An Experience in using Machine Learning for Short-term Predictions in Smart Transportation Systems

Davide Bacciu[a], Antonio Carta[a], Stefania Gnesi[b], Laura Semini[a,b,*]

[a] *Dipartimento di Informatica, Università di Pisa*
Largo Bruno Pontecorvo 3, I-56127 PISA, Italy
[b] *Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", CNR*
Via G. Moruzzi 1, 56124 Pisa, Italy

Abstract

Bike-sharing systems (BSS) are a means of smart transportation with the benefit of a positive impact on urban mobility. To improve the satisfaction of a user of a BSS, it is useful to inform her/him on the status of the stations at run time, and indeed most of the current systems provide the information in terms of number of bicycles parked in each docking stations by means of services available via web. However, when the departure station is empty, the user could also be happy to know how the situation will evolve and, in particular, if a bike is going to arrive (and viceversa when the arrival station is full).

To fulfill this expectation, we envisage services able to make a prediction and infer if there is in use a bike that could be, with high probability, returned at the station where she/he is waiting. The goal of this paper is hence to analyse the feasibility of these services. To this end, we put forward the idea of using Machine Learning methodologies, proposing and comparing different solutions.

Keywords: Machine learning techniques, Prediction, Bike-sharing systems

1. Introduction

Bike-sharing systems (BSS) are a sustainable means of smart transportation with a positive impact on urban mobility. The design of a BSS is multi-faceted and complex. First of all, BSS are composed of many components, among which bikes and stations, but also human users. The latter form an intrinsic part of the BSS and their individual patterns of behaviour have a decisive impact on the collective usability and performance of a BSS, which is highly dynamic.

An important characteristic of BSS, as for any other smart transportations system, is to avoid those situations which may generate distress among users

*Corresponding author
Email addresses: bacciu@di.unipi.it (Davide Bacciu), a.carta@outlook.it (Antonio Carta), stefania.gnesi@isti.cnr.it (Stefania Gnesi), semini@di.unipi.it (Laura Semini)

and discourage them from using the BSS. In an online survey conducted by Froehlich [1] on the *Bicing* BSS in Barcelona in 2009 about the experience of users with bike sharing, it turned out that 75% of the users stated commuting as a motivation to sign up for membership. Moreover, the same users identified "finding an available bike and a parking slot" as the two most important problems encountered (76% and 66% of the 212 respondents, respectively). These problems should therefore be addressed in the best possible way within the obvious budget constraints of cities, while keeping the number of kilometers made by vans that are involved in the (unavoidable) redistribution of bikes as small as possible. Furthermore, to improve the satisfaction of a user of the BSS, it is useful to inform her/him on the status of the stations at run time. Most of the available systems currently provide the information in terms of number of bicycles parked in each docking stations by means of services available via web. However, when the departure station is empty, a BSS user could be happy to know how the situation will evolve in the next few minutes and if a bike is going to arrive. Viceversa, when an arrival station is full, a prevision on the expected pick-ups at that time of the day would be very useful for the users.
To fulfill these expectations, we envisage services able to make predictions inferring the expected movements towards and from a given docking station. Indeed, several possible smart predictive services can be identified for improving the satisfaction of a user of a BBS using the available information and data.

The goal of this paper is hence to analyse the feasibility of these services. To this end, we put forward the idea of using Machine Learning (ML) methodologies that permit to build predictive models from BSS historical data proposing and comparing different solutions.

The advantage of a ML approach is twofold: on the one hand, such methodologies provide a powerful and general means to realize a wide choice of predictive services for which there exist sufficient (and significant) historical data. On the other hand, trained ML models are provided with a measure of predictive performance that can be used as a metric to assess the cost-performance trade-off of the service. This provides a way to analyze the runtime behavior of different variants of the preview service before putting it into operation. The above mentioned variability depends on the nature of the data used to train the ML models, along two dimensions: the amount (in months) of usage data needed to build a stable model, telling when it is reasonable to deploy a preview service in an existing BSS; the way data are selected and aggregated to build the ML models, different choices returning different predictive performances. The results that we obtain can serve to vendors and administrators of BSSs that plan to deploy the service in their town.

The idea underlying this paper is to assess the exploitation of mature ML methodologies as a design tool, hence we consider them from a software engineering perspective. In this sense, our attention focuses more on reusing as much as possible cheap and consolidated ML technologies that we deem adequate for the problem, resorting as much as possible to general off-the-shelf solutions rather than developing from scratch new learning models tweaked to maximum performance on a specific dataset, as advocated also in [2].

This paper extends [3], where these ideas were first put forward, by providing an in-depth experimental assessment of the preview service realization by different ML models, by comparing several configurations and alternative means to aggregate the historical data with different predictive performances and implementation costs. Precisely, the paper is organised as follows. After presenting some background, we introduce the case study (Sections 2 and 3). In Section 4 we discuss the methodology used. The setup for the experiments is in Section 5, and results in Section 6. Finally, in Section 7 we assess the preview services on the bases of the results obtained, and in Section 8 we discuss some lessons learned. In Section 9 we compare our work with related ones, and in Section 10 we draw some concluding remarks.

2. Background: Machine Learning

Machine learning (ML) is an active and wide research field comprising several paradigms, e.g. neural-inspired, probabilistic, kernel-based approaches, and addressing a variety of computational learning task types [4, 5]. For the purpose of the BSS modeling and evaluation, we focus on ML models and algorithms targeted at solving *supervised learning tasks*. Supervised learning refers to a specific class of ML problems that comprise learning of an (unknown) map $M : \mathcal{X} \rightarrow \mathcal{Y}$ between input information $x \in \mathcal{X}$ (e.g. a vector of attributes) and an output prediction $y \in \mathcal{Y}$ (in general, a vector of different dimensionality with respect to the input). Such an unknown map is learned from couples $\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of input-output data, referred to as *training examples*, following a numerical routine targeted at the optimization of an error/performance function $E(\mathcal{D})$ which measures the quality of the predictions generated by the ML model.

The actual form of the performance measure $E(\mathcal{D})$ depends on the nature of the learning task, but it typically evaluates the discrepancy between the output predicted by the learning models and the desired (ground-truth) output. The Mean Absolute Error (MAE) is a popular choice to estimate the performance in regression task as the absolute value of the difference between the model output and the expected target output, averaged over the number of samples under consideration. For classification tasks, performance is often assessed as class *accuracy*, *precision* or *recall*

$$acc_i = \frac{TP_i + TN_i}{N_i} \qquad prec_i = \frac{TP_i}{TP_i + FP_i} \qquad rec_i = \frac{TP_i}{TP_i + FN_i}$$

where N_i is the number of samples in the i-th class, while TP_i, TN_i, FP_i, and FN_i are the number of true positive, true negative, false positive and false negative (resp.) classifications predicted by the model for the i-th class.

A combined evaluation of the precision and recall can be obtained through the balanced F-score, or F1-score, allowing to take into consideration into a single score both true positives as well as false positives and negatives, that is

$$F1_i = 2\frac{prec_i \cdot rec_i}{prec_i + rec_i}$$

where higher $F1$ values denote better classification performances.

ML models are characterized by a three-step process to build effective predictive learning models and reliably assess their performance, followed by a fourth operational phase, where the built model is finally used:

1. *Training* (or *learning*) phase, where ground-truth teaching information (encoded in training samples) is used to adapt the parameters regulating the response of the model so that its error (performance) $E(\mathcal{D})$ is reduced (increased, respectively).
2. *Model selection*, which consists in estimating performance achieved by different learning models, including different hyper-parameter settings (i.e. model-tuning parameters set by the developer), in order to select the best model (with respect to the performance function).
3. *Final assessment*, which consists in evaluating the performance of the selected model on new data, providing a measure of the generalization performance of the ultimately chosen model. This step can be interpreted as a robust estimation of the performance of the feature implemented by the learning model when deployed in the run-time system.
4. The *testing* (or *prediction*) phase, instead, supplies a trained model with novel input information (typically unseen at training time) to generate run-time predictions (i.e. to compute the learned map on novel data). This phase is not always distinct: *incremental learning* approaches exist that allow to continuously adapt the parameters of a ML model while this keeps providing its predictions in response to new input data.

3. A Case Study: Pisa Bike-Sharing System

Many cities are currently adopting fully automated public BSS as a green urban mode of transportation. The concept is simple: a user arrives at a station, takes a bike using e.g. RFID card, uses it for a while and returns it to another station. Most of the BSSs permit to monitor the status of the stations, recording the user ID, the bicycle ID, date&time and departure station and stall, to control if borrowed bikes are returned completing the record of the ride adding arrival date&time, name of the station and stall number.

In the context of an European project, QUANTICOL[1], we started a collaboration with PisaMo S.p.A., an in-house public mobility company of the Municipality of Pisa, which introduced the BSS *CicloPi* in Pisa three years ago. This BSS was supplied by Bicincittà S.r.l.[2], it currently controls roughly 140 bikes and 15 stations, distributed over the town as shown in Figure 1. Some of them are marginal and less used, others support heavy traffic, the ratio of pickups in these stations being 1 to 20, approximately.

Currently, *CicloPi* offers two basic services, one reserved to its administrators and one meant for its users. The first, which we call *Bikes Hired*, registers,

[1] `http://www.quanticol.eu`
[2] `www.bicincitta.com`

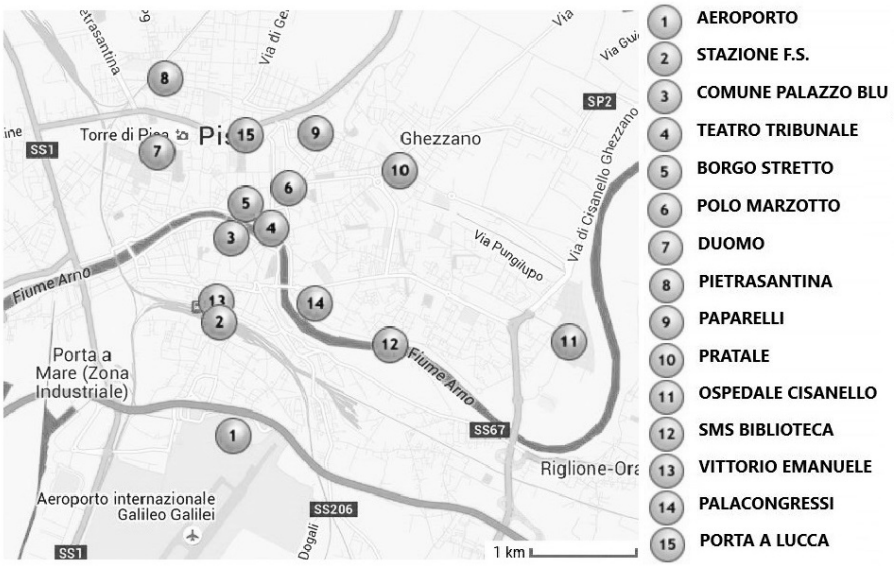

Figure 1: Location of the stations in Pisa. The most used are the Railway Station, *Stazione F.S.*, the University campus *Polo Marzotto*, and *Borgo Stretto*, in the center of the town. The more remote stations are the least used: the airport; the hospital, *Cisanello*; the park&ride *Pietrasantina*; the library, *SMS*; and the residential neighborhood of *Porta a Lucca*. The other stations are: town hall, *Comune Palazzo Blu*; theater and law court, *Teatro e Tribunale*; the leaning tower and cathedral area, *Duomo*; the weekly market, *Paparelli*; another park&ride, *Pratale*; *Vittorio Emanuele*, next to the railway station; *Palacongressi*, where there is another University campus.

for each hired bike, the user ID and the departure station and time of departure. It is needed to control whether all bikes are returned and it makes it possible to offer a billing service to the administrators (e.g. a fee applies for rides longer than 30 minutes). The second basic service of *CicloPi* is one that we call *Stations Snapshot*. It allows the user to perform a real-time check, for each docking station, of the availability of bikes at that station. Users can access this service using a Web browser or their mobile phone to find the closest station with an available bike (or with a free slot, for returning a bike) before actually going there. To realize this service the BSS uses data recored at run-time reporting on the number of available stalls per station.

4. Machine Learning for Improving BSS Users Services

Several possible smart predictive services can be identified for improving the satisfaction of a user of a bike-sharing system using the current available information and data, maintained by *CicloPi*: the number of free stalls; the users currently piking and returning a bike; the time and date of the operations. Furthermore, historical data on past hires are stored and we call *History* the

feature of the BSS maintaining these data. These data can be stored in two different sets by means of the two features:

1. *User History* collecting and maintaining the data of all hires of each user.
2. *Station History* collecting instead the history of all hires station-to-station, with date and time, without user ID[3].

In our study, we received real-world usage data covering all hires in Pisa across two years and comprising more than 280.000 entries, each corresponding to one hire.
These entries have the form:

$$\langle \texttt{UserID}, \underbrace{\texttt{station}, \texttt{stall}, \texttt{date\&time}}_{departure}, \underbrace{\texttt{station}, \texttt{stall}, \texttt{date\&time}}_{arrival} \rangle$$

For instance, the following record says that User 12345 went from Duomo, stall number 3, to Pratale, stall number 5, on 2015-05-14, leaving at 9:42:03..., and the ride took approximately 11 minutes.

$$\langle \begin{array}{llllll} \texttt{12345}, & \texttt{Duomo}, & \texttt{3}, & \texttt{2015-05-14}, & \texttt{09:42:03.4230}, & \\ & \texttt{Pratale}, & \texttt{5}, & \texttt{2015-05-14}, & \texttt{09:53:15.2700} & \rangle \end{array}$$

If we want to realize predictive services we need to use part or all of the above information in order to understand the dynamics of a bike-sharing system: considering this information in isolation or in combination may be useful in order to implement more accurate services. A family of different *Location Preview* services can be envisaged and planned by resorting to ML methodologies to analyze usage patterns and define computational learning models of the respective features from logs of usage of the bike-sharing system.

More in details, *Location Preview* services are realized starting from the information provided by *Bikes Hired*, to know the bikes in use, and by means of three different features:

1. *Station Profile All*, which in turn uses *Station History* and is based on the training of a global ML model for all the stations of the BSS.
2. *Station Profile*, which builds on ML models specialized per station through *Station History*.
3. *User Profile*, which uses *User History* to learn user specific ML models.

The new services are all concerned with the prediction of the destination station of a circulating bike given the information on its pickup details. Operationally, these services are polled every time a bike pickup takes place: a suitable ML model is retrieved and fed with pickup information such as departure station, pickup time, and, optionally, user id of the customer picking up

[3]We have taken into account the possibility that the administrator of a BSS may not be entitled to profile users, even if anonymized, and that the coupling user-ride has to be forgotten after a while.

the bike. This way, the BSS system will constantly have an up-to-date picture of where the circulating bikes are most likely be headed, updated right at the very moment in which a new bike is picked up.

The differences among them are in the pickup information used, and in the model used to make the prediction, namely in the way the ML model integrates historical information to form the decisional knowledge upon which the prediction is built. Hence, at run-time, the prediction is obtained differently for each of the three features.

For *StationProfileAll*[4] this is straightforwardly obtained by feeding a unique ML model, built learning from *Station History*, with information on pickup time and station.

StationProfile requires selecting the trained learning model associated with the pickup station and feeding it with only information concerning pickup time.

Finally, *UserProfile* receives pickup information including the id of the customer picking up the bike in addiction to the departure station id and the pickup time. The system will retrieve the personalized learning model associated to this user, learned from *User History*, and supply the model with the received information on departure station and pickup time.

In other words, all these different services of the *Location Preview* family use the data supplied by *History* to build a ML model, but they differ in the aggregation/disaggregation of historical data and require or the information provided by *Station History* or by *User History* according to the type of profile. The first, *Station Profile All*, is the simplest to implement and maintain but, as we will show in the paper, it has the lowest predictive performance. On the other side, we'll see that *User Profile* has the highest development and maintenance cost, but the best performance.

4.1. Learning User and Station Profile Features

In this section, we analyze the nature of the tasks associated with the realization of the preview services and we discuss the supervised learning approaches suitable for modeling such features from logs of bike usage.

In *StationProfile*, for instance, the learning model is built specifically for each departure station, encoding predictive knowledge solely on the bike pickups originating there. A ML approach to realize this requires to train a different ML model for each station, using as training data only those records that have the station as origin of bike pickup. On the contrary, *StationProfileAll* builds a single model aggregating information for all the departure stations: this is obtained by training a single ML model using all available training data. *UserProfile*, on the other hand, is built to predict the destination station of a bike given knowledge of the BSS usage of the person who has it picked-up. A ML approach to realize such feature requires to train a different ML model for each user, using its personal usage logs as training data.

[4] From now on we refer to a service with the name of the feature we use to implement it.

The training examples for the three feature implementations described above share the same attribute structure: the dataset $\mathcal{D}$ (see Section 2) contains vector couples (x_n, y_n) where the input attributes x_n are a numerical encoding of the departure time and station, while the target y_n encodes the associated arrival station.

The prediction of the arrival station is an instance of a *classification problem* whose objective is to assign an input pattern to one of K different and finite classes, that are the bike stations in our case study. *Support Vector Machines* (SVMs) [6] are a popular family of supervised learning models that are widely used to perform classification (and regression) tasks. SVMs build on the concept of a linear separator (e.g. an hyperplane separating vectors in the positive class from those in the negative class in binary classification) and extend it to deal with non-linear problem by exploiting the so-called *kernel trick*, that is an implicit map of the input vector into a high-dimensional feature space by means of a non-linear map induced by a kernel function. For problems characterized by identically and independently distributed (i.i.d.) training vectors (such as our case study), the Gaussian kernel is often the most popular choice due its theoretical properties (i.e. the ability to define an infinitely dimensional implicit feature space) resulting in excellent predictive accuracies of the trained SVM [6].

Random Forests [7, 8] are another popular and effective approach to deal with regression and classification problems with i.i.d data, in particular when dealing with large sample size, thanks to their computational efficiency in training. Random Forests are based on the decision tree approach, where a tree describes a decisional process such that at each node of the tree a branching decision is taken based on the values of the attributes of the data. Simply put, if the attribute of the current sample is smaller than a threshold, then the decisional process continues on the left child of the current node, otherwise it progresses on the right. The thresholds used to perform the tests are determined during the learning phase and so does the structure of the tree. The branching process terminates when a leaf node is reached which, for a classification task, associates the predicted class to the current sample. The Random Forest classifier is an ensemble learning method that, during training, constructs multiple decision trees trained on the classification problem and, at prediction time, operates by submitting the current sample to all the trees in the forest and determining its final classification as the mode of the classes predicted by the trees.

SVM with Gaussian kernels and Random Forests are both highly effective classifiers for a wide-class of learning problems, with several stable implementations freely available (see [9–11] for SVM and [12–14] for Random Forests). For this reasons, we use them as reference models for our case study and, in the next Section, we show how they can be used to implement and assess *StationProfileAll*, *StationProfile* and *UserProfile*.

Training the learning models can also be performed at run-time: for instance, for *UserProfile*, when a new customer subscribes to the BSS, it starts collecting his/her usage information; as soon as sufficient data is collected, it is used to train a new learning model specific for the usage patterns of that customer. The same approach can be used to maintain the knowledge encoded in an existing

customer model up-to-date: new examples are added to the log as the customer uses the system and a re-training of the learning model is performed as soon as a sufficient amount of new data is collected. Similarly, this can be done for the (simpler to maintain) *StationProfile* and *StationProfileAll* features.

In the next sections, we will show how *StationProfileAll*, *StationProfile* and *UserProfile* can be learned from historical data on bike usage and discuss how the prediction performance computed on the machine learning models can be used to assess the performance-cost trade-off of the features before their deployment.

5. Data Preparation and Experimental Setup

Preprocessing is a fundamental preparatory step to produce good quality training data for the learning models. Data is processed to remove erroneous records and noisy samples, to either remove or impute missing observations, and, in general, to get rid of any observation that carries no informative content for the task, based on background knowledge on the application. The Pisa BSS usage data described in Section 3 has undergone a preprocessing step targeted at:

1. removal on uninformative attributes, e.g. ID of the departure/arrival bike stall (but not the station information);
2. removal of maintenance-related record, such as non-existing station codes and bike pickups associated to user identifiers of maintenance personnel (which moves the bikes to redistribute them not for actual usage);
3. removal of erroneous pickups, such as those lasting only a couple of seconds between departure and arrival;
4. parsing of the pickup timestamps to obtain an encoding suitable to highlight cyclic patterns and to account for seasonality in the data, i.e. transforming a timestamp to the corresponding weekday, month and daytime of pickup.

The attributes of the dataset resulting from the preprocessing step are summarized in Table 1. Table 2 shows the number of bikes delivered to each station in the cleaned dataset and the total number of *clean* records available.

5.1. Data Aggregation

Modeling of *StationProfile*, *StationProfileAll* and *UserProfile* is performed as a classification task, where the learning model is trained to classify an input representing a bike pickup into one of the classes identifying to the bike arrival stations in Table 2. Each bike pickup is encoded as a vector x_i containing a variable number of attributes (out of those in Table 1) depending on the feature type. *StationProfileAll* and *UserProfile*, for instance, use as input attributes both the information on bike pickup time encoded in the last four attributes in Table 1, as well as the `DepStation` attribute. *StationProfile*, instead, only uses the bike pickup time as the information on departure station is implicitly encoded in the fact that the model is trained on bike pickups specific for a departure station. All feature types use the `ArrStation` information (see Table 1)

Attribute	Type	Description
`UserID`	Int	Unique BSS user identifier
`DepStation`	Int	Numeric identifier for pickup station
`ArrStation`	Int	Numeric identifier for arrival station
`DepWeek`	Int	Day of the week of pickup ([1-7])
`DepMonth`	Int	Month of pickup ([1-12])
`DepHour`	Int	Pickup hour of the day ([0-23])
`DepMin`	Int	Pickup minute ([0-59])

Table 1: Attributes of the preprocessed Pisa BSS dataset.

as target class y_i for the bike pickup. Training of the learning models entails aggregating data differently into datasets depending on the specific feature being implemented. *StationProfile*, for instance, requires allocating one dataset for each departure station by partitioning data based on the `DepStation` attribute in Table 1. Similarly, *UserProfile* requires allocating one dataset for each user by partitioning data on the `UserID` attribute. Clearly, *StationProfileAll* does not require any partitioning of the dataset.

Table 2 provides a picture of the class membership and reports the classification performance of a baseline classifier returning predictions that are randomly sampled following the distribution of class labels in the dataset (hence, most frequent class label is predicted more often, proportionally to its popularity). The figures in Table 2 show a considerable amount of class imbalance, since certain arrival station appears considerably more popular than others. Class imbalance is a major issue to be taken into consideration when confronting with a classification task: Section 6 will discuss how this aspect affects the predictive quality of the BSS features. The preprocessed datasets resulting from the aggregation/partitioning operations described above have then been split into a training set, comprising 70% of the samples, and a test set, comprising 30% of the records, preserving the proportion of samples in each of the classes (i.e. stratification [15]). The training set would serve for training and model selection purposes, while the test set contains out-of-sample data that will be used for the sole purpose of assessing the future performance of the developed predictive feature, i.e. its generalization ability.

5.2. Learning Models and Hyperparameters Selection

We address the predictive modeling of the Station and User Profile features by resorting to effective off-the-shelf models in literature with computational requirements suitable for the size of the problem at hand. In this sense, we first consider addressing the predictive model by means of SVM classifiers with Gaussian kernel, using a one-versus-one approach to multi-classification. The experimental analysis with support vector classifiers is complemented with the results obtained by the Random Forest approach, another effective and consolidated machine learning model for supervised tasks that is quite popular in Big Data analytics due to the efficiency of its training phase (note that the SVM

Station	Sample number	Baseline classifier		
		prec	rec	F1
Aeroporto	1971	0.01	0.01	0.01
Borgo Stretto	36710	0.15	0.15	0.15
Comune Palazzo Blu	18956	0.07	0.07	0.07
Duomo	21773	0.08	0.08	0.08
Ospedale Cisanello	4814	0.02	0.02	0.02
Palacongressi	23467	0.09	0.09	0.09
Paparelli	14109	0.05	0.05	0.05
Pietrasantina	6645	0.03	0.03	0.03
Polo Marzotto	39509	0.13	0.13	0.13
Porta a Lucca	4796	0.01	0.01	0.01
Pratale	20368	0.08	0.08	0.08
Sms Biblioteca	5600	0.02	0.02	0.02
Stazione F.S.	43316	0.13	0.13	0.13
Teatro Tribunale	12227	0.05	0.05	0.05
Vittorio Emanuele	26647	0.09	0.09	0.09
Total/Average	280908	0.10	0.10	0.10

Table 2: Number of samples per classes (arrival stations) and performance of the baseline classifier returning random predictions following the station popularity (i.e. most popular class label is predicted more often, proportionally to its frequency in the dataset).

classifier is, on the other hand, more efficient on the prediction phase). For the sake of conciseness, results for the Random Forest approach are reported only for the most interesting *UserProfile*, where it will be noted how its performance is in line with that of the SVM approach.

The SVM model with Gaussian kernel is characterized by two meta-parameters which regulate its generalization performance, i.e. the misclassification cost parameter C and the Gaussian width γ whose values are determined by model selection (see Section 2) using a grid search on the following ranges of hyperparameter values (following typical choices in literature), i.e. $C \in \{4^{-3}, 4^{-2}, \ldots, 4^{5}\}$ and $\gamma \in \{4^{-8}, 4^{-7}, \ldots, 4^{2}\}$. For the Random Forest approach, we have performed a grid search on the following hyperparameters and associated values: number of trees in the forest $N_T \in \{5, 10, 20, 40\}$; maximum number of features considered when determining the best split $N_F \in 2, 3, 4$. A 5-fold cross validation is applied to the training set to compute the validation performance for different configurations of the meta-parameters: the best-performing configuration in validation is then selected and trained on the full training data before being subject to final assessment on the test set. Given the unbalanced nature of the dataset, performance is evaluated using the precision, recall and F1 scores discussed in Section 2. Data preprocessing has been realized using the `Pandas`[5] library, while the SVM and Random Forest models, as well as the cross-fold

[5] `http://pandas.pydata.org/`

validation procedures are based on the `scikit-learn` API[6].

The directory `https://github.com/AntonioCarta/ciclopi_model` contains the main python files used for preprocessing and data aggregation as well as for feature training.

6. Experimental Analysis

The preprocessed Pisa BSS usage data has been used to assess the implementation of different predictive features. The results of this analysis are summarized in the following subsections. Here, for the sake of conciseness, we only report performance results on the test-set obtained by the best model selected in the validation phase. Overfitting and generalization performance has been carefully monitored by adopting a robust model selection and cross-fold validation approach. In particular, note that the difference in performance between the test data and the validation sets stayed under 0.2% for *StationProfile* and 2.6% for *UserProfile*, which are normal performance fluctuations that can be expected due to sample randomization and do not indicate any overfitting issue.

6.1. Station Profile Features

We have considered two alternative solutions for learning from the usage data stored by *Station History*: building a single predictive model aggregating knowledge for all the departure stations, as in feature *StationProfileAll* or building a different predictive model for each departure station as in *StationProfile*. The former requires maintenance of a single predictive model and has hence a smaller deployment impact than the second that will require to deploy and maintain a different predictive model for each BSS station. Performances also differ, as shown in Table 3, reporting the test-set performance of the Gaussian SVM classifier that implements these features.

In the case of *StationProfileAll*, two results are shown, one for the standard SVM predictor and one for a predictor trained using a sample reweighing scheme to penalize errors for less popular destination stations (i.e. penalizing errors inversely proportional to class frequencies). The station-specific predictor *StationProfile*, on the other hand, is obtained by training the learning model only on data corresponding to bike pickups occurring at the target station. Similarly, at prediction time (and during testing), the model will be queried every time there is a pickup at its target station (which is a known information in the system), to predict where the bike is headed. For the sake of conciseness, results for *StationProfile* are reported only for the Gaussian SVM classifier with class reweighing to contrast class imbalance (models have also been trained without imbalance-prevention mechanisms with no better results hence are omitted here for brevity). Table 3 shows, in the rightmost column, the test-set performance for *StationProfile* integrating the predictions of all the 15 station specific Gaussian SVM classifier trained with class imbalance reweighing. These are obtained

[6] `http://scikit-learn.org`

	StationProfileAll						StationProfile		
	no reweighing			imbalance reweighed					
Station	prec	rec	F1	prec	rec	F1	prec	rec	F1
Aeroporto	0.00	0.00	0.00	0.08	0.05	0.06	0.09	0.16	0.12
Borgo Stretto	0.20	0.36	0.26	0.21	0.30	0.25	0.28	0.28	0.28
Comune Pal. Blu	0.18	0.03	0.06	0.16	0.17	0.17	0.28	0.27	0.27
Duomo	0.23	0.04	0.07	0.21	0.22	0.21	0.31	0.33	0.32
Ospedale Cisanello	0.00	0.00	0.00	0.11	0.09	0.10	0.17	0.15	0.16
Palacongressi	0.14	0.04	0.07	0.17	0.16	0.16	0.24	0.21	0.23
Paparelli	0.16	0.03	0.05	0.15	0.14	0.15	0.24	0.24	0.24
Pietrasantina	0.19	0.02	0.03	0.17	0.14	0.15	0.20	0.33	0.25
Polo Marzotto	0.28	0.51	0.36	0.30	0.29	0.30	0.36	0.40	0.38
Porta a Lucca	0.00	0.00	0.00	0.15	0.11	0.13	0.11	0.25	0.15
Pratale	0.17	0.04	0.06	0.18	0.17	0.18	0.27	0.27	0.27
Sms Biblioteca	0.00	0.00	0.00	0.07	0.05	0.06	0.13	0.12	0.12
Stazione F.S.	0.26	0.63	0.37	0.28	0.26	0.27	0.39	0.37	0.38
Teatro Tribunale	0.00	0.00	0.00	0.13	0.10	0.11	0.18	0.13	0.15
Vittorio Emanuele	0.14	0.09	0.11	0.17	0.15	0.16	0.27	0.22	0.24
Average	0.19	0.24	0.18	0.20	0.21	0.20	0.29	0.28	0.28

Table 3: On the left part of the table, predictive performance on the hold-out test-set for *StationProfileAll* with a single model for all stations with and without reweighing to contrast class imbalance. Results refer to the best Gaussian-SVM model identified by grid search using the 5-fold cross-validation error and having hyperparameters $C = 0.5$ and $\gamma = 4$. On the right part of the table, predictive performance on the hold-out test-set for *StationProfile* with a model trained specifically for each station (using class imbalance reweighting). The hyperparameterization of each station-specific model has been chosen independently by grid search on the cross-validation error and cannot be reported here due to the large number of user-specific predictive models.

	SVM			Random Forest		
Station	prec	rec	F1	prec	rec	F1
Aeroporto	0.61	0.49	0.54	0.57	0.50	0.53
Borgo Stretto	0.52	0.57	0.54	0.56	0.57	0.56
Comune Palazzo Blu	0.63	0.54	0.58	0.58	0.56	0.57
Duomo	0.65	0.64	0.65	0.65	0.67	0.66
Ospedale Cisanello	0.68	0.65	0.67	0.70	0.71	0.71
Palacongressi	0.66	0.64	0.65	0.67	0.67	0.67
Paparelli	0.63	0.59	0.61	0.62	0.61	0.61
Pietrasantina	0.69	0.67	0.68	0.71	0.70	0.70
Polo Marzotto	0.67	0.72	0.69	0.68	0.71	0.69
Porta a Lucca	0.62	0.56	0.59	0.59	0.56	0.57
Pratale	0.71	0.76	0.73	0.75	0.76	0.75
Sms Biblioteca	0.58	0.57	0.58	0.60	0.58	0.59
Stazione F.S.	0.69	0.72	0.70	0.70	0.71	0.71
Teatro Tribunale	0.58	0.50	0.54	0.57	0.52	0.54
Vittorio Emanuele	0.57	0.52	0.54	0.55	0.52	0.53
Average	0.63	0.63	0.63	0.64	0.64	0.64

Table 4: Predictive performance on the hold-out test-set for *UserProfile* implemented by Gaussian-SVM and Random Forests. The hyperparameterization of each personalized predictive model has been chosen independently by grid search on the cross-validation error and cannot be reported here due to the large number of user-specific predictive models.

by using the station-specific model of the test-bike pickup station to classify each sample in the test set.

6.2. User Specific Features

The experimental analysis on the *StationProfile* features suggests that we should consider building a different, more articulated predictive model, that can capture more fine-grained usage behaviours. To this end, we focus on evaluating the implementation of *UserProfile*, which provides personalized predictive models at the level of the single user behaviour. Such an approach requires to build a predictive model for each BSS customer for which there exists sufficient usage data. In particular, we begin our analysis by building a personalized model only for those users who have at least 20 bike pickups; the remainder of the data is used to train a generic model aggregating data from sporadic BSS users. The dataset includes information from a total of 2999 users, where 1835 are regular users with more than 20 pickups, corresponding to 61% of the total. Note that the total amount of pickups associated with such regular users is 273150 out of a total of 281006. Hence, regular users make-up for 97% of the bike pickups suggesting that *UserProfile* has the potential to be used in a practical deployment of the system.

The test-set performance of the *UserProfile* configuration is shown in Table 4 both for the SVM-based predictor as well as for an alternative learning model, that is the Random Forest. Note that here it is not needed to use sample

	Working Day			Holiday		
Station	prec	rec	F1	prec	rec	F1
Aeroporto	0.62	0.47	0.54	0.52	0.44	0.48
Borgo Stretto	0.56	0.57	0.56	0.49	0.49	0.49
Comune Palazzo Blu	0.65	0.56	0.60	0.47	0.38	0.42
Duomo	0.65	0.67	0.66	0.48	0.41	0.44
Ospedale Cisanello	0.68	0.67	0.68	0.48	0.57	0.52
Palacongressi	0.66	0.67	0.66	0.53	0.59	0.56
Paparelli	0.64	0.61	0.63	0.40	0.38	0.39
Pietrasantina	0.70	0.70	0.70	0.56	0.55	0.55
Polo Marzotto	0.68	0.72	0.70	0.52	0.57	0.54
Porta a Lucca	0.64	0.57	0.60	0.52	0.36	0.42
Pratale	0.71	0.78	0.75	0.63	0.73	0.68
Sms Biblioteca	0.58	0.60	0.59	0.48	0.51	0.49
Stazione F.S.	0.69	0.74	0.71	0.44	0.38	0.40
Teatro Tribunale	0.60	0.52	0.56	0.45	0.46	0.46
Vittorio Emanuele	0.59	0.53	0.55	0.48	0.48	0.48
Average	0.64	0.65	0.64	0.50	0.50	0.50

Table 5: Breakup of *UserProfile* predictive performance between working days and non-working days (Sundays and holidays).

reweighing when training the models, as the performance of the single user models is not affected negatively by class imbalance.

In terms of comparison between the two machine learning models, Table 4 shows that Random Forest performance does not differ significantly from that achieved by Gaussian-SVM. There are fluctuations of a couple of decimal points on the score of the single stations, resulting in a difference of 1% on the average score, which seems more due to random fluctuations (order of splits in the tree, initialization aspects, etc) than to a significant advantage in predictive performance. This confirms the fact that the problem being tackled in this experiment is quite complex to solve and the moderately good predictive performances do not depend on the specific learning model adopted but rather on inherent characteristics of the data. Since there is no significant difference between the Gaussian-SVM and the Random Tree results, in the following we will continue the analysis using only the former model.

A key aspect to take into consideration when building personalized profiles of BSS usage is the effect of holiday and Sundays, as these days tend to break the regularities of the weekly and seasonal routines. To assess the extent of this phenomenon, Table 5 provides a break-up of the performance of *UserProfile* (whose aggregated results are in Table 4) in working days and holidays. Here it can be noted, perhaps unsurprisingly, that performance for working days is much higher than for holidays. However, the actual impact of such drop in performance during holidays is limited by the considerably lower BSS usage. Consider that the test set includes only 3677 pickups during holiday periods confronted with 66009 test-set pickups during working days. This also suggests that a pickup destination prediction service is going to be very less needed during

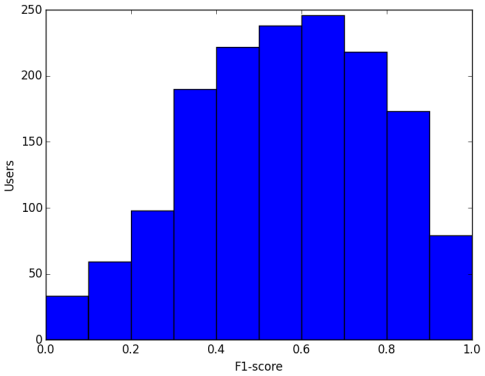

Figure 2: Distribution of users with respect to the F1-score of their personalized predictive model.

weekends and holidays due to the limited system usage, at least for the specific Pisa BSS. One last aspect needs to be assessed to complete the characterization of the performance of *UserProfile*, that is data support, which raises a number of questions: are there users whose behaviour is better predictable? What is the trade-off between the number of personalized models and the predictive performance? Figure 2, for instance, provides a view over the distribution of users with respect to the F1-score performance of their personalized predictions. There appears to be a consistent group of users whose usage behaviour is extremely well predictable with F1-scores over 0.95 which is likely to correspond to people using the Pisa BSS for daily commuting to work/school. Even more importantly, we would like to assess how the predictive performance of a personalized user profile changes when we increase the minimum number of bike pickups required to define a personalized predictive model. In Table 6, we show the performance achieved by *UserProfile* when personalized models are trained only for users characterized by at least 100 pickups and considering only stations with at least 5 bike deliveries (referred to as *UserProfile100*). As for the earlier case in this section, the remainder of the data not included in the personalized profiles is used to train a generic model aggregating data from sporadic BSS users. Table 6 shows that the higher quality of the filtered data used for the personalized predictors yields to another increase in the predictive performance which is now close to an overall $F1$-score of 0.7, suggesting that the final implementation of *UserProfile* would benefit from using only very consolidated usage data for the deployment of personalized predictive models.

7. Results Analysis and Features Assessment

We here analyse and discuss the experimental results obtained, dwelling on the most interesting feature, based on user profiling.

Station	prec	rec	F1
Aeroporto	0.65	0.64	0.64
Borgo Stretto	0.63	0.62	0.62
Comune Palazzo Blu	0.72	0.64	0.67
Duomo	0.69	0.69	0.69
Ospedale Cisanello	0.72	0.76	0.74
Palacongressi	0.71	0.71	0.71
Paparelli	0.70	0.69	0.69
Pietrasantina	0.75	0.75	0.75
Polo Marzotto	0.71	0.78	0.74
Porta a Lucca	0.69	0.60	0.64
Pratale	0.74	0.80	0.77
Sms Biblioteca	0.61	0.65	0.63
Stazione F.S.	0.73	0.77	0.75
Teatro Tribunale	0.65	0.56	0.60
Vittorio Emanuele	0.67	0.59	0.63
Average	0.69	0.69	0.69

Table 6: Predictive performance for *UserProfile* on filtered data plus a *StationProfileAll* predictor for left-out samples.

7.1. Assessing StationProfileAll and StationProfile

Results in Table 3 show a poor predictive performance of *StationProfileAll*, with a low average F1-score across the stations for the standard SVM predictor, although higher than in the baseline (see Table 2). Such low performance is partly motivated by the imbalanced nature of the data, which leads to a small number of highly popular stations being predicted with a good recall and F1-score (e.g. Stazione F.S. and Polo Marzotto), while the majority of the stations have zero (or close to zero) recall resulting in low overall precision. In this respect, the reweighing approach manages to slightly increase the average precision and F1-score by increasing the recall of less popular stations. At the same time, the overall performance of the reweighed *StationProfileAll* remains low, which suggests that class imbalance is not the sole factor influencing the accuracy and that a different feature design must be considered to achieve predictions of sufficient quality.

Analysing again Table 3, we can observe that the station-specific profile, *StationProfile*, yields to better results than the *StationProfileAll* approach, which confirms the intuition that data should be appropriately aggregated (in this case, on a per-station basis) in order to let regularities in the usage patterns emerge from the data. At the same time, it can also be noted that performance is not yet satisfactory to build an effective feature even when imbalance is taken into account and dealt with. All in all, this suggests that information on the pickup station and time alone is not always sufficient to identify regularities in the destination of the bike pickup when data is aggregated at the station level, and that, again, such a behaviour cannot be simply attributed to class imbalance.

7.2. Assessing User Profile

The results in Table 4 clearly show that aggregating data at the user level yields to a considerable increase with respect to using station-specific aggregation. In particular, it is quite interesting to note how *UserProfile* yields a prediction performance that is generally homogenously distributed between the stations, without sharp precision or F1-score variations.

The *UserProfile* feature has higher deployment costs compared to *StationProfile*, requiring training and maintenance of a consistent number of predictive models, which scales linearly with respect to the size of the BSS user population. At the same time, the quantitative assessment described so far anticipates that *UserProfile* will be the one characterized by the best effectiveness when deployed into the system. The predictive models used to perform such an analysis can be used to further understand the deployment-time behaviour of the system, e.g. by simulating usage cases for some example daytimes. Operationally, *UserProfile* can work as a service that it is polled every time a bike pickup takes place. The interface of this service will receive pickup information similar to that encoded in the dataset including, at least, the `UserID` of the customer picking up the bike (see Table 1), the departure station id and the pickup time. The system will retrieve the personalized learning model associated to the `UserID`, or the general one in case the user is not a regular one, and supply the model with the received information on departure station and time. By this means, the BSS system will constantly have an up-to-date picture of where the circulating bikes are most likely to be headed, updated right at the very moment in which a new bike is picked up.

We have simulated the activity of such a system using predictions of *UserProfile* queried on the 25th of February 2015 at 1p.m., which is a peak time for BSS usage. The second column in Table 7 shows the predictions returned by the system concerning 14 bikes circulating at that time of the day. One can see that, for instance, a user heading to the Borgo Stretto station might change her/his mind and deliver the bike to a nearby station (e.g. Teatro Tribunale) where she/he is more likely to find a free stall. At the same time, a person that could not find a bike at the train station (Stazione F.S.) might decide to wait a couple of minutes knowing that 2 bikes are likely to be on their way to the Stazione F.S. stalls.

The simulation above provides an example of how *UserProfile* can be exploited to provide a useful service for the BSS user which might be informed if at least a bike will be reaching the station in a reasonable time. The extent of the time the user has to wait can be straightforwardly estimated from the statistics of bike pickups in the dataset, for instance by computing the average renting time between the pickup station and the predicted arrival station. For a small city like Pisa, such average renting times vary very little depending on the departure/arrival stations and a simple statistics on the Pisa BSS data shows that the average pickup time is typically around 15 minutes. Of course, more refined time estimation policies can be defined, for instance taking into consideration personalized renting time predictions for each user, but they are

Station	Bikes Prediction 25/02/15:1am	Precision 15mins
Aeroporto	0	0.63
Borgo Stretto	7	0.71
Comune Palazzo Blu	0	0.73
Duomo	1	0.71
Ospedale Cisanello	1	0.74
Palacongressi	1	0.72
Paparelli	0	0.69
Pietrasantina	0	0.76
Polo Marzotto	1	0.75
Porta a Lucca	0	0.68
Pratale	1	0.78
Sms Biblioteca	0	0.64
Stazione F.S.	2	0.76
Teatro Tribunale	0	0.68
Vittorio Emanuele	0	0.67
Average	n.a.	0.73

Table 7: The table shows information on a simulated use of *UserProfile*. Second column shows the destination of the bikes circulating on the 25th of February 2015 at 1p.m. as predicted by the feature. Third column reports the precision (on the test-set) of *UserProfile* in predicting the arrival of a bike at a destination station within the 15 minutes timespan.

outside of the scope of this paper. What we are more interested in assessing here is what predictive quality we can expect by an actual usage of the service. In other words, we would like to evaluate if *UserProfile* can reliably estimate the arrival of a bike at a stall in the next 15 minutes, which is a metric providing information on the quality of service to be expected from the BSS user side. This aspect is shown in the third column in Table 7, which reports the performance of *UserProfile* measured in terms of the precision of the system in predicting the arrival of a bike at a certain station within the 15 minutes from its pickup time. To obtain this prediction in a reliable way, we have partitioned the days in the dataset in slices of 15 minutes to extract a proportion of 30% of hold-out samples on which to test the system (these samples have been obviously excluded from the training and validation sets). The average precision of such system with respect to all destination stations is 0.73 which is considerably higher than the 0.63 precision reported in Table 4 for the more challenging problem of predicting the destination station of a random circulating bike. This also shows the potential of *UserProfile* which addresses the complex predictive problem of providing a constantly updated picture of the circulating bikes' destination, rather than simply predicting the arrival of at least one bike at a certain station, that is the typical task addressed by other works in literature.

8. Lessons Learned

Our analysis has focused on the experimental assessment of a machine learning solution for implementing features of a bike destination preview service using

off-the-shelf computational learning models. In this sense, the goal of this work is not to propose a novel, specialized learning model specifically tuned for the problem of bike destination prediction on *CicloPi* data. Rather, we are interested in studying the task from a software engineering point-of-view reusing as much as possible cheap and consolidated machine learning technologies that we deem adequate for the problem. The ultimate aim is to provide indications that, combined with those of [16–18], can serve to vendors and administrators of BSSs to help planning the deployment of the service in their town by proposing a product line framework defining a family of advanced prediction services.

The experimental analysis in Section 6 and 7 has highlighted some key issues that deserve attention to implement the system and to optimize its predictive performance.

8.1. Development, deployment and maintenance costs

The previous sections provide indications that allow evaluating such services with respect to their predictive performance, upon which customer satisfaction depends. Development, deployment and maintenance costs are another relevant dimension upon which such services needs evaluating and that we are going to discuss in the following, mostly from a perspective of model deployment complexity and computational effort required by the training phase.

The computational cost for training the predictive models varies greatly between the features. *StationProfileAll* is perhaps the feature with the least deployment cost, as it requires maintaining a single model for all the BSS system, but it is also the most expensive to train, requiring roughly 200 hours to train a Gaussian-SVM classifier on the full Pisa BSS training set. The cost required to build the *StationProfile* feature is obtained by summing training times of the single station specific predictors. These times vary greatly depending on the popularity of the specific station, while the total training cost for *StationProfile* is slightly under 9 hours. Training of *UserProfile* requires the least amount of time, i.e. a total of 47 minutes for all the Gaussian-SVM learning models, despite being the most complex configuration to deploy (thousands of learning models to maintain). The implementation of the same feature with Random Forest is even less expensive (at learning time), requiring only 16 minutes to complete training on all the models. The time required to perform a prediction is negligible for all the models, although it is expected to be more expensive for the Random Forest model due to tree traversal time. All the timings reported above refer to learning models being trained and executed on a Server equipped with 4 Xeon E5-2620 (24 cores total) and 128 Gb of RAM. As a general comment on the computational cost, one can note how splitting the problem into a number of smaller ones, either on a per-station or per-user basis, yields to a considerable computational advantage which can have a considerable impact in terms of model maintenance. Retraining of the *StationProfileAll* on the basis of fresh data is in fact quite costly, whereas keeping up to date the single station predictors seems quite feasible. Even more so for *UserProfile*, notwithstanding the large number of personalized predictive models involved.

8.2. Data aggregation and timeseries considerations

The analysis suggests that the task of learning features performing short term predictions on bike usage in BSS is a complex one to solve due to the nature of the data and of the process being modelled. Only a very limited portion of data provide informative content that can be used to predict the destination station of a circulating bike. In particular, knowledge of other bikes circulating at the same time or information concerning what bikes were circulating before the pickup (e.g. by performing some form of timeseries analysis) cannot be expected to reduce the uncertainty of where a target bike is headed, as this is essentially a completely independent event from the pickup of other bikes. The experimental analysis has shown that aggregating data so to obtain specific usage profiles, and hence specialized features, may allow regularities in the data to emerge. Specifically, the aggregation of data on a per-user basis, as in *UserProfile*, allows to convey sufficiently regular and characterizing patterns to discriminate the destination station of a circulating bike, even for those stations that are less popular overall. On the contrary, station-specific profiles result in low predictive performances even when using mechanisms to prevent negative effects from class imbalance. In addition to that, the analysis has shown that the regular users amount to almost the totality of the bike pickups, at least for the Pisa BSS, hence confirming the viability of the approach to deploy an actual preview service.

8.3. Class imbalance

We can observe how the BSS problem is inherently affected by a strong class imbalance resulting from the fact that certain stations are more popular than others. At the same time, we have to note that class imbalance is not the key driver in determining the feature performance. On the one hand, such aspect does not affect the performance of personalized learning models as each of them is trained on a subset of the overall data where class imbalance has little impact. On the other hand, when using mechanisms for class imbalance reweighing on the *StationProfile* features we could not appreciate sufficient increases in their predictive performances when compared to a version of the same features obtained without such mechanisms.

8.4. Data preparation and profile consolidation

The analysis of predictive performance for users with a different number of pickups confirms the importance of working with consolidate users (i.e. characterized by a good number of pickups) to achieve an high overall accuracy. At the same time, the experimental results suggest that having a larger number of user personalized profile, even at the cost of including users with less consolidated usage data, might increase the accuracy of the service at the level of the single stations, typically the less popular ones. This provides clear indications that, at system deployment, it might be required to assess the trade-off between having an higher overall precision, which can be achieved by building personalized predictors only for very consolidated users, as compared to an higher precision on the single stations and possibly to a better customer experience for a larger

number of users. In this context, we can foresee that the final preview system will have to implement routines to periodically update the learning models as soon as more usage data becomes available for the corresponding users so as to optimize both the overall predictive performance as well as the accuracy on the single stations.

8.5. Deployment of a newly installed bike-sharing system

Another relevant aspect that needs considering is how to address the deployment of a newly installed bike-sharing system that initially has no historical data. The question of vendors and administrators is: after how many months of usage the collected data can make a predictive service accurate enough to be safely provided to the users? Unfortunately, in ML, the prediction performance is not necessarily proportional to the size of a dataset when this is collected over a relatively short period of time. For instance, it could happen that the behaviour of the BSS users shows a regularity in the first months, which is lost afterwards due to seasonality effects in the process being modeled. In our case study, the intuition that the larger the time-span considered for data collection the better the precision, was confirmed: we experimented with a ML model built using only two months of data for the training phase and the resulting predictive services were definitely inaccurate, e.g., with an F1-score of 0.45 for *UserProfile*. Hence, the optimal configurations of the product line of services for bike-sharing systems evolve over time as well: the idea is to gradually introduce more advanced prediction services as over time more data on the system in operation becomes available. The availability of a set of tools to support this evaluation will aid the maintainers of the bike-sharing system in dynamically choosing the optimal configuration during its life cycle.

8.6. Scalability

An aspect of interest for administrators of a BSS in a town of different size, that want to adopt a preview service, is scalability, the question being if it is feasible to adopt the proposed methodology in a small village or, more interestingly, in a metropolis. As an example Vélib′ in Paris has $20,000$ bikes, $1,800$ stations, and 225.000 users. The question is twofold: on the one side, if there is hardware scalability, on the other, adequacy of the results, will the same numerical results still hold?

Since the best performance was exhibited by *User Profile*, hardware scalability is not an issue: models of different users can be distributed on different machines, both at learning and at testing time. The second question is more complex and a definitive answer is difficult give: size is not the only variable, other aspects to consider are the morphology of the town (seaside or not, hilly or flat, ...), the presence of a university, the climate, etc. However, we can expect the existence of a consistent group of users with a well predictable usage behaviour to be an invariant and consequently, thanks again to the fact that the solution we suggest is to profile users, we can safely state that numerical results in different cities can be similar to ours.

9. Related works

The recently developed mean field model checker FlyFast [19, 20] has been used to assess performance and user satisfaction aspects of variants of large-scale bike-sharing systems. The combined use of these approaches is a preliminary step in the direction of automatic decision support for the initial design of a bike-sharing system as well as its successive adaptations and reconfigurations that considers both qualitative and performance aspects.

In [21], it has been studied the combination of novel spatio-temporal model-checking techniques, and of a recently developed model-based approach to the study of bike sharing systems, in order to detect, visualize and investigate potential problems with bike sharing system configurations. In particular the formation and dynamics of clusters of full stations is explored. Such clusters are likely to be related to the difficulties of users to find suitable parking places for their hired bikes and show up as surprisingly long cycling trips in the trip duration statistics of real bike sharing systems of both small and large cities. Spatio-temporal analysis of the pattern formation may help to explain the phenomenon and possibly lead to alternative bike repositioning strategies aiming at the reduction of the size of such clusters and improving the quality of service.

In [16, 17], the Pisa BSS case study was presented and defined a discrete feature model [22], specifying several kinds of nonfunctional quantitative properties and behavioral characteristics. In particular, a chain of tools was established, each of them used to model a different aspect of the system, from feature modeling to product derivation and from quantitative evaluation of the attributes of products to model checking value-passing modal specifications.

One of the main problem in BSSs is the distribution of bicycles in the various stations. Some stations tend to fill up, others to be often empty, and these situations dynamically vary during the day. The application of machine learning techniques to the forecasting of station occupancy in BSS has been explored by [1], where station-specific Naive Bayes models (i.e. similar in concept to the *StationProfile* aggregation) are use to predict bike availability based on current time, time frame for the prediction (i.e. in the next 10, 20, 30, 60, 90 and 120 minutes in the future) and the current proportion of occupied parking slots. In [23] the activity of the various stations is studied, and the stations are classified based on the occupancy during the day (free along the day, full in the evening, active overnight, etc.). Clustering algorithms (k-means, Expectation Maximization and sequential-Information Bottleneck) have been used and the results are compared and evaluated with different statistics. In [24], it is proposed an autoregressive model where the station-specific model also incorporates occupancy information from neighboring stations. The relationships with other stations in the BSS network has been further analyzed (by time-lagged cross-correlation) and incorporated into the predictions by [25] which also provided a first attempt to model time inhomogeneity in the probabilistic process regulating station occupancy. More refined time-inhomogeneous models have been explored by [26], based on Markov chains, and more recently by [27], using probabilistic queuing models for the single stations of the BSS. All

the approaches discussed above take a *StationProfile* approach, as they try to model a single station of the network, possibly incorporating information from neighboring stations. Further, the forecasting problem they address is either the classification of whether a station is almost empty or almost full, e.g. [1], or if there is at least one bike available in the station [27]. In this paper, on the other hand, we take a completely different approach which allows tackling a more complex and challenging problem. In fact, we provide forecasting models for the destination stations of all the circulating bikes, which allows to provide a complete and detailed picture of the number of bikes that will be available in each station within a short-time frame, rather than a fuzzy indication of wether there will be at least one bike or a space available.

In the product-line community, ML approaches have already been applied to feature models to predict the quality attributes, in particular performance, of the possible system variants [28–30]. The use of ML techniques is complementary to the approach taken here. We build ML models to implement the single features, which are prediction features, belonging to a small product line. They consider large product lines and predict quality attributes of the variants: the learning set is a set of couples $\langle$ configuration, performance$\rangle$, where a configuration corresponds to a set of features.

10. Conclusions

We have discussed how a machine learning approach can be used to both implement and assess predictive services for the users of a bike-sharing system (BSS): the services are concerned with the prediction of the destination station of a circulating bike, given information on its pickup details. We addressed the case study of the European project QUANTICOL[7], concerning the quantitative analysis of BSS.

The features required by the case study are paradigmatic of a class of learning tasks which require static learning models for vectorial data. Such models are trained on historical usage data to realize a deployable implementation of the preview service. In addition to that, a trained computational learning model is characterized by a measure of predictive performance that can be used to assess the cost-performance trade-off of the services before putting them into operation. In fact, a key aspect of ML models is the assessment of their predictive performance. As such, it can be used as part of the BSS design to straightforwardly assess the efficiency-cost trade-off of the features implemented by ML models using the performance measure.

We have trained and validated some learning models for implementing the *LocationPreview* services, using real-world usage data comprising more than 280.000 entries covering all hires in Pisa across two years. The models differ in the dataset used during the training phase and in the ML approach applied. They share the choice of relying on off-the-shelf ML tools.

[7] `http://www.quanticol.eu`

Note that BSS usage data has an inherent non-stationary and seasonal nature that must be taken into account when designing the system. Seasonality, for instance, manifests in, possibly sharp, changes in the typical pattern of system usage due to the presence of holidays or changes in the season (e.g. bikes tend to be used more in spring with respect to winter). We captured such seasonal patterns by learning models through an appropriate encoding of the bike usage time which explicitly models cyclic information such as weekday and holiday. Less regular forms of non-stationariety, e.g. a strike in the public transportation system, can be more difficultly captured by the model due to their episodic nature. However, this very same sparse-event nature maintains their impact on the overall performance of the system very limited.

A future development is to design a GPS based Preview service, assuming bikes can be equipped with a GPS. The service should predict the same output as *LocationPreview* using different input data, that are GPS trajectories corresponding to journeys performed by the BSS users. Trajectory data encode a form of dynamical information of different nature with respect to the static vectorial data in *LocationPreview*, requiring a radically different ML approach. A GPS trajectory is a form of sequential data, a type of structured information where the observation at a given point of the sequence is dependent on the context provided by the preceding or succeeding elements of the sequence. In this sense, it will be interesting to exploit efficient learning models specifically tailored to the predictive analysis of sequential data, such as Recurrent Neural Networks from the Reservoir Computing paradigm [31].

Acknowledgments

This research has been partly supported by the EU FP7-ICT FET-Proactive project QUANTICOL. We also thank *CicloPi* and Bicincittà for having made available the usage data.

References

[1] J. Froehlich, J. Neumann, N. Oliver, Sensing and predicting the pulse of the city through shared bicycling, in: Proceedings of the 21st International Jont Conference on Artifical Intelligence, IJCAI'09, Morgan Kaufmann Publishers Inc., 2009, pp. 1420–1426.

[2] J. S. Di Stefano, T. Menzies, Machine learning for software engineering: Case studies in software reuse, in: 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2002), 4-6 November 2002, Washington, DC, USA, IEEE Computer Society, 2002, pp. 246–251.

[3] D. Bacciu, S. Gnesi, L. Semini, Using a machine learning approach to implement and evaluate product line features, in: M. H. ter Beek, A. Lluch-Lafuente (Eds.), Proceedings 11th International Workshop on Automated

Specification and Verification of Web Systems, WWV 2015, Oslo, Norway, 23rd June 2015., Vol. 188 of EPTCS, 2015, pp. 75–83.

[4] L. Getoor, B. Taskar, Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning), The MIT Press, 2007.

[5] D. Bacciu, A. Micheli, A. Sperduti, Compositional generative mapping for tree-structured data - part i: bottom-up probabilistic modeling of trees, IEEE Transactions on Neural Networks and Learning Systems 23 (12) (2012) 1987–2002.

[6] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.

[7] T. K. Ho, The random subspace method for constructing decision forests, Pattern Analysis and Machine Intelligence, IEEE Transactions on 20 (8) (1998) 832–844.

[8] L. Breiman, Random forests, Machine learning 45 (1) (2001) 5–32.

[9] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT Press, Cambridge, MA, 1999, Ch. 11, pp. 169–184.

[10] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27, software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] R. Collobert, S. Bengio, SVMTorch: Support vector machines for large-scale regression problems, Journal of Machine Learning Research 1 (2001) 143–160.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[13] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, A. Talwalkar, MLlib: Machine learning in Apache Spark, Journal of Machine Learning Research 17 (2016) 1235–1241.

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

[15] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Vol. 14, Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1145.

[16] M. H. ter Beek, A. Fantechi, S. Gnesi, Challenges in Modelling and Analyzing Quantitative Aspects of Bike-Sharing Systems, in: T. Margaria, B. Steffen (Eds.), Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA'14), Vol. 8802 of LNCS, Springer, 2014, pp. 351–367.

[17] M. H. ter Beek, A. Fantechi, S. Gnesi, Applying the product lines paradigm to the quantitative analysis of collective adaptive systems, in: D. C. Schmidt (Ed.), Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015, ACM, 2015, pp. 321–326.

[18] M. ter Beek, A. Fantechi, S. Gnesi, L. Semini, Variability-based design of services for smart transportation systems, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, October 10-14, Proceedings, Part II, Vol. 9953 of LNCS, 2016, pp. 465–481.

[19] D. Latella, M. Loreti, M. Massink, On-the-fly Fast Mean-Field Model-Checking, in: M. Abadi, A. Lluch-Lafuente (Eds.), Trustworthy Global Computing, LNCS, Springer International Publishing, 2014, pp. 297–314.

[20] D. Latella, M. Loreti, M. Massink, On-the-fly fluid model checking via discrete time population models, in: M. Beltrán, W. J. Knottenbelt, J. T. Bradley (Eds.), Computer Performance Engineering - 12th European Workshop, EPEW 2015, Madrid, Spain, August 31 - September 1, 2015, Proceedings, Vol. 9272 of LNCS, Springer, 2015, pp. 193–207.

[21] V. Ciancia, D. Latella, M. Massink, R. Pakauskas, Exploring spatio-temporal properties of bike-sharing systems, in: 2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, Cambridge, MA, USA, September 21-25, 2015, pp. 74–79.

[22] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study, Tech. rep., Carnegie-Mellon University Software Engineering Institute (November 1990).

[23] P. Vogel, T. Greiser, D. C. Mattfeld, Understanding bike-sharing systems using data mining: Exploring activity patterns, Procedia - Social and Behavioral Sciences 20 (2011) 514 – 523.

[24] A. Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, R. Banchs, Urban Cycles and Mobility Patterns: Exploring and Predicting Trends in a Bicycle-based Public Transport System, Pervasive and Mobile Computing 6 (4) (2010) 455–466.

[25] J. W. Yoon, F. Pinelli, F. Calabrese, Cityride: A predictive bike sharing journey advisor, in: Mobile Data Management (MDM), 2012 IEEE 13th International Conference on, 2012, pp. 306–311.

[26] M. C. Guenther, J. T. Bradley, Journey data based arrival forecasting for bicycle hire schemes, in: Analytical and Stochastic Modeling Techniques and Applications, Springer, 2013, pp. 214–231.

[27] N. Gast, G. Massonnet, D. Reijsbergen, M. Tribastone, Probabilistic forecasts of bike-sharing systems for journey planning, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM '15, ACM, New York, NY, USA, 2015, pp. 703–712.

[28] A. Sarkar, J. Guo, N. Siegmund, S. Apel, K. Czarnecki, Cost-efficient sampling for performance prediction of configurable systems, in: M. B. Cohen, L. Grunske, M. Whalen (Eds.), 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015, IEEE, 2015, pp. 342–352.

[29] N. Siegmund, A. Grebhahn, S. Apel, C. Kästner, Performance-influence models for highly configurable systems, in: E. D. Nitto, M. Harman, P. Heymans (Eds.), Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015, ACM, 2015, pp. 284–294.

[30] J. Guo, K. Czarnecki, S. Apel, N. Siegmund, A. Wasowski, Variability-aware performance prediction: A statistical learning approach, in: E. Denney, T. Bultan, A. Zeller (Eds.), 2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013, IEEE, 2013, pp. 301–311.

[31] D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, A. Micheli, An experimental characterization of reservoir computing in ambient assisted living applications, Neural Computing and Applications 24 (6) (2014) 1451–1464.