in X, \tag{1}$$

where f represents the unknown true function.

The algorithm used to obtain the $\hat{f}$ function is called induction algorithm or learner. The particular values for the input parameters of the induction algorithm are often referred to as the parameter set. The result of the induction algorithm is the $\hat{f}$ function, called model, predictor or regressor. The data set used by the algorithm to induce $\hat{f}$ is the training set. However, in practice, the output of the learned function $\hat{f}$ will not coincide with the output of the real function f for every input value $\mathbf{x}$. The typical result of the learner is an approximate function that minimizes the difference between $\hat{f}$ and f. In order to evaluate this difference it is used a data set (known as test set). Its instances have not been used for training assuring that the distance measured between $\hat{f}$ and f is a good approximation for the difference between the predictions of previously unseen instances and their actual travel times. We evaluate this difference using the *variation index* measure Eq. (3). The reason for using this measure is three-fold, the quadratic measure of the distance (which penalizes the distance more than a linear measure), its interpretability and the fact that it is a known measure in the context of operations planning at public transport companies [46,50]. The *variation index* gives the ratio between a dispersion measure and the average.

$$mse = \frac{1}{n}\sum_{i=1}^{n}(\hat{f}(\mathbf{x_i}) - f(\mathbf{x_i}))^2, \tag{2}$$

$$varIndex = \frac{\sqrt{mse}}{\bar{y}}, \tag{3}$$

where $\bar{y}$ is the average of $f(\mathbf{x_i}), i = 1, \ldots, n$. For the particular case of travel time prediction, $f(\mathbf{x_i})$ represents the actual travel time of trip i and $\bar{y}$ represents the average travel time for the n trips.

We have used a three-day prediction horizon. The reason is that, from the point-of-view of transport planners, a shorter horizon would be not very realistic. Three days is what we believe to be the shortest realistic planning horizon.

The experimental setup we use is a sliding window with a 30-day time stamp (Fig. 1). This sliding window is used because it is expected that the last days are the ones that can give more information on what will happen three days ahead. The use of a time stamp-based window against a sequence-based one, as well as the use of 30 days of data for training, has no scientific support. For this particular data set, the average number of trips for each 30-day window is around 9 hundred. Nevertheless, the most

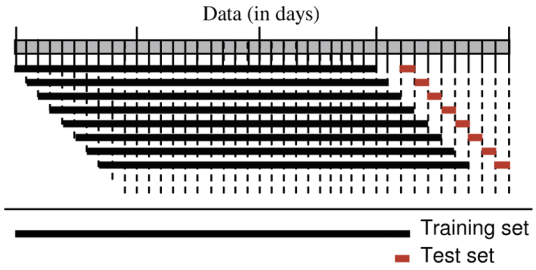

Fig. 1. Experimental setup using a time stamp sliding window.

adequate window size can and should be studied also because fixing the number of days is oriented towards constraining the computational complexity instead of improving the accuracy. This problem is discussed in [53]. However, it is not the focus of this research.

All the experiments use the R statistical package [48].

4. Choosing the regression methods

Before choosing which induction algorithm to use, we must define which characteristics they should have. Often, such methods are classified according to their interpretability and predictive power, although other characteristics may be considered [20]. In any case, for TTP, the predictive capability is the one that shall be optimized. In fact, from the point of view of the goal of this research, the usefulness of interpretability is to get insights into the contribution of each input variable to the prediction performance. The interpretability from the transport planners point of view is not important because the variables that could be useful for them are not the ones that are important for the improvement of the prediction's accuracy. An example is the existence of bus lanes. The impact of bus lanes is relevant for business negotiations between the STCP company and the Oporto authorities, namely the town hall, but this variable is not interesting from the point of view of the prediction's accuracy because it is implicit in the choice of the route. Several other variables exist in the same situation.

In [20], artificial neural networks (ANN) [2], Support Vector Machines (SVM) [45] and instance local regression / instance based methods [1,11] are the ones reported to be the most accurate when compared against decision trees [7] and Multivariate Adaptive Regression Splines (MARS) [16]. This evaluation was not based in tests using a known set of problems. It is only a general classification based on the authors' experience.

In any case, the main reference for the choice of the induction algorithms to use for TTP was the work by Meyer et al. done on nine real and three artificial benchmark data sets from different domains. They test SVM, linear regression [28], decision trees, ANN, MARS, BRUTO (similar to MARS but cannot handle categorical input variables), Multiple Additive Regression Trees (MART) [17], Projection Pursuit Regression (PPR) [18] and two decision tree-based ensemble algorithms: Random Forests (RF) [5] and bagging [4]. Globally ANN, SVM, PPR and RF are the best [35].

From the four induction algorithms previously referred, we have selected three of them: SVM, PPR and RF. There is no special reason to select these three since there is no evidence of the advantage of any of the four methods mentioned over the other three from the benchmark. The aim was to reduce the time of the experimental process.

In the remainder of this section we briefly describe each one of the three selected induction algorithms. Besides these algorithms we have also used a baseline method for comparison that is described next.

4.1. A baseline method

In the spirit of a common recommendation on prediction methods [34], we start the experiments using a simpler approach to be used as a baseline method. The goal is, obviously, to obtain the first results for comparison with more sophisticated approaches. The baseline method we use does not perform any learning.

The main idea of this method is to select the past example most similar to the one we want to predict. However, considering the different behaviour of travel time in different days (for example, weekdays, Sundays, holidays, etc.), we search the most similar example only from days equivalent to the example we want to predict. Consequently, the baseline method has two steps: (1) the first step filters data regarded as equivalent to the day we want to predict; and (2) the second one uses the filtered data set from which to select the nearest past example.

The first component filters the data according to the equivalent day's group on which the data to predict falls. If the equivalent day's group is empty then all data is used at the second step. These groups are:

- Day Type = Normal and Week Day = working days (from Monday to Friday);
- Day Type = Normal and Week Day = Saturdays;
- Week Day = Sundays;
- Day Type = bank holiday and Week Day on Monday, Friday;
- Day Type = bank holiday and Week Day on Tuesday, Thursday;
- Day Type = bank holiday and Week Day = Wednesday;
- Day Type = bank holiday and Week Day = Saturdays;
- Day Type = bridge day;
- Day Type = tolerance day.

The equivalent day's groups were defined by visual inspection and using experts knowledge.

The second component uses a distance measure to obtain the nearest neighbor. The selected distance measure is the Heterogeneous Euclidean-Overlap Metric (HEOM) because it is one of the mentioned measures that handle numeric and nominal attributes simultaneously [54]:

$$\mathit{heom}(\mathbf{x}, \mathbf{t}) = \sqrt{\sum_{a=1}^{v} d_a(x_a, t_a)^2}, \tag{4}$$

where

$$d_a(x,t) = \begin{cases} 1, & \text{if } x \text{ or } t \text{ is unknown, else} \\ \mathit{overlap}(x,t), & \text{if } a \text{ is nominal, else,} \\ \mathit{rn\_diff}_a(x,t) & \end{cases}, \tag{5}$$

$$\mathit{overlap}(x_1, x_2) = \begin{cases} 0, \text{if } x_1 = x_2 \\ 1, \text{otherwise} \end{cases}, \tag{6}$$

and

$$\mathit{rn\_diff}_a(x_1, x_2) = \frac{|x_1 - x_2|}{max_a - min_a}. \tag{7}$$

v is the number of input variables (also named attributes), min_a and max_a are, respectively, the minimum and the maximum values of attribute a.

The baseline method calculates the HEOM distance between the test instance (the $\mathbf{t}$ variable in Eq. (4)) and each element $\mathbf{x}$ of the training set that belongs to the same equivalent day. The target value of the instance with the shortest HEOM distance is the prediction of the baseline method.

Using the experimental setup described in Section 3, the *variation index* obtained with the baseline method was 12.32%. This value will be the reference for comparison along this paper whenever appropriate.

4.2. PPR – Projection Pursuit Regression

Projection Pursuit Regression (PPR) is an additive model [20], i.e., it can be decomposed as a sum of univariate functions called ridge functions. However, instead of the original features, it uses a linear combination of these. The V dimensional original space is linearly projected and the image of this projection in the univariate f_i ridge functions is added, this is,

$$\hat{f}(x_1, x_2, \cdots, x_V) = \sum_{i=1}^{t} f_i \left(\sum_{j=1}^{V} (\beta_{i,j} \times x_j) \right), \tag{8}$$

where t is the number of iterations. Its value is obtained on the fly.

PPR was presented in 1981 [18] and this can perhaps explain the relative lack of use of this promising method. One of the main drawbacks of PPR is its computational cost. For modern-day computers, the effort is reasonable, in contrast to the past. Another disadvantage of PPR is its difficult interpretation. However, the prediction accuracy and the ability to bypass the curse of dimensionality are two of the most important characteristics of PPR [22]. An important result of PPR is that it is an universal approximator, i.e., for t arbitrarily large and for an appropriate choice of the f_i ridge functions, PPR can approximate any continuous function in $\Re^V$ [20].

The algorithm first adds up to *max.terms* ridge terms f_i. It then removes the least 'important' term at each step until *nterms* terms are left. *max.terms* and *nterms* are two parameters that must tuned. There is a third parameter: *optlevel*. It controls which elements of the PPR formulation are refitted at each step. Further details on the parameters of PPR can be found in [48].

In all the experiments done with PPR we use *max.terms* as the number of input variables (in this case, *max.terms* $= 4$). For experiments on parameter set selection, *nterms* $\in \{1, 2, 3, 4\}$ and *optlevel* $\in \{0, 1, 2, 3\}$.

Furthermore, PPR has a method for smoothing the ridge functions, called smoothers. We assume that different smoothers induce different algorithms. The *ppr* function from the R-project [48] has three different options for smoothers [48]:

- The super smoother (*supsmu*);
- The splines (*spline*);
- The Generalized Cross-Validation (GCV) spline (*gcvspline*).

Depending on the smoother there are different parameters that must be tunned. The set of values tested for those input parameters that depend on the smoother is presented in Table 2.

4.3. SVM – Support vector machines

Support vector machines (SVM) is a relatively recent family of successful algorithms for both regression and classification. Based on the statistical learning theory (VC theory) developed during the

Table 2
Input parameters for PPR

algorithm	bass	span	df	gcvpen
PPR – supsmu	idx_1	0		
	0	$idx_2/10$		
PPR – spline			2^{idx_1}	
PPR – gcvspline				2^{2*idx_3}

$idx_1 = 0, 1, \cdots, 10; idx_2 = 1, 2, \ldots, 10; idx_3 = -2, -1, \cdots, 6.$

sixties/seventies, it was mainly in the nineties, when Vladimir Vapnik began to work at AT&T Bell Laboratories, that SVM were largely developed. Nowadays SVM is an area of research in its own right with a large number of successful applications. This section follows closely [45].

The basic idea of $\varepsilon-$SVM, the most used SVM algorithm, is to learn $\hat{f}$ such that the error for the training data is the minimum possible higher than a predefined ε and, simultaneously, keeping $\hat{f}$ as flat as possible. This is achieved using the ε-insensitive loss function:

$$|y - \hat{f}(\mathbf{x})|_\varepsilon = \begin{cases} 0, & \text{if } |y - \hat{f}(\mathbf{x})| \leqslant \varepsilon \\ |y - \hat{f}(\mathbf{x})| - \varepsilon, & \text{otherwise} \end{cases}. \tag{9}$$

The examples that minimize the second part of the loss function, thus implicitly defining the margin, are called support vectors.

For the case of linear functions, the problem can be written as a linear programming one:

$$\begin{aligned} &\text{minimize } \frac{1}{2} \parallel \mathbf{w} \parallel^2 + C \sum_{i=1}^{M} (\xi_i + \xi_i^*) \\ &\text{subject to: } \begin{cases} ((\mathbf{w} \cdot \mathbf{x}_i) + \beta_0) - y_i \leqslant \varepsilon + \xi_i \\ y_i - ((\mathbf{w} \cdot \mathbf{x}_i) + \beta_0) \leqslant \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geqslant 0 \end{cases}, \end{aligned} \tag{10}$$

where $\mathbf{w} = (\beta_1, \beta_2, \cdots, \beta_V)$ ($\beta_i, i = 1, 2, \cdots, V$ are the coefficients of the linear model), ξ_i and ξ_i^* are slack variables, and ε and C are input parameters. ε is the limit for non-penalized errors and C determines the trade-off between the flatness of $\hat{f}$ and the tolerance to errors larger than ε. The flatness of $\hat{f}$ is guaranteed by the first term of the objective function.

After some reformulation, using the dual problem and generalization for the nonlinear case, the problem takes the form:

$$\begin{aligned} &\text{minimize } \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \text{Kn}(\mathbf{x}_i, \mathbf{x}_j) \\ &\qquad + \varepsilon \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) - \sum_{i=1}^{M} y_i (\alpha_i - \alpha_i^*) \\ &\qquad \text{subject to: } \begin{cases} \sum_{i=1}^{M} (\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}, \end{aligned} \tag{11}$$

Table 3
Input parameters for SVM

algorithm	cost (C)	nu (ν)	gamma (γ)
SVM – linear	2^{2idx_1}	$\frac{idx_2}{10}$	
SVM – radial	$2^{2idx_3} \times 1000$	$\frac{idx_4}{5}$	$6idx_5/100000$

$idx_1 = -2, -1, \ldots, 6$; $idx_2 = 1, 2, \ldots, 10$; $idx_3 = 1, 2, \ldots, 5$; $idx_4 = 1, 2, \ldots, 4$; $idx_5 = 1, 2, 3$.

where Kn is a kernel function. This function is of major importance in the SVM algorithm since it allows the generalization for the nonlinear space. Research exists to define appropriate kernels for specific types of problems.

A more complete overview on support vector regression can be obtained in [45,51].

Schölkopf et al. [44] propose an alternative formulation to the one given by Eq. (11). The motivation for this new formulation, called ν-SVM, is that ε-SVM needs to give the desired accuracy of the approximation as an input parameter (the ε parameter) while in most cases the goal is to obtain a $\hat{f}$ function as accurate as possible. ν-SVM automatically minimizes ε. However, it is not expected that one of the formulations will give better results than the other. The ν parameter is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors. An important difference for the selection of the parameter set is that $\nu \in [0, 1]$ while $\varepsilon \in [0, +\infty[$. Consequently, it is easier to tune ν than ϵ. The ν-SVM formulation for regression is

$$\text{minimize } \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\text{Kn}(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{M} y_i(\alpha_i - \alpha_i^*) \tag{12}$$

$$\text{subject to: } \begin{cases} \sum_{i=1}^{M}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in \left[0, \frac{C}{M}\right] \\ \sum_{i=1}^{M}(\alpha_i + \alpha_i^*) \leqslant C \cdot \nu \end{cases}.$$

In this paper we have adopted this later approach and the kernels used were:

– linear: $\text{Kr}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j$;
– Gaussian radial basis function: $\text{Kr}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2)$.

SVM are sensitive to the scale used by the numeric variables. In order to avoid this problem, a common approach is to standardize those variables to have zero mean and standard deviation one [20]. This is done by default in the SVM implementation used in the R statistical package.

While ν and C are needed whatever the kernel is, γ is kernel dependent (radial kernel).

The experiments on SVM use the parameters presented in Table 3. The used values were obtained firstly from [35] and secondly by preliminary tests in order to get some insights into the range of values to use.

4.4. RF – Random Forests

A Random Forest (RF) is an ensemble of models, i.e., it uses more than one model, instead of just one, for the prediction task. The main idea of ensemble methods is to generate different models in such a way that the combination of their results reduces the generalization error, when compared to the use of just one model.

Breiman's Random Forests method [5] uses an algorithm for induction of decision trees [7] which is modified to incorporate some randomness: the split used at each node is defined a randomly selected feature subset. The subset considered in one node is independent of the subset considered in the previous one. This strategy based on the manipulation of the learning algorithm is combined with subsampling, since the ensemble is generated using the bagging approach [4]. The strength of the method is the combined use of bootstrap sampling and random feature selection.

RF constructs a pre-defined number of trees (it is one of the input parameters of the method, called *ntree*) using a variant of the CART algorithm and averages the results obtained by the set of trees. The CART algorithm [7] is manipulated by selecting randomly, at each split, the set of variables from where the split variable is selected [5]. The number of such variables is another input parameter, named *mtry*. Several other input parameters exist but are not relevant for the prediction accuracy [6,31]. They are important for the interpretability and consequently they are not discussed here. A property of RF is that its result converges as *ntree* increases. In all the experiments with RF performed here, *ntree* is set to 1000 as suggested in [6]. The values for *mtry* are, obviously, *mtry* $\in \{1, 2, \cdots, V\}$. In the present case, *mtry* $\in \{1, 2, 3, 4\}$.

5. Experiments

Data can be manipulated in order to increase accuracy and reduce the computational cost, among other benefits. There are three pre-processing tasks: feature selection, example selection and domain values definition. All of them can be used to increase accuracy, while example and feature selection can also be used to address the computational cost issue. In fact, the computational complexity for each of the methods is a function of M, the number of examples in the training set, and V, the number of features/variables:

- SVM: $O(N_{SV}^3 + N_{SV}^2 \times M + N_{SV} \times V \times M)$ [47], where N_{SV} is the number of support vectors;
- RF: $O(\sqrt{V} \times M \times \log M)$ [43];
- PPR: $O(t \times V \times M \times \log(M))$ [18] where t is the number of iterations (Section 4.2).

This section follows this order: (1) feature selection; (2) example selection; and (3) domain values definition. In the last two experiments we also test different parameter sets.

5.1. Feature selection

The experiments on feature selection, and only in these ones, use data from January 1st to August 30th of 2004. It would be better to use, at least, one year of data. However, such amount of data was not available when these experiments were done.

Some authors classify the features as relevant or irrelevant according to how the inclusion of these features in the training set improves the results of the prediction task (considering just supervised learning) [26,36]. Even knowing that the feature subset that optimizes accuracy depends on the induction algorithm to be used [30], we will start by removing overall irrelevant features. Using the definition given by Molina et al. for irrelevant features, they are the ones "not having any influence on the output, and whose values are generated at random for each example" [36]. Molina et al. compare some of the most well known algorithms for feature subset selection in order to evaluate some particularities, one of which is irrelevance. The method with best results for detecting irrelevance is Relief. According to Dietterich, "[ReliefF (a posterior version of Relief algorithm)] is one of the most successful preprocessing

algorithms to date" [15]. One decade has passed but the Relief family of algorithms is still frequently used for data mining applications [12,21,24,25,32].

The Relief family of algorithms [27] weighs the features according to "how well their values distinguish between instances that are near to each other" [42]. Originally the Relief algorithm only addressed the classification problem. The regression version, called RReliefF, was introduced later.

Our approach for the detection of irrelevant features uses the RReliefF algorithm [42]. This algorithm computes the weights of the input variables, giving a measure of how much each variable "explains" the target variable. Its pseudo code is:

```
Program RReliefF(R, τ, t, k)
{where R = {R_i, i = 1, 2, ···, M} and each R_i is an input vector;
τ is a vector with the target values corresponding to R;
 t is the number of iterations; and
 k is the number of nearest examples}
begin
  set N_dC, N_dA[A], N_dC&dA[A], W[A] to 0, where A = 1, 2, ···, V
  for (i:=1 to t)
    randomly select example R_i from R
    select k examples I nearest to R_i
    for (j:=1 to k)
      N_dC := N_dC + diff(τ(.), R_i, I_j) × d(i, j)
      for(A:=1 to V)
        N_dA[A] := N_dA[A] + diff(A, R_i, I_j) × d(i, j)
        N_dC&dA[A] := N_dC&dA[A] + diff(τ(.), R_i, I_j)×
                                  diff(A, R_i, I_j) × d(i, j)
      end for
    end for
  end for
  for (A=1 to V)
    W[A] := N_dC&dA[A]/N_dC − (N_dA[A] − N_dC&dA[A]) × (t − N_dC)
  end for
  return(W)
end.
```

The variable V in the algorithm is the number of input variables of the data set, $\tau(.)$ is the target variable, i.e., the travel time, and M is the number of examples in the training set. The parameter t is the number of iterations. Its choice is problem dependent. The larger the t value is, the more stable the weights estimations are. Additionally, the computational cost also increases. Robnik-Sikonja & Kononenko state that the stability of the results is obtained, typically, for values between 20 and 50 iterations. We use $t = 50$. The value used for k (the number of nearest examples) was 10, as suggested by the authors. From the several distance functions proposed by the authors, we use one that quadratically increases the cost of the distance:

$$d(i,j) = \frac{d1(i,j)}{\sum_{l=1}^{k}(d1(i,l))}, \tag{13}$$

$$d1(i,j) = \frac{1}{(\sum_{l=1}^{V} \mathit{diff}(A_l, R_i, I_j))^2}, \text{ and} \tag{14}$$

$$\mathit{diff}(A, I_1, I_2) = \begin{cases} 0 & : \quad d \leqslant t_{eq} \\ 1 & : \quad d > t_{diff} \\ \frac{d - t_{eq}}{t_{diff} - t_{eq}} & : \quad t_{eq} \leqslant t_{diff} \end{cases}, \tag{15}$$

where the values of t_{eq} and t_{diff} are, respectively, 5% and 10% of the length of the input variable's value interval, as suggested by the authors; and the value d represents the absolute difference of the input variable A for the two examples, I_1 and I_2.

The first experiment on feature selection uses RReliefF in order to select, from the initial set of features (Table 1), a subset of them. The selected subset will be used in a second experiment on feature selection. The other features will be discarded. A feature is selected if its RReliefF weight is larger than 0.01, as suggested in [19]. However, remembering the experimental setup (Section 3), for 60 days of data, for instance, there are $60 - 30 + 1$ different training sets, where 30 is the number of days in the time stamp. Results (Table 4) present five statistics: the percentage of days where the RReliefF weight is larger than 0.01 and the minimum, maximum, mean and standard deviation of the RReliefF weight values.

Some comments can be drawn:

- The low values for day type, entrance flow and exit flow can be explained by the use of 30 days in the training set and the previous knowledge that these values are rarely different to the standard ones.
- The low value for Sundays until next pay day can be due to the use of just one seasonal-cycle of this event. This variable could be useful if using a larger training set.
- Of the two variables used to capture the seasonality of the year: day of the year and week of the year, the first one is the most relevant. Due to the dependence between these two variables, the respective weights are expected to be lower than they should be [42].
- A natural doubt is if a feature with a low percentage of days with a weight larger than 0.01 has, in those days, a weight much larger than 0.01. 'Day type' is one of those cases (even if not too evident), i.e., when a test day is of a day type not 'normal', the most common is that in the training set there are no examples from the same day type. However, if there are, the feature day type can be relevant, specially when the test day is a working day.

For the above reasons, the variables week of the year, school break, Sundays until next pay day, entrance flow and exit flow are classified as irrelevant.

The following step was to select the feature subset that maximizes accuracy. The used approach takes advantage of the embedded feature selection used in random forests. Taking advantage of RF capacity to ignore irrelevant features, the approach we use runs RF using pre-specified subsets of input variables. Using this approach it is expected that the best result is obtained using all the variables. It is also expected that removing irrelevant variables, performance keeps stable. The tested subsets are:

- RedSet: the reduced set, i.e., the set of four variables considered the most important by the experts;
- RedSet+Meteo: the reduced set with the meteorologic variables;
- AllRRF-Meteo: all variables excluding the irrelevant and the meteorological ones;
- AllRRF: all variables excluding the irrelevant ones;
- All15: all the 15 variables.

For each of these subsets, all possible values of the $mtry$ parameter are tested.

The best subset is, as expected, All15. However, AllRRF-Meteo uses just 7 variables and obtains a very similar result. Since the meteorological variables are the only ones that are obtained from outside

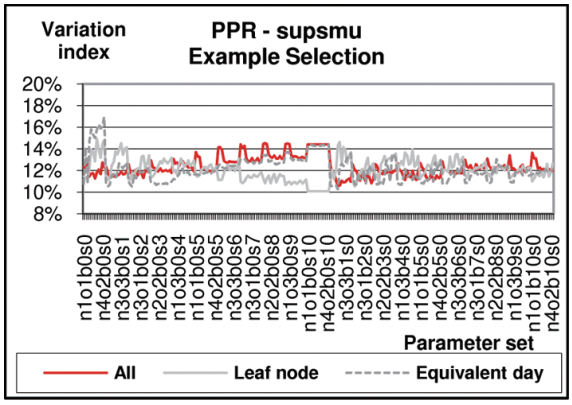

Fig. 2. The *variation index* for PPR – supsmu using different methods for example selection.

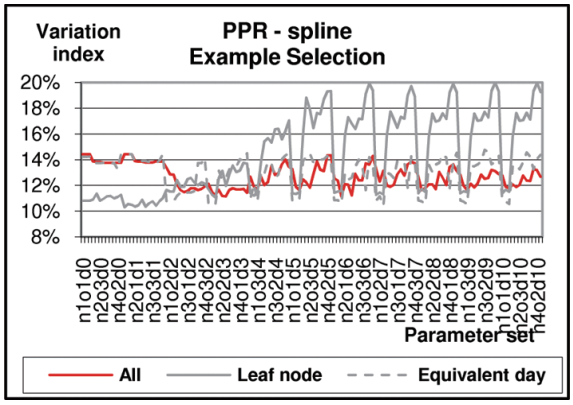

Fig. 3. The *variation index* for PPR – spline using different methods for example selection.

the company, it is relevant to get insights into how valuable these variables are for prediction. Comparing results for AllRRF and AllRRF-Meteo, the difference among the best results for each of these two subsets is around 0.3%. It is important to note that the meteorological variables used as input variables have the actual values, instead of the predictions. This happens for all the variables used in the experiments. However, if inspecting the promising subset RedSet+Meteo, three variables (weekday, day type and day of the year) are certain, departure time is uncertain but it is the one that we hope to keep unchanged when predicting travel time three days ahead. The meteorological variables are the ones whose predictions three days ahead are expected to be more erratic. If we use predicted values for the input variables instead of the actual ones, the difference among the results for AllRRF and AllRRF-Meteo is expected to be higher than 0.3%.

The main result of the experiments on feature selection is that the RedSet, with or without meteorological variables depending on their availability, is a good feature subset to be used in further experiments.

5.2. *Example selection*

The main idea of example selection is to increase accuracy by selecting from the training set just a subset of the examples for the training task.

Two approaches were tested:

- Equivalent days (ed): uses the examples from identical past days according to several groups, defined by visual inspection and using domain knowledge. These groups are the ones defined in Section 4.1. If there is not a minimum of examples from equivalent days, according to the input parameter *min members*, all the examples from the training set are used. The value of *min members* was always set to 10 in all the experiments using the equivalent days approach.
- Leaf node (ln): uses the examples that fall in the same leaf node of a CART tree [7] as the current test example. The CART model is induced on the full training set.

The idea behind using example selection can be stated in this way: as happens with autoregressive methods [34] the use data from equivalent past periods to predict the unknown future (using typically a weighted average of the past equivalent data), it is expected that by using only similar past data for training, the methods can predict better because there will be less 'noise'. Both example selection use this principle: to select only the most promising past data. In the first case, the approach could be easily

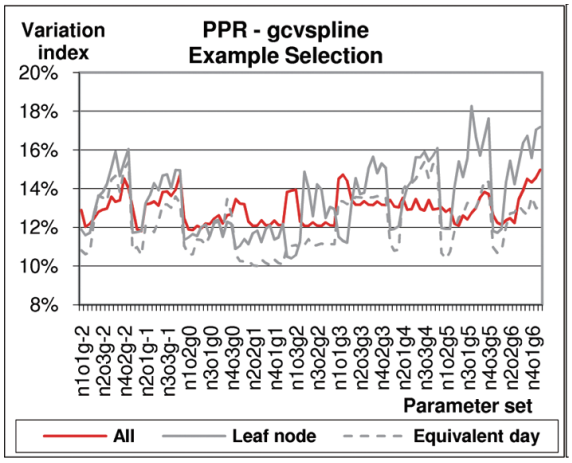

Fig. 4. The *variation index* for PPR – gcvsline using different methods for example selection.

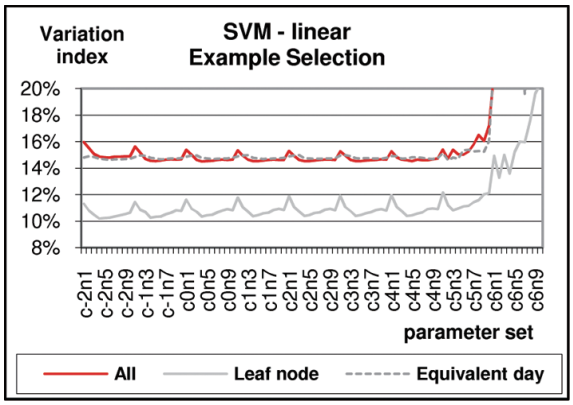

Fig. 5. The *variation index* for SVM – linear using different methods for example selection.

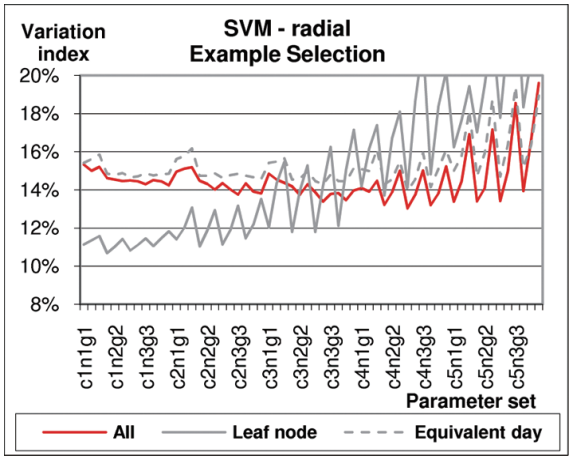

Fig. 6. The *variation index* for SVM – radial using different methods for example selection.

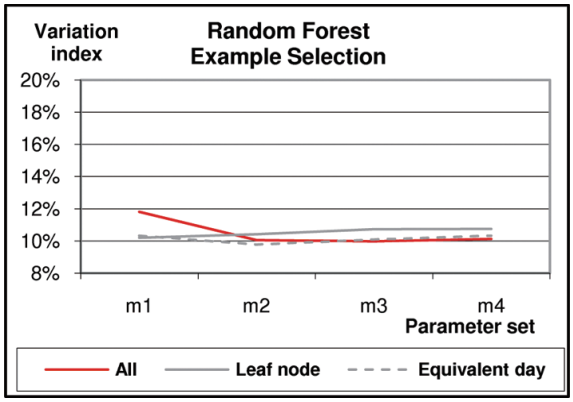

Fig. 7. The *variation index* for RF using different methods for example selection.

improved by using just past trips with similar departure times. The advantage of the leaf node approach is that it is completely independent of the existing expertise on the problem and, consequently, it can easily be adapted to other problems, as discussed later in this section.

5.2.1. Results on example selection

Results for each induction algorithm are presented in Figs from 2 to 7. In all these figures, the series 'All' presents the results using all the examples from the training set, i.e., without example selection. The values of the abscissa are represented as follows:

– For PPR, each parameter set is represented and ordered by n[*nterms* index]o[*optlevel* index]b[*bass* index]s[*span* index] for the super smoother, by n[*nterms* index]o[*optlevel* index]d[*df* index] for spline and by n[*nterms* index]o[*optlevel* index]g[*gcvpen* index] for gcvspline. The meaning of the indexes are explained in Table 2.

Table 4
Statistics on RReliefF weights using 207 training windows

variable	% of days where RW † > 0.01	min	max	mean	sd
Departure time	96.62%	0.08%	30.36%	11.56%	5.90%
Week day	58.45%	0.01%	5.78%	1.57%	1.26%
Day of the year	14.98%	0.00%	2.87%	0.54%	0.48%
Week of the year	0.97%	0.00%	1.44%	0.11%	0.21%
Day type	1.45%	0.00%	2.48%	0.09%	0.25%
School break	0.00%	0.00%	0.35%	0.01%	0.04%
Sundays unpd	0.97%	0.00%	1.30%	0.15%	0.23%
Bus type	64.25%	0.01%	6.05%	1.80%	1.42%
Driver	88.89%	0.01%	17.74%	5.14%	3.84%
Service	89.86%	0.04%	15.60%	5.04%	3.59%
Entrance flow	0.00%	0.00%	0.78%	0.03%	0.12%
Exit flow	0.00%	0.00%	0.82%	0.02%	0.09%
Wind speed	66.18%	0.01%	10.66%	2.45%	2.18%
Temperature	63.29%	0.01%	7.64%	1.80%	1.57%
Precipitation	16.91%	0.00%	5.92%	0.60%	0.98%

† RW means RReliefF weight.

Table 5
The *variation index* for RF using different subsets (the best value per subset is in bold face)

mtry	All15	AllRRF	AllRRF-Meteo	RedSet+Meteo	RedSet
1	14.03%	12.93%	12.31%	12.68%	11.85%
2	12.34%	10.96%	10.21%	10.67%	10.06%
3	11.24%	10.11%	**9.93%**	9.93%	**9.96%**
4	10.51%	9.74%	9.99%	9.66%	10.11%
5	10.11%	9.64%	10.09%	**9.62%**	
6	9.84%	**9.64%**	10.11%	9.70%	
7	9.69%	9.73%	10.11%	9.78%	
8	9.63%	9.79%			
9	**9.59%**	9.78%			
10	9.63%	9.79%			
11	9.64%				
12	9.65%				
13	9.66%				
14	9.65%				
15	9.64%				

- For SVM, each parameter set is represented and ordered by c[C index]n[μ index] for the linear kernel and by c[C index]n[μ index]g[γ index] for the radialkernel. The meaning of the indexes are explained in Table 3.
- For RF, each parameter set is represented and ordered by m[*mtry* value].

In Table 6 the best results for each algorithm and for each different method to select the training set are presented. It is interesting to observe that after the introduction of a method to select examples, the best results for each different algorithm are much closer to each other.

Next, we discuss the main results for each of the three methods. We divide this discussion in general comments (that includes discussion on parameter set selection) and on comments about example selection.

PPR – general comments:

Table 6
The best *variation index* for each example selection method × algorithm

algorithm	all	ln	ed
Baseline	12.32%		
SVM – linear	14.50%	**10.18%**	14.59%
SVM – radial	13.03%	**10.68%**	13.95%
RF	9.92%	10.29%	**9.80%**
PPR – supsmu	10.60%	**10.09%**	10.27%
PPR – spline	11.15%	**10.30%**	10.40%
PPR – gcvspline	11.80%	10.39%	**10.00%**

– Results for PPR – supsmu have two distinct parts. The first one is easily identifiable because the determination of the parameter set finishes with $s0$ (i.e., *span*$= 0$). It corresponds to set span by local cross validation. When the identification of the parameter set finishes with an index for s larger than 0, the span is directly set. Results using local cross validation to set span are better.
– The super smoother is the one that gives the best result among the three available smoothers in R.

PPR – example selection: For PPR, both approaches perform better than the all approach, even if for some parameter sets they perform worse. Anyway, it is clear that both approaches for example selection can ameliorate the accuracy.

SVM – general comments: Since the goal is to find the best parameter set for each algorithm, there is no interest in a deep analysis of the sensitivity of each algorithm to the parameters. Additionally, the range of values tested in those experiments was selected from previous experiments using a larger and wider grid (in particular for unbounded parameters). Consequently, the results are already conditioned by these previous selections. Since C determines the trade-off between the flatness of $\hat{f}$ and the tolerance to errors, it is expected that the larger C is, the larger the number of support vectors will be. Also, the larger ν is, the larger the number of support vectors is. As the SVM computational complexity is given by $O(N_{SV}^3 + N_{SV}^2 \times M + N_{SV} \times V \times M)$ [47], (N_{SV} is the number of support vectors and M and V are, respectively, the training set size and the number of input variables, as usual) the larger C and ν are the larger the computational time for training is.

SVM – example selection: For SVM, the leaf node approach is the best one. This result is particularly impressive. As previously mentioned, the leaf node approach is not problem dependent and, consequently, there is an open question: is the leaf node approach promising in other domains? The answer to this question is given in [38]. The leaf node approach is tested in eleven regression data sets [49] just for SVM – linear. The regression data sets are not time varying and, consequently, the experimental setup is necessarily different. We use 10-fold cross validation. The leaf node approach increases accuracy in seven data sets, draws in two and loses in the remaining two data sets. These results were obtained with 5% statistical significance.

RF – general comments:

– Random Forests is a very appealing 'off-the-shelf' method because parameter selection is easy (there is only one relevant parameter that is two side bounded) and performs well. Additionally it is interpretable (in this case some more parameters are needed).
– RF is not very sensitive to the value of *mtry*. This is apparent from Fig. 7 and is confirmed in [5].
– Despite the fact that RF is not a deterministic model, due to its random nature, just one result of each experience is shown and the variance of the result is ignored. This happens because the *variation index* for different runs of the same experience, using 1000 trees, never differed more than four units in the fourth decimal number.

Table 7
The *variation index* using example selection (ES) and different data types for the variable week day (WD)

	WD (All)		ES (WD = sym)		ES & WD	
	Best option	Value	Best option	Value	Best options	Value
Baseline		12.32%				
SVM – linear	sym	14.50%	ln	**10.18%**		
SVM – radial	sym	13.03%	ln	**10.68%**		
RF	num	9.88%	ed	**9.80%**		
PPR – supsmu	num	10.15%	ln	10.09%	ed-num	**9.73%**
PPR – spline	num	10.24%	ln	10.30%	ed-num	**9.79%**
PPR – gcvspline	num	10.21%	ed	10.00%	ed-num	**9.55%**

ES = example selection; WD = week day; sym = symbolic; num = numeric; ln = leaf node; ed = equivalent day.

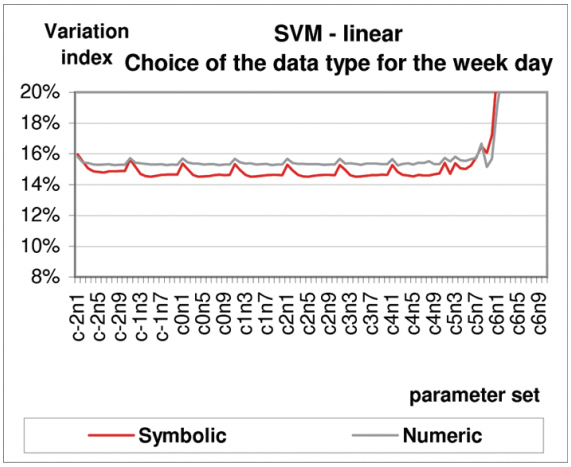

Fig. 8. The *variation index* for SVM – linear using different data types for the variable week day.

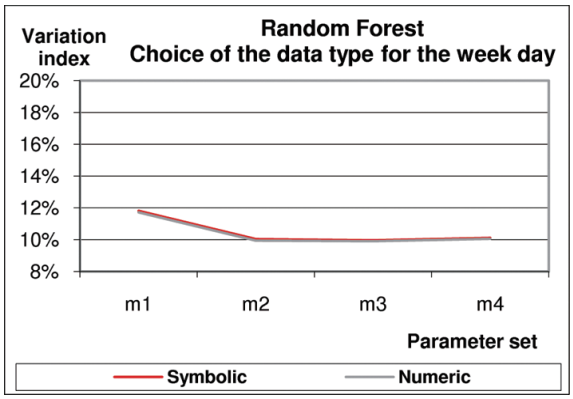

Fig. 9. The *variation index* for RF using different data types for the variable week day.

RF – example selection: Results for RF are not surprising. In fact, the leaf node approach uses a filter identical to the one embedded in CART, and the equivalent day approach applies a kind of filter that can be, potentially, applied in CART when it is statistically relevant. In fact, the sequence of splits, implemented by CART, splits the data according to a set of rules in order to minimize, with some constraints (such as the minimum number of examples per leaf node), the sum of the variances of the resulting subsets. The equivalent day approach uses a pre-specified set of such rules as a previous step to the use of the CART approach.

5.3. Domain values definition

The choice of domain values can also affect the predictive ability of a method. Therefore, we have also observed how the variation index for the different methods is affected by different choices on the domain values.

Domain values definition can include: (1) the choice of the data type for each variable; (2) the discretization of continuous variables; or (3) the choice of appropriate values for a symbolic variable. Any of these tasks could be tested for this particular data set. Analyzing Table 1, we have identified the following possible experiments:

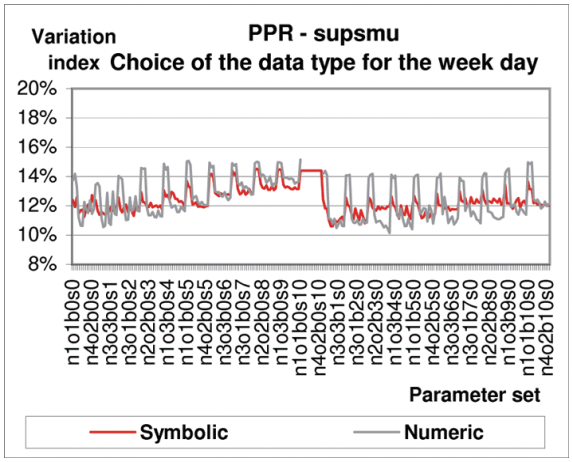

Fig. 10. The *variation index* for PPR – supsmu using different data types for the variable week day.

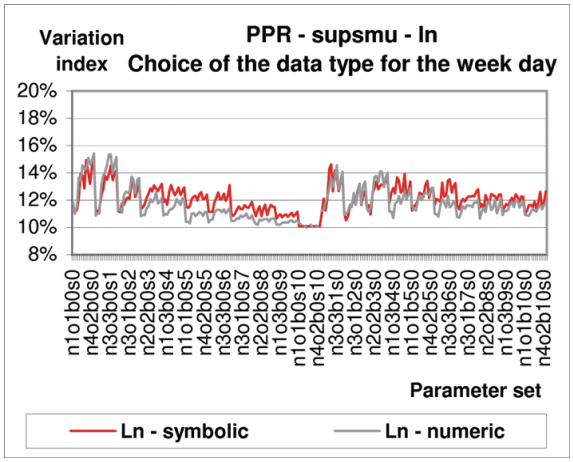

Fig. 11. The *variation index* for PPR – supsmu using different data types for the variable week day and the leaf node approach for example selection.

Table 8
Ranks of the Friedman test

	1	2	3	4	5	6	7	8	9	10	11	12	mean
Baseline	7	7	7	7	3	1	7	7	7	7	7	7	6.17
SVM linear	4	1	1	1	6	3	3	6	6	6	5	3	3.75
SVM radial	6	6	6	6	5	5	1	4	4	4	6	6	4.92
RF	5	3	4	4	4	6	2	3	3	3	4	4	3.75
PPR supsmu	1	2	5	5	1	4	5	5	1	5	1	5	3.33
PPR spline	3	4	3	3	2	7	6	1	5	1	3	1	3.25
PPR gcvspline	2	5	2	2	7	2	4	2	2	2	2	2	2.83

- Some variables, namely, day of the year, week day and week of the year, can be defined as numeric or as symbolic;
- Some continuous variables can be discretized by the use of the integer part when expressed in different scales, namely:
  - ∗ departure time: seconds or minutes;
  - ∗ travel time: seconds or minutes.
- Other variables can cluster their values. Then, the cluster identifier is used instead of the original value [3]. This is particularly relevant for variables with higher cardinality. Possible variables to be clustered are: driver, service, day of the year and week of the year.

Of all these tests, we have only done the one on the data type for the variable week day. The remainder will be explored in future work. Results for SVM using different kernels present the same behavior. Results for the linear kernel are shown in Fig. 8. For the radial kernel, see [37], Appendix A. The use of the numeric data type for the variable week day when using SVM is not promising.

For RF using numeric or symbolic type for the variable week day yields similar results (Fig. 9) for any value of the *mtry* parameter.

For PPR (Fig. 10 and [37], Appendix A) results vary erratically as different parameters are used. For this reason, the same tests were done using the leaf node and the equivalent day approaches for example selection. Results for the super smoother are presented in Figs 11 and 12. Results for the other smoothers are in [37], Appendix A.

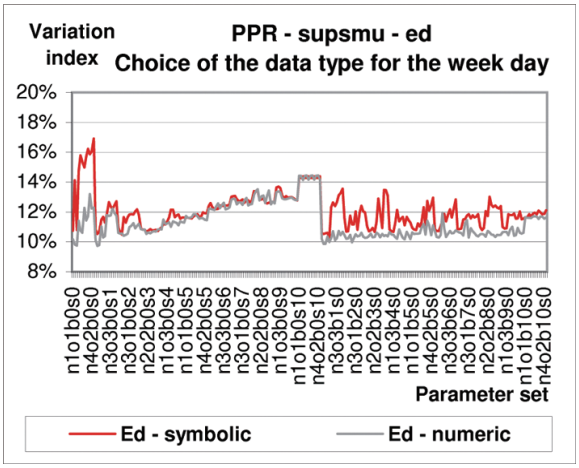

Fig. 12. The *variation index* for PPR – supsmu using different data types for the variable week day and the equivalent day approach for example selection.

In all these figures on domain values definition, the series 'Symbolic' are the same as the series 'All' presented in Section 5.2. This happens because in all the experiments on varying parameters, we used the symbolic data type for the variable week day.

6. Analysing results

After the three pre-processing tasks the best result for each algorithm improved. However, the best setting is algorithm dependent (Table 7). For SVM, the best setting is the leaf node approach for example selection and the use of a symbolic data type for the variable week day. RF is not very sensitive to the different settings tested. The results for PPR were the most erratic ones. The best setting for PPR, whichever the smoother used, was obtained using the equivalent days approach for example selection and the numeric data type for the variable week day.

The statistical validation of these results was done using the Friedman rank test with the statistic derived by Iman and Davenport as described in [13]. We have compared the best predictor of each algorithm, i.e., seven different predictors. The 1788 trips tested were divided in 12 groups according to their time order. The ranks obtained are shown in Table 8. The p value for the null hypothesis of equivalence between the seven predictors is 0.0008. Comparing the six predictors against the baseline for a 5% significance level, it is proved that all algorithms but SVM radial are better than the baseline.

These results show that the study on the best parameter set should be done together with the studies on each of the focusing tasks. This is not the usual procedure, probably due to the large amount of time required by such experiments. However, doing parameter tuning before the focusing tasks does not guarantee the best parameter set. This is applicable whichever the dataset is.

7. Conclusions

The main purpose of this research was to assess the predictive ability of different methods on long term travel time prediction. We have used three state-of-the-art induction algorithms for the regression

problem: Projection Pursuit Regression (PPR), Support Vector Machines (SVM) and Random Forests (RF). We have done an extensive search for the best parameters. Additionally, we have done experiments on the three pre-processing tasks in order to increase accuracy. These tasks are: feature selection, example selection and domain values definition. The main results of all these experiments are:

- RF is the most appealing method from an off-the-shelf point-of-view. Without any pre-processing it obtains competitive results. Furthermore, it has only one input parameter that is two-bounded.
- PPR is the method with the best overall result, but it is obtained after an extensive search of the right parameter set and after different experiments on pre-processing tasks.

This study assumes that, in order to use this approach in practice, the work described should use data from one year long, at least. This would guarantee a representative sample of the problem. There is no guarantee that, for other routes, the results would be the same. However, we believe that only the parameter tuning would vary. In fact, using always the same features and methods for example selection favors the choice of algorithms that best behave with such approach. However, this problem could be studied as a data streaming problem. In such study, the online choice of examples, features and parameter set should be addressed. This could be done using ensemble learning approaches in order to dynamically choose the most promising models [53].

The use of long term TTP in order to enhance transportation services is still very incipient or even unexisting. Processes for gathering data in freight transports must be developed. For public transport companies the problem is diverse. There is already data but the operational processes in companies do not take advantage of it. The study on the use of long term TTP for operational purposes is still very limited. But there is a strong perception that some improvements can be obtained by its use. This is an excellent research topic for the future. We believe that, despite the current operational difficulty for using long term predictive methods, the experimental proof of their predictive ability above currently employed methods, may foster its wider application in companies.

This paper is organized according to the usual steps given by any practitioner on applied supervised learning, namely, choice of the algorithms, tuning parameters and focusing tasks [41]. Despite some of the methods we have used are problem dependent, namely the baseline approach, we believe that this paper is useful not only for researchers on travel time prediction but also for researchers on applied supervised learning.

Acknowledgments

This work is funded by the ERDF through the Programme COMPETE and by the Portuguese Government through FCT - Foundation for Science and Technology, project ref. PTDC/EIA-EIA/098355/2008.

References

[1] C.G. Atkeson, A.W. Moore and S. Schaal, Locally weighted learning, *Artificial Intelligence Review* (11) (1997), 11–71.
[2] I.A. Basheer and M. Hajmeer, Artificial neural networks: fundamentals, computing, design, and application, *Journal of Microbiological Methods* **43** (2000), 3–31.
[3] Y. Benjamini and M. Igbaria, Clustering categories for better prediction of computer resources utilizations, *Applied Statistics* **40**(2) (1991), 295–307.
[4] L. Breiman, Bagging predictors, *Machine Learning* **26** (1996), 123–140.
[5] L. Breiman, Random forests, *Machine Learning* **45** (2001), 5–32.
[6] L. Breiman, Manual – setting up, using, and understanding random forests v4.0. Technical report, 2003.

[7] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and regression trees*, Chapman and Hall/CRC, 1984.
[8] A. Ceder, Urban transit scheduling: framework, review and examples, *Journal of Urban Planning and Development* **128**(4) (2002), 225–244.
[9] A. Ceder, B. Golany and O. Tal, Creating bus timetables with maximal synchronization, *Transportation Research Part A* **35** (2001), 913–928.
[10] S. Chien and C. Kuchpudi, Dynamic travel time prediction with real-time and historic data, *Journal of Transportation Engineering* **129**(6) (2003), 608–616.
[11] W.S. Cleveland and C. Loader, Smoothing by local regression: principles and methods, in: *Statistical theory and computational aspects of smoothing*, W. Hardle and M.G. Schimek, eds, Physica Verlag, Heidelberg, 1996.
[12] D. Cukjati, M. Robnik-Šikonja, S. Reberšek, I. Kononenko and D. Miklavčič, Prognostic factors in the prediction of chronic wound healing by electrical stimulation, *Medical and Biological Engineering and Computing* **39**(5) (2001).
[13] J. Demsar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research* **7** (2006), 1–30.
[14] T.G. Dias, *A new approach to the bus driver scheduling problem using multiobjective genetic algorithms*, Phd, 2005.
[15] T.G. Dietterich, Machine-learning research: four current directions, *AI Magazine* **18**(4) (1997), 97–136.
[16] J. Friedman, Multivariate adaptive regression splines, *The Annals of Statistics* **19**(1) (1991), 1–141.
[17] J.H. Friedman, Greedy function approximation: A gradient boosting machine, *Annals of Statistics* **29**(5) (2001), 1189–1232.
[18] J.H. Friedman and W. Stuetzle, Projection pursuit regression, *Journal of American Statistical Regression* **76**(376) (1981), 817–823.
[19] M.A. Hall, Correlation-based feature selection for discrete and numeric class machine learning, In *International Conference on Machine Learning*, 2000, pp. 359–366.
[20] T. Hastie, R. Tibshirani and J.H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Springer series in statistics. Springer, 2001.
[21] Y. Huang, P. McCullagh and N. Black, Feature selection via supervised model construction, In *International Conference on Data Mining*, 2004.
[22] P.J. Huber, Projection pursuit, *Annals of Statistics* **13**(2) (1985), 435–475.
[23] W. Inmon, *Building the data warehouse*, Wiley, 1996.
[24] X. Jin, Y. Deng and Y. Zhong, Mixture feature selection strategy applied in cancer classification from gene expression, In *IEEE Annual Conference on Enginnering in Medicine and Biology*, Shanghai – China, 2005, pp. 4807–4809.
[25] X. Jin, R. Li, X. Shen and R. Bie, Automatic web pages categorization with relieff and hidden naive bayes, In *SAC*, Seoul – Korea, 2007. ACM, pp. 617–621.
[26] G.H. John, R. Kohavi and K. Pfleger, Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, Morgan Kaufmann Publishers, 1994.
[27] K. Kira and L.A. Rendell, A practical approach to feature selection, In *International Wokshop on Machine Learning*, Morgan Kaufmann, 1992, pp. 249–256.
[28] D.G. Kleinbaum, L.L. Kupper, K.E. Muller and A. Nizam, *Applied regression analysis and other multivariablte methods*, Duxbury Press, 1998.
[29] G. Klunder, P. Baas and F.O.D. Beek, A long-term travel time prediction algorithm using historical data. Technical report, TNO, 2007.
[30] R. Kohavi and G.H. John, Wrappers for feature subset selection, *Artificial Intelligence* **97**(1–2) (1997), 273–324.
[31] A. Liaw and M. Wiener, Classification and regression by randomforest, *R News* **2**(3) (2002), 18–22.
[32] H. Liu, H. Motoda and L. Yu, A selective sampling approach to active feature selection, *Artificial Intelligence* **159**(1-2) (2004), 49–74.
[33] H.R. Lourenço, J.P. Paixão and R. Portugal, The crew-scheduling module in the gist system. Tech. Report UPF Economics Working Paper No. 547, Universitat Pompeu – Spain, 2001.
[34] S. Makridakis, C. Wheelwright and R.J. Hyndman, *Forecasting: methods and applications*, Wiley, 1998.
[35] D. Meyer, F. Leisch and K. Hornik, The support vector machine under test, *Neurocomputing* **55**(1–2) (2003), 169–186.
[36] L.C. Molina, L. Belanche and A. Nebot, Feature selection algorithms: a survey and experimental evaluation. In *IEEE International Conference on Data Mining*, 2002, pp. 306–313.
[37] J.M. Moreira, *Travel Time Prediction for the Planning of Mass Transit Companies: a Machine Learning Approach*, Ph.d. thesis, 2008.
[38] J.M. Moreira, A.M. Jorge, C. Soares and J.F. Sousa, Improving svm-linear predictions using cart for example selection. In *International Symposium on Methodologies for Intelligent Systems*, volume LNAI 4203, Springer, 2006, pp. 632–641.
[39] J.M. Moreira, A.M. Jorge, J.F. Sousa and C. Soares, Trip time prediction in mass transit companies. a machine learning approach, in: *Adavnced OR and AI methods in transportation*, A. Jaszkiewicz, M. Kaczmarek, J. Zak and

M. Kubiak, eds, Poznan, Publishing House of Poznan University of Technology. http://www.liaad.up.pt/~amjorge/docs/Triana/Moreira05a-EWGT05.pdf, 2005, pp. 276–283.

[40] J.M. Moreira, C. Soares, A.M. Jorge and J.F. d. Sousa, The effect of varying parameters and focusing on bus travel time prediction, In *the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining* (*PAKDD 2009*), volume LNAI 5476, Bangkok – Thailand, 2009, pp. 689–696.

[41] T. Reinartz, A unifying view on instance selection, *Data Mining and Knowledge Discovery* **6**(2) (2002), 191–210.

[42] M. Robnik-Šikonja and I. Kononenko, Theoretical and empirical analysis of relieff and rrelieff, *Machine Learning* **53**(1-2) (2003), 23–69.

[43] W.R. Rudnicki, M. Kierczak, J. Koronacki and J. Komorowski, A statistical method for determining importance of variables in an information system, In *International Conference on Rough Sets and Current Trends in Computing*, volume LNAI 4259, Springer-Verlag, 2006, pp. 557–566.

[44] B. Scholkopf, A.J. Smola, R.C. Williamson and P.L. Bartlett, New support vector algorithms, *Neural Computation* **12** (2000), 1207–1245.

[45] A.J. Smola and B. Scholkopf, A tutorial on support vector regression, *Statistics and Computing* **14** (2004), 199–222.

[46] J.G. Strathman, K.J. Dueker, T. Kimpel, R. Gerhart, K. Turner, P. Taylor, S. Callas, D. Griffin and J. Hopper, Automated bus dispatching, operations control and service reliability: analysis of tri-met baseline service date. Technical report, University of Washington – U.S.A., 1998.

[47] D. Tao and X. Tang, Svm-based relevance feedback using random subspace method, In *IEEE International Conference on Multimedia and Expo*, 2004, pp. 269–272.

[48] R.D.C. Team, R: A language and environment for statistical computing. Technical report, R Foundation for Statistical Computing, 2006. ISBN 3-900051-07-0.

[49] L. Torgo, Regression data repository, unknown. Retrieved October 20, 2005, from http://www.liacc.up.pt/~ltorgo/Regression/DataSets.html.

[50] S.M. Turner, W.L. Eisele, R.J. Benz and D.J. Holdener, Travel time data collection handbook. Technical Report FHWA-PL-98-035, Texas Transportation Institute, 1998.

[51] V.N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag, 1995.

[52] V.R. Vuchic, *Urban transit: operations, planning, and economics*. Wiley, 2005.

[53] H. Wang, W. Fan, P.S. Yu and J. Han, Mining concept-drifting data streams using ensemble classifiers. In *ACM International Conference on Knowledge Discovery and Data Mining*, 2003.

[54] D.R. Wilson and T.R. Martinez, Improved heterogeneous distance functions, *Journal of Artificial Intelligence Research* **6** (1997), 1–34.