POLITECNICO DI TORINO

Master degree course in Computer Engineering

Master Degree Thesis

Station-level forecasting in bike sharing systems using data mining techniques

Supervisors:
prof. Silvia Anna Chiusano

prof. Tania Cerquitelli

Candidate
Khalid Benihich

Academic year 2016-2017

Ai miei genitori, alle mie sorelle e alla mia famiglia tutta, grazie ai quali sono la persona che sono e ai quali posso dire solo un sincero grazie

Summary

With a growing levels of air pollution and the consequent climate changes, the development of new sustainable mobility strategies in urban areas has become a necessity. A green transport systems that are economical, healthy and less pollutant. Bicycle sharing systems are eco-friendly transportation systems that have found wide application in urban environments, for the so called last mile commute. This solution has shown a great success and a continuous growth all over the world.

The purpose of this thesis work is to mitigate this issue, Trying to answer a question of "*What are the stations that are going to be in a critical occupancy level in a future certain time*" by making predictions of the upcoming stations status using a classification model obtained by mean of predictive data-mining technique. Answering this question it's possible to create a tool, addressed to system managers, that make possible the re-balancing operations schedule anticipating eventual criticisms.

To achieve the goal three different prediction approaches has been assessed using real data of New York, one of the most populated city in the world. Then, It has been developed a modular structured framework that tracks the stations status and performs a prediction when requested. The framework is made available through a web-based application.

Finlay, and Android-based application has been realized to make the prediction tool more accessible and easy to use in a mobility environment.

Acknowledgements

I would like to thank my supervisors, Professor Silvia Anna Chiusano and Tania Cerquitelli, for their guidance, assistance and expertise through the realization of this project and the writing of this thesis.

I would like to thank Professor Paolo Garza for its technical support during the framework deployment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With a growing levels of air pollution and the consequent climate changes, the development of new sustainable mobility strategies in urban areas has become a necessity. A green transport systems that are economical, healthy and less pollutant. Bicycle sharing systems are eco-friendly transportation systems that have found wide application in urban environments, for the so called last mile commute. This solution has shown a great success and a continuous growth all over the world.

A bicycle-sharing system, public bicycle system, or bike-share scheme, is a service that allow users to rent bicycles for a limited period of time. Bike share schemes allow people to borrow a bike from a station equipped with a fixed number of docks and return it to any other station with available free docks. This allows each bike to serve several users per day limiting bike parks issues and making unnecessary a personal bike. Many bike-share systems offer subscriptions that make the use either free or very inexpensive, encouraging it's use as transportation. The success of this systems is principally due to its numerous benefits, that comprise:

- Environmental safeguar, beacause bicycle are eco-friendly ways of transport, which preserve the air quality of the city.
- Prevention of traffic congestion, because people who need to move for short trips (first mile/last mile connections) may avoid using cars or other pollutant means. A consequent parking demand reduction.
- People wellness, because riding a bicycle is a good way to maintain or improve their fitness.
- Other perks including mobility and access for residents improvement, greater connectivity between various transit modes, an affordable alternative to other transportation systems.

As in any system, advantages are accompanied by disadvantages and sub-optimal functioning. Often, typically during the peak times, stations close to aggregation points (e.g. universities, offices, etc.) fills up with the bikes coming from the suburbs, making difficult or even impossible to find free parking facilities to leave the shared bicycle. At the same time, stations in the outskirts get empty, and users who arrive later don't find bicycles to rent. This make necessary a repositioning operation of bicycles to re-balance the distribution over the city. These situations, in addition to being one of the most inconvenience and unpleasant situation that distance users from this mean of transport, they represent a relevant coast for the administrators of these systems. This issue moved a great interest and research effort devoted to optimizing service provision.

The purpose of this thesis work is to mitigate this issue, trying to answer a question of "*What are the stations that are going to be in a critical occupancy level in a future certain time*" by making predictions of the upcoming stations status based on the past and present data. Answering this question it's possible to create a tool, addressed to system managers, that make possible the re-balancing operations schedule anticipating eventual criticisms.

Nowadays the technological advancement in data collection and storage has given to companies the opportunity to store great quantities of data. Several bicycle sharing organizations have made their network data publicly available, making possible their collection and processing. A statistical analysis of the collected data contributes to leverage the development of new and innovative approaches to a better understanding of urban mobility, as well as of the use and performance of bike sharing systems. This analysis helps us to predict the future state of stations based on past data collected. However, the data collected on such systems are considerably large. It is, therefore, difficult to gain knowledge from them with classical analysis, but is necessary the help of advanced and automatic algorithms such as data mining. With them, it's possible to extract spatiotemporal patterns and give a synthetic and effective view of the extracted knowledge.

Predictive data mining automatically creates classification model from a prepared training dataset, and applies such model to automatically predict a class label to be applied on unclassified datasets. The model is trained based on past experience and apply knowledge gained to forecast upcoming situations and trends. There are many predictive data-mining techniques (regression, neural networks, decision tree, etc.) but in this work, only the classification models (Naive Bayesian) are covered.

In an attempt to handle the challenges above,this thesis has the following main contributions. A theoretical and analytical process has been proposed. A process composed of three phases describing the transformation of raw data in a suitable form. Then the enrichment with temporal information of different granularities and the windowing process to obtain training datasets are defined. Finally, the classification model learning procedure and its use to perform the prediction task are described.

Once defined the procedure to be adopted, we investigated three different possible approaches for the model creation. Respectively: a creation of a unique model used for all stations making up the network, a model for each station including information about the neighbors and finally a model for each station including only information about the itself. These three approaches are compared by mean of experimental analysis on real data and then evaluated the results in term of accuracy measures. In the experimental phase, models generated starting from real data have been analyzed. Two cities has been selected to conduct the experiments on real occupancy data. One is *New York*, a large and the most populous city in the United States (8.4 million inhabitants). The second one is *Turin*, a large Italian city (around 870.702 inhabitants). The New York bike sharing system scheme, CitiBike, is composed of 600 stations across Manhattan, Brooklyn, Queens and Jersey City with and 10,000 bikes. While the Turin one, [TO]Bike, is composed of 116 stations. For each city three datasets, two months sized and in different periods of the year, have been used for the creation of models based on the three approaches described above.

Once conducted the experimental phase and obtained the models, a software implementation of the prediction framework has been developed. The software made available as a web application has been developed adopting a modular structure that allows flexibility and possible future integration of new features. The framework implementation is composed of an interface element that receives prediction requests from users. By a *Manager* module that maintains the status information, and coordinates the other elements which are: the *Tracker* module who collect periodically stations occupancy level, and the *Data Transformer* which extracts the most recent gathered data and prepares the data sets for the prediction task. The last module is the *Predictor* which task to apply the station prediction model and forecast the upcoming status.

The final phase of this thesis work is the development of a mobile application. After a careful analysis of necessary requirements suggested by the information visualization literature and of which mobile platform is the right one. An Android mobile application has been developed giving some features like bike sharing system network map navigation, station current status visualization and upcoming criticality prediction. This represents a useful tool for the bike sharing systems operators responsible for re-balancing operations and future network improvements of the service.

The thesis is organized as follows, In Chapter 2 the relevant literature review is presented by introducing technical terms, algorithms and measurement techniques. In Chapter 3 the bike sharing is introduced highlighting the current issues. Then the methodological approach adopted for the data mining process is described. In Chapter 4 the experiments conducted with real data, in order to investigate three different approaches in the model creation are described, and the obtained results are exposed and analyzed. In Chapter 5 the actual implementation of the prediction framework as a web application is described. Finally, in Chapter 6 the mobile application developed to interact with the framework is described.

Chapter 2

Predictive Data Mining Algorithms

2.1 Introduction to Data Mining

The data mining discipline has as main goal the extraction of information and knowledge from vast amounts of data. The continuous progress in data collection and storage has lead companies and organization to store huge amount of data, in huge databases, called warehouse. As a consequence, the interest in data mining has grown significantly in both companies and in several scientific disciplines. In fact, this huge amount of data is difficult to manage using just the classical analysis techniques, data mining instead combines traditional techniques, such as sampling, estimation, testing with sophisticated search algorithms, pattern recognition, artificial intelligence theories, having as a goal the processing of the huge volume of data and the extraction of implicit informations, unknown previously.

The data mining is suitable for all the sectors where are available big quantity of data, that has to be processed and analyzed. In fact, the discovery of unknown patterns and the prediction of future trends are some of the salient aspects of this discipline, that attracted a great attention in several areas.

Data mining is an integral part of *knowledge discovery in databases (KDD)*, which is the global process of primary data into useful information. This process, as depicted in Figure 2.1, consists of a chain of transformation steps that has the data as input and the knowledge as output. The data mining block is preceded by a phase called data *preprocessing* and followed by the other one called *postprocessing*. In the first one data from different sources are fused, cleaned to remove noise and duplicates, and then relevant records and features are selected. This phase is typically the most laborious and time-consuming in the overall process, because of inhomogeneous ways in which data are collected and stored. The postprocessing purpose is to identify important and valid information extracted from data mining process and to make it available for further steps, fo instance visualization.

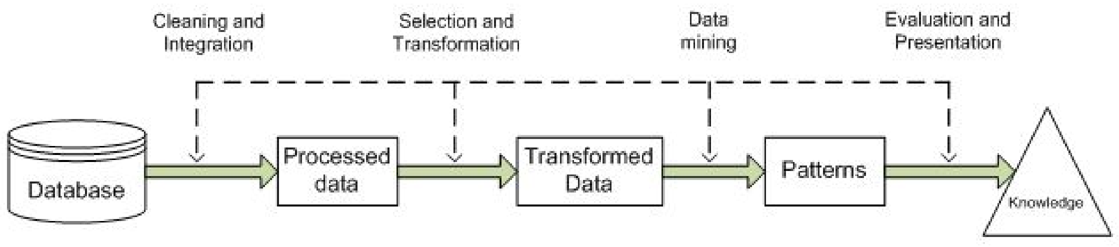


Figure 2.1: The knowledge discovery in databases process (KDD)

Data mining models are generally categorized into four main types: association rules, clustering, classification, and prediction (Fayyad and Piatetsky–Shapiro 1996). Association rules consist in discovering rules showing conditions for attribute values that usually occur together in a dataset. Clustering is the process of detaching a dataset into several different groups or clusters. Clustering objects goal consists in minimizing the inter-class similarity and maximizing the intra-class similarity. Classification approach develops a function or model that identifies the categorical class of an object based on its attributes. A classification model is generally constructed by analyzing the relationship between the attributes and the object classes in the training dataset followed by validation using testing dataset.

Prediction involves a model that infers a continuous value or future data trends. These models must accurately recognize particular data patterns that indicate upcoming data behavior. Prediction models may be based on classification or regression. In classification, the predicted variable is a binary or categorical variable. Some popular classification methods include decision trees, logistic regression and support vector machines. In regression, the predicted variable is a continuous variable. Some popular regression methods include linear regression, neural networks, and support vector machine regression (Kumar and Vijayalakshmi 2011).

2.2 Classification Tecniques

Among the data mining categories mentioned, one, in particular, is of interest in this thesis, and it's the classification. The general task of a classification model is the assignment of objects to one category among a set of predefined categories.

The input of a classification task is a collection of *records*, also called *instance* or *example*. A record is characterized by a tuple (x, y), where x is the attribute set and y is a special attribute, designated as the *class label*. The attribute set can contain discrete and continuous attributes. The class, on the other hand, must be a discrete attribute, and this is the main characteristic that distinguishes classification from the *regression*, a predictive modeling task in which y is a continuous attribute. A classification problem can be formalized as follows:

Definition 1. ***(Classification)*** *Given the training data* $\{(x_1, y_1), \ . \ , (x_n, y_n)\}$, *(where* x_i *are vectors). Classification is the task of learning a target function* $h : \boldsymbol{X} \to \boldsymbol{Y}$ *that maps* $x \in \boldsymbol{X}$ *to a class label* $y \in \boldsymbol{Y}$.

The target function is also informally called *classification model* and it's the term that will be used in the following discussion.

Classification models can be distinguished based on their purpose in two types:

2.2.1 Descriptive Modeling

A classification model can serve as an explanatory tool to distinguish between objects of different classes. For example, it would be useful to have a descriptive model that summarize the data of emails and explains what features defining if an e-mail contains spam or not.

2.2.2 Predictive Modeling

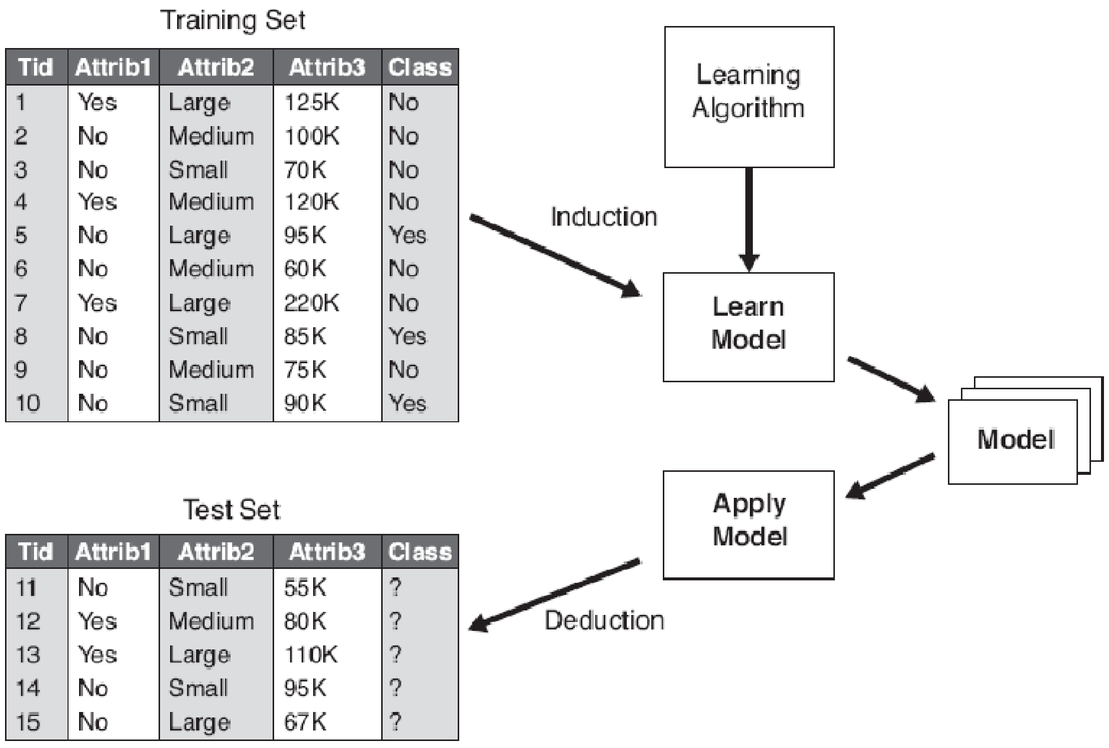
A classification model can also be used to predict the class label of unknown records. This kind of classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record. Classifications techniques are most suitable for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories.

The prediction of the particular attribute is based on the values of other attributes. The attribute to be predicted is commonly known as the *target* or *dependent variable*, while the attributes used for making the prediction are known as *explanatory* or *independent variables.* This classification model category is the one applicable to this thesis work. Because the main goal is a prediction of particular situations, which in general can be critical or not critical, then a representable by a binary category.

2.3 General approach to Solving a Classification Problem

A classification technique (or classifier) is a systematic approach to building classification models from an input data set. Some examples are decision tree classifiers, rule-based classifiers, neural networks, support vector machines and naive Bayes classifiers, this last category represents the base of classification technique adopted in this thesis work. Each technique employs a *learning algorithm* to identify a model that best fits the relationship between the attribute set and class label of the input data. The learning algorithm should produce a classification model that describe perfectly the input data and correctly predict the class label of never seen records. Hence, the main goal of the learning algorithm is to produce models with good generalization capability.

In Figure 2.2 is depicted the general approach for solving classification problems. A data set typically called *training set* consisting of records whose class labels are known must be provided. The training set is used to build a classification model, which is subsequently applied to the test set, which consists of records with unknown class labels. These two phases are respectively known as *training* and *testing*.



Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Figure 2.2: General approach for building a classification model [1]

2.4 Performance evaluation

The performance evaluation of an algorithm it's a crucial aspect in the model creation process. Not only to have an idea of how well the model could perform but also to compare different models, which is an important aspect of this thesis. In the following are described some metrics and techniques that are involved.

2.4.1 Confusion matrix

A classifier performance can be evaluated examining the number of records correctly predicted and incorrectly predicted. These values are represented in a table called *confusion matrix*. As it's observable in Table 2.1, it contains as many columns as the number of class labels and each entry f_{ij} represents the number of records belonging to a class i predicted as records of a class j.

Table 2.1: Confusion matrix for 2-class problem

		Predicted	
		Class=1	Class=0
Actual	Class=1	f_{11} (TP)	f_{10} (FN)
	Class=0	f_{01} (FP)	f_{00} (TN)

Although a confusion matrix provides the information needed to determine how well a classifier model performs, summarizing this information with a single value would make it more convenient to compare the performance of different models. This can be done using a performance metric such as accuracy, which is defined as follows:

$$Accuracy, a = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Equivalently, the performance of a model can be expressed in terms of its error rate, which is given by the following equation:

$$Error\ rate = \frac{Number\ of\ wrong\ predictions}{Total\ number\ of\ predictions} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Most classification algorithms seek a model that attain the highest accuracy, or equivalently, the lowest error rate when applied to the test set.

2.4.2 Class Imbalance

In many real applications, it's common to meet up with data sets characterized by an imbalanced class distributions. An example of this situation could be a system for credit card fraud detection, where fraudulent transactions are outnumbered by legitimate transactions. Despite their infrequent occurrences, a correct classification of the rare class in these applications often has greater value than a correct classification of the majority class. However, because the class distribution is imbalanced, this presents a number of problems to existing classification algorithms.

The accuracy measure, which is used extensively to compare the performance of classifiers, may not be well suited for evaluating models derived from imbalanced data sets. Since the accuracy measure treats every class as equally important. For binary classification, the rare class is often denoted as the *positive class*, while the majority class is denoted as *negative class*. For the evaluation of this kind of classifiers two more suitable metrics are usually adopted, and they are known as precision and recall.

2.4.3 Precision and Recall

Precision and Recall are two widely used metrics employed in applications where successful detection of one of the classes is considered more significant then detection of other classes. A formula of these metrics is given below.

$$Precision, p = \frac{TP}{TP + FP}$$

$$Recall, r = \frac{TP}{TP + FN}$$

Precision determines the fraction of records that actually turns out to be positive in the group the classifier has declared as a positive class. The higher the precision is, the lower the number of false positive errors committed by the classifier.

Recall measures the fraction of positive examples correctly predicted by the classifier. A classifier with large recall has very few positive examples misclassified as the negative class. In fact, the value of recall is equivalent to the true positive rate. It is often possible to construct baseline models that maximize one metric but not the other. For example, a model that declares every record to be a positive class will have a perfect recall, but very poor precision. Conversely, a model that assigns a positive class to every test record that matches one of the positive records in the training set has very hight precision, but low recall. Building a model that maximize both precision and recall is the key challenge of classification algorithms.

Precision and recall can be summarized into another metric known as the *F1 measure.*

$$F_1 = \frac{2rp}{r+p} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

In principle, F1 represents a harmonic mean between recall and precision. The harmonic mean of two numbers x and y tends to be closer to the smaller of the two numbers. Hence, a hight value of F1-misure ensures that both precision and recall are reasonably hight.

$$F_1 = \frac{2}{\frac{1}{r} + \frac{1}{p}}$$

2.4.4 Overfitting and Validation

The training phase is usually followed by another validation phase, useful for the best model choice and the performance estimation. The validation is also useful to mitigate the over-fitting problem, that arises when a model fit too well the training data but it doesn't generalize sufficiently. The over fitting can be caused by the lack of representative samples in the training data or by the presence of noise. Several validation techniques are available [2], some of the most commonly used are Holdout Method, Random Subsampling, CrossValidation e Bootstrap.

2.5 Bayesian Classifiers

In many applications the relationship between the attribute set and the class variable is non-deterministic. In other words, the class of a test record cannot be predicted with certainty even though its attribute set is identical to some of the training examples. This situation may arise because of noisy data or the presence of certain confounding factors that affect classification but are not included in the analysis. In the case of a bike sharing system the prediction of a station status based on the system usage in different moments of the day, could be affected by uncertainties introduced by other factors like weather, traffic or other unpredictable events that may occur in a city environment. The Bayesian classifiers model a probabilistic relationship between the attribute set and the class attribute based on the Bayes theorem.

As easy to guess, the Bayesian classifiers use the *Bayes theorem* to calculate the posterior probability, estimating the necessary statistical data on the training set (supervised learning). Assuming to have a set of possible classes C and a series of attributes (or features) $A_1, A_2, ..., A_k$ used to discriminate between classes, and we indicate with lowercase letters $(c, a_1...)$ particular values assumed by the variables. The optimal classification will be the one, that given a set of values assumed by the attribute, the outcome probability for a certain class is the maximum:

$$\max Pr(C = c \mid A_1 = a_1,\ A_2 = a_2,\ ...\ ,\ A_k = a_k)$$

According to Bayes theorem, the probability is equal to:

$$\max \frac{Pr(A_1 = a_1,\ A_2 = a_2,\ ...\ ,\ A_k = a_k \mid C = c)}{Pr(A_1 = a_1,\ A_2 = a_2,\ ...\ ,\ A_k = a_k)} Pr(C = c)$$

The prior probability to have a given class c can be estimated easily on the training set (provided that its representative), while the equation denominator it's irrelevant, because it's the same for all classes of set C, since independent of them. Hence it's a matter of estimate the conditioned probability in the numerator for each attribute. When it's done, the result is an optimum Bayesian classifier, able to produce the absolute better classification. The problem is that this estimation is very complicated because of the attributes dependence, and also with few features is likely to be computationally unfeasible. To mitigate the problem it's possible to use a Naive Bayes Classifier which, as the name suggests it make a naive hypothesis on the features, that they are completely independent. If it's the case, we have that

$$Pr(A1 = a_1 \mid A_2 = a_2,\ ...\ ,\ A_k = a_k, C = c) = Pr(A_1 = a_1 \mid C = c)$$

and therefore that $Pr(A1 = a_1, A_2 = a_2,\ ...\ ,\ A_k = a_k \mid C = c)$ is equivalent to

$$Pr(A_1 = a_1 \mid C = c) Pr(A_2 = a_2 \mid C = c)\ ...\ Pr(A_k = a_k \mid C = c)$$

At this point it's possible to derive everything is needed from the training set, like as follows:

$$Pr(A_j = a_j \mid C = c) = \frac{count(A_j = a_j,\ C = c)}{count(C = c)}$$

through which are available all the data necessary to calculate the posterior probability $Pr(C = c \mid A_1 = a_1...)$ and then to choose the class with the highest probability. The naive classifiers are quite simply realizable and don't require a great computational load. The independence assumptions of features are unrealistic in the majority of cases, but if well trained these classifiers have good performance. To find a middle way between these two extremes (total dependence/ total independence) have been developed several algorithms called not so naive classifiers, which aim is to enhance the accuracy of NB by relaxing the attribute independence assumption.

Notwithstanding the fact that some violations of the attribute independence assumption do not matter (Domingos and Pazzani, 1996), it is clear that many do, and there is an increasing body of work developing techniques to retain NB's desirable simplicity and efficiency while alleviating the problems of the attribute independence assumption.

Of these techniques, two have demonstrated remarkable accuracy. Lazy Bayesian Rules (LBR) (Zheng and Webb, 2000) has demonstrated accuracy comparable to boosting decision trees (Zheng, Webb, and Ting, 1999) and Super-Parent TAN (SP-TAN) (Keogh and Pazzani, 1999), a variant of Tree Augmented Naive Bayes (TAN) (Friedman, Geiger, and Goldszmidt, 1997), has demonstrated accuracy comparable to LBR (Wang and Webb, 2002). However, these two techniques have high computational overheads, SP-TAN having high computational complexity at training time and LBR having high computational complexity at classification time. This reduces their usefulness as an alternative to NB. The Aggregating One-Dependence Estimators (AODE) [3] appears to deliver lower variance but higher bias than LBR, TAN, and SP-TAN.

Table 2.2: Computational complexity of some NB classification algorithms [3]

	Training		Classification	
Algorithm	Time	Space	Time	Space
NB	$O(nt)$	$O(knv)$	$O(kn)$	$O(knv)$
TAN	$O(tn^2 + kn^2v^2 + n^2 \log n)$	$O(k(nv)^2)$	$O(kn)$	$O(knv^2)$
SP-TAN	$O(tkn^3)$	$O(tn + k(nv)^2)$	$O(kn)$	$O(knv^2)$
LBR	$O(tn)$	$O(tn)$	$O(tkn^3)$	$O(tn)$
AODE	$O(tn^2)$	$O(k(nv)^2)$	$O(kn^2)$	$O(k(nv)^2)$

k is the number of classes.
n is the number of attributes.
v is the average number of values for an attribute.
t is the number of training examples.

2.6 The AODEsr Classifier

To perform the occupancy level prediction of the Bike Sharing System, the forecasting framework developed in this thesis integrates a state-of-the-art Bayesian classifier, called AODEsr [4]. This algorithm is a combination of AODE with the Subsumption Resolution (SR).

The Subsumption Resolution (SR), which was first presented in Zheng and Webb (2006) under the name *Lazy Elimination*. It's a type of semi-naive Bayesian operation that identifies pairs of attribute-values such that one is a generalization of the other and deletes the generalization. SR can be applied at either training time or classification time. This adjustment is theoretically correct and demonstrated experimentally that it can considerably improve both zero-one loss and RMSE.

Chapter 3

The Prediction Approach

3.1 Bike Sharing Systems

Bicycle sharing systems consist of a set of stations spread all over the urban area. Users rent the bicycle at any station and return them to any other station where free slots are available. Hence to guarantee the quality of the service, system managers are interested in constantly monitoring the number of occupied docks per station. These systems are becoming more and more widely used. A decade has passed since the world's first large-scale bike-sharing scheme deployment, the Velib in Paris, whose immediate success (20 million users in its first year) prompted cities across the world to make available their own copies. A decade later there are 1,000 of these schemes, from Milton Keynes to the medina in Marrakech (the first scheme in Africa), and more opening currently. Nowadays over 400 cities have operating bike sharing programs worldwide, including North and South America, Europe and Asia [14].

Figure 3.1 gives a quantitative global overview of the bike share systems phenomena. While Figure 3.2 depicts the last decades growing trend of this means of transportation, in terms of total systems and cities.
The success of this systems is principally for its numerous benefits, that comprise:

- Environmental safeguard, because the bicycle is eco-friendly ways of transport, which preserve the air quality of the city. Many bike-sharing programs take pride in their environmental contribution. Some examples: Montreal's Bixi proudly states that its program has saved over 3,000,000 pounds of greenhouse gases since inception in May 2009 (Bixi 2009a). Lyon states that its program, which launched in 2005, has saved the equivalent of 18,600,000 pounds of CO2 pollution from the atmosphere (Greater Lyon 2009).

- Prevention of traffic congestion, because people who need to move for short trips (first mile/last mile connection) may avoid using cars or other pollutant

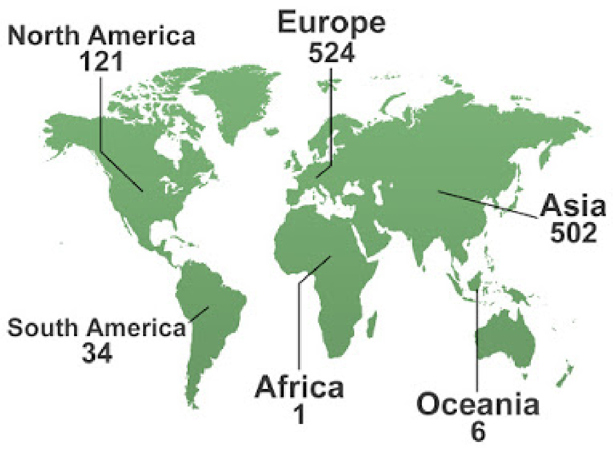


Figure 3.1: Public use bicycle programs by continent, year end 2016 [15]

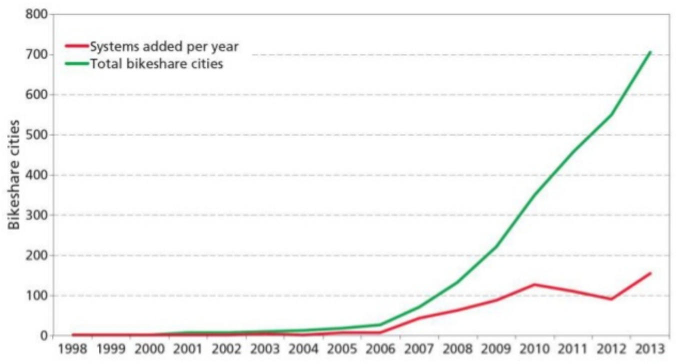


Figure 3.2: Bike sharing systems growing trend datasets [7]

means. Another consequence of congestion reduction, even small, is the evidence that even when controlling for the lost parking space (used to build the parking stations) the parking demand in the nearby areas was reduced by approximately 2% [8].

- People wellness, because riding a bicycle is a good way to maintain or improve their fitness.

- Other perks including mobility and access for residents improvement, greater connectivity between various transit modes, an affordable alternative to other transit systems.

3.2 The station occupancy prediction

As in any system, advantages are accompanied by disadvantages and sub-optimal functioning. Often, typically during the peak times, stations close to aggregation points (e.g. universities, offices, etc.) fills up with the bikes coming from the suburbs, making difficult or even impossible to find free parking facilities to leave the shared bicycle. At the same time, stations in the outskirts get empty, and users who arrive later don't find bicycles to rent. This makes necessary a repositioning operation of bicycles to re-balance the distribution over the city. These situations, in addition to being one of the most inconvenience and unpleasant situation that distance users from this mean of transport, it represents a relevant coast for the administrators of these systems. This issue moved a great interest and research effort devoted to optimizing service provision.

In bicycle sharing systems, a key performance indicator to monitor is the number of occupied docks (slots) per station, which corresponds to the number of parked bicycles. Monitoring the level of occupancy could highlight critical situations, which may lead to service disruption. This thesis work aim is to develop a data mining framework that analyzes the occupancy levels and consequently predict critical situations. In particular, has been analyzed the response of three models obtained by different approaches:

(i) A model for each station.

(ii) A model for each station and the neighbors.

(iii) A model for the whole network.

3.2.1 Station-level model

One approach is to get trained model of each station, trained using only the station data. The result of this case is a specific model for each station separately. One of the drawbacks of this approach is that to apply it to future unseen data it will have to consider a specific model for each station of the scheme available di in the dataset. Consequently, a training and testing phases are required to obtain each model and evaluate the quality. Furthermore, the data gathered from the system has to be differentiated in a way to extract a data set for each station. However, the expectation is to obtain models characterized by the best outcome in terms of accuracy, compared to the other two approaches.

3.2.2 Station-level with neighbors

This approach is a generalization of the preceding one. Also, in this case, the result of this is a specific model for each station separately, but considering the data describing the status of nearest stations. To define how a station is a neighbor or not has been defined a radius. This means that a new variable has been introduced in approaches evaluation. Hence as a different value is used a model has to be produced. The station neighborhood radius could be defined based on population density, as it could be a correlation element. But for this framework are been considered simply some different values, and then a comparison considering the model's accuracy has been done. Also for this approach, the same drawbacks of the preceding one are faced. Hence, training and testing phases are required to obtain each model. The difference, in this case, is that in both processes the data sets are not containing just the information of a single station. Therefore, each model has to be trained on a dataset containing not just records relating to the current station, but also about all stations inside the neighborhood around. Consequently, a similar process is required before a model application on unseen data. As one can easily guess, this approach a greater level of complexity is required. Nevertheless, it could be a good manner to improve the station-level model's accuracy. Because it considers partially the interactions, in terms of bikes exchange, that take place in small areas of the city.

3.2.3 Global scheme

In this case, instead of focusing on each station separately, are been considered all stations of the bike sharing scheme. This means that a data set containing records of all stations is used. Consequently, a unique model has been trained using data from all stations of the network, which are available in the data set. Respect to the two previous approaches, the model training, and the future application on never seen data have a smaller degree of complexity. In fact, the dataset does not require a split operation, in order to obtain a dataset for each station. Second, as the records are related to all stations in the scheme, a research of a station neighbors is not required.
This type of model has been evaluated and widely treated in literature, and has shown a quite good performance and accuracy. Therefore this approach has been considered for a comparison with the two station-level approaches. And then to define which is the best one.

3.3 The adopted procedure

In this thesis work, a theoretical procedure has been adopted. In a way to have a systematic approach and defined methods to be applied, in order to obtain the three different type of models, and after to have a base for the framework implementation. The main blocks of the procedure are:

(i) *Data collection and preprocessing*, where for each station the number of occupied docks is gathered at different sampling time instants, collected in a unique data repository and then prepared for the following data mining step.

(ii) *Data transformation*, where a class label is applied on the acquired records, defining if they represent a *Critical* or *Non-critical* situation, and then linked with related spatio–temporal data to generate a training dataset.

(iii) *Station criticality prediction*, where a classification model for each station, or for the global scheme, is learned from training data and then applied to forecast if status will be Critical or Non-critical in the near future.

Follows a more detailed description of each phase. Some parts of the process have been described based on previews work illustrated by Cagliero et al in [10].

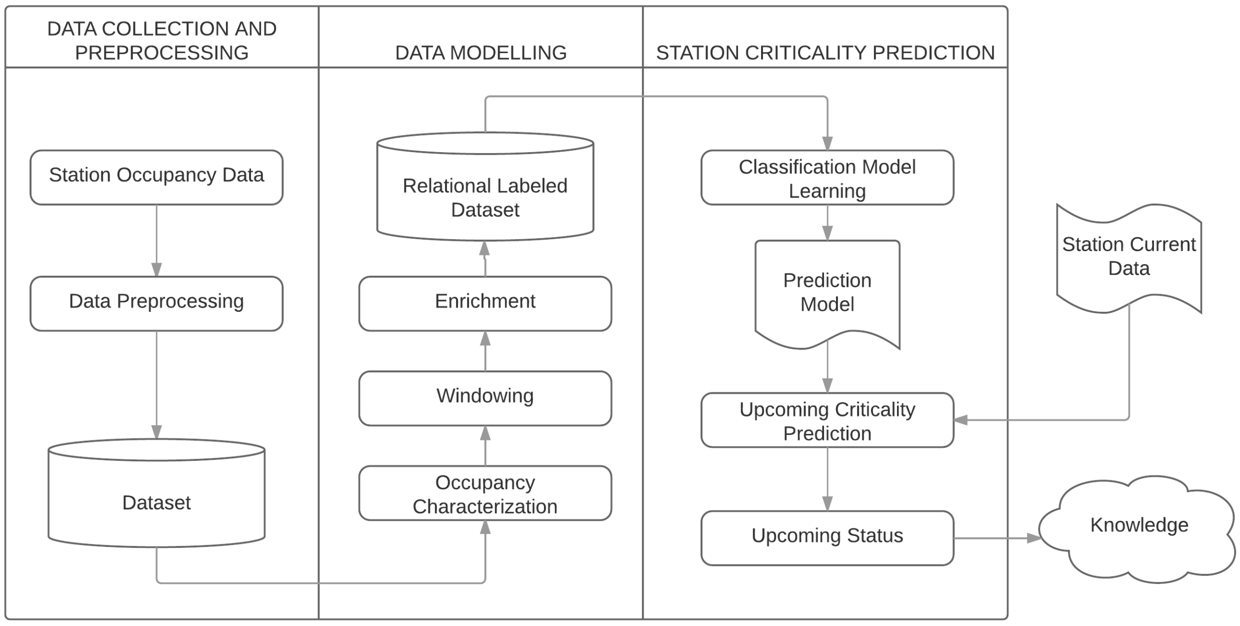

DATA COLLECTION AND PREPROCESSING
DATA MODELLING
STATION CRITICALITY PREDICTION
Station Occupancy Data
Data Preprocessing
Dataset
Relational Labeled Dataset
Enrichment
Windowing
Occupancy Characterization
Classification Model Learning
Prediction Model
Upcoming Criticality Prediction
Upcoming Status
Station Current Data
Knowledge

Figure 3.3: Schema of the theoretical procedure for the model generation and application

3.4 Data collection and preprocessing

Station occupancy data are gathered at a given sampling frequency for all the stations in the bicycle sharing network and then collected into a unique data repository.

This procedure can be formalized as follows. Considering S as the set of stations composing the bicycle sharing system and $c(s_j)$ as the capacity of stations $s_j \in S$, where the capacity is considered as the total number of docks available in the station. For each station $s_j \in S$, the procedure considers the following complementary data features.

(i) The *number of occupied docks* $o(s_j, t_i) \leq c(s_j)$ at timestamp t_i, which corresponds to the number of parked bicycles at a given instant of time and indicates the station level of usage.

(ii) The *time-stamp* t_i (time and date) at which the station data with the occupancy level $o(s_j, t_i)$ has been acquired.

The number of occupied docks $o(s_j, t_i)$ is an interesting feature because low values of parked bicycles at the station represent an alarm bell, that indicates a potential disruption. Hence, a re-balance operation could be scheduled by the service manager.
While monitoring the number of occupied slots, a few readings could be missed to tackle this issue, various approaches have been proposed to replace missing samples with reliable estimates (e.g. average, the mode)[9]. In this thesis, the missing values occurring in the acquired data are substituted with average measurement reading over all the sampling time instants.

3.5 Data transformation

In this phase, the preparation of data to the next classification process is arranged. Each sampled measurement is labeled with a class label (critical or non-critical). Then a *windowing technique* is applied to the acquired measurements according to the corresponding timestamps. Then the resulting dataset is transformed into a relational data representation to support the subsequent classification step.

3.5.1 Critical occupancy categorization

The measures on station occupancy acquired at given time instant (time-stamp) are classified as the corresponding occupancy level, according to an occupancy threshold *thr*. More specifically, the number of occupied slots per station can be labeled as follows:

(i) *Critical*, if the percentage of occupied docks in the station is below a given (analyst-provided) occupancy level threshold *thr*.

(ii) *Non-critical*, if the percentage of occupied docks in the station is above or equal *thr*.

3.5.2 Data windowing

Considering a station $s_j \in S$ in the bike sharing system. Is defined as *prediction time* t_p the time instant at which the prediction is triggered, for station s_j, the occupancy level (critical or non-critical) at a subsequent instant $t_f(t_f > t_p)$. The time gap $\gamma = |t_f - t_p|$ defines the *prediction horizon*.

In a future occupancy prediction, the values of occupied docks in the past are a key element. Because the number of occupied docks per station is likely to be temporarily correlated with each other. Consequently, for each station s_j and prediction time t_p is consider not only the current number occupied docks but also a snapshot of the occupation data samples at the previous instants. Doing this the temporal evolution of occupancy is considered while predicting the future occupancy level of a station. This concept could be described more formally as follow: given a station s_j and a prediction time instant t_p, is defined a time window $W(s_j, t_p)$ including an ordered sequence of WL timestamps. This sequence includes the prediction time instant t_p and $WL-1$ timestamps preceding t_p, i.e., $t_{p-1}, t_{p-2}, ..., t_{p-WL-1}$. For each time-stamp t_k in window $W(s_j, t_p)$m the station occupancy measurement $o(s_j, t_k)$ is considered. The measurements inside the time window are to be considered uniformly distributed. Hence, $\forall t_i, t_{i-1} \in W$ the sampling interval $\mathit{\Delta} = t_i - t_{i-1}$ is fixed. Adopting the windowing approach to consider the time evolutions of station occupancies, two parameters are to be defined:

(i) The *time window size* (WL), which is the attribute that defines the with of the learning time period. Based on the value assumed two different situations can occur.A too small window may bias the quality of the prediction because only the most recent occupancy measurements are taken into account. On the other hand, a too large window may include also past samples which are almost uncorrelated to the one measured in prediction time r_p. Therefore, medium sizes are preferable.

(ii) The *prediction horizon* (γ), which defines the temporal gap between the prediction instant time and the time instant at which the station occupancy level must be predicted. Small or medium values of γ are preferable. In fact, considering large y values the risk is to have low correlation with the past values, then a less accurate prediction.

3.5.3 Data enrichment

The life in a city typically follows a routine, but it varies based on various temporal factors. Fore instance, the people movement in a certain working day may differ from a weekend day, as it can differ from a month to another or from morning to midday ecc. To consider this kind of temporal information at different granularity levels during the occupancy level prediction and characterization, the time instant at which the prediction is triggered is enriched with additional information characterizing its temporal context.

To achieve this goal, a temporal hierarchy to aggregate date and time information into higher level groups has been created. Starting from the date, it has been generalized as the corresponding day of the week (1 to 7) and as the weekend day or working day. The time instead, has been generalized as hourly time slot, 2 hour time slots, and 4 hour time slot. Adopting this generalization with 4-h time slots, a day is partitioned into *morning* $([4\ am - 8\ am), [8\ am - 12\ am))$, *afternoon* $([12\ pm - 4\ pm), [4\ pm - 8\ pm))$ and *evening* $([8\ pm - 12\ pm), [12\ pm - 4\ am))$. Each of these time-zones is partitioned into shorted periods of 2h and 1h size. To make the concept easier to understand, in Table 3.1 has been described schematically the enrichment result starting from the time-stamp.

Table 3.1: Data enrichment example

time-stamp	
date	time

enrichement information				
day of week	weekend/weekday	4h-timeslot	2h-timeslot	1h-timeslot

3.5.4 Data representation

In order to obtain a training dataset for the subsequent phase of the procedure, it's necessary to define a suitable data representation. This representation has to describe, for each station of the scheme, the station occupancy level measurements correlated with the temporal information described in Sects. 3.5.1,3.5.2 and 3.5.3. To this aim, a relational dataset modeling has been adopted. A *relational dataset* is a set of *records*, each one is a set of items, where *item* is a $pair(attributename, value)$. While the *attribute name* is the description of the specific data feature, a value is the collected information. In the context analyzed in this thesis, each record corresponds to a different prediction time-stamp t_p for a station s_j in the scheme. Follows a formal description of the record structure.

Definition 2. ***(Station status relational record)*** *Let $s_j \in S$ be an arbitrary station and t_p the given prediction time-stamp. Let $W(s_j, t_p)$ be the time window of size WL for the pair of station s_j and time-stamp t_p). Let $t_p, t_{p-1}, ..., t_{p-WL-1}$ be the ordered sequence of timestamps contained in W. The record $r(t_p, s_j)$, corresponding to a prediction time-stamp t_p, is characterized by the following attributes:*

(i) *Prediction time-stamp t_p*

(ii) *Attributes describing the station occupancy, as a number of occupied docks, at timestamps $t_p, t_{p-1}, ..., t_{p-WL-1}$ corresponding to the sampling instants t_k in the time window $W(s_j, t_p)$.*

(iii) *Attributes describing the temporal context at timestamp $t_p + \lambda$.*

(iv) *Class, indicating if the station status in the near future, i.e., at time-stamp $t_p + \gamma$, is critical or not.*

Given this definition of what relational record represents in this work context. Starting from it are been defined two typologies of records, based on the presence or absence of the class attribute: a station record is a *training record* when the value of the class attribute is known. this first typology is the one that populates the *relational training dataset* used to learn the station model. Instead, any new station record for which the class at an upcoming time instant is unknown is considered as *test record.*

3.6 Station criticality prediction

The final phase of the theoretical procedure aim is twofold. In fact, it provides the following two subsequent steps:

(i) *Training phase.* For each station, a classification model is generated by applying a classification algorithm on a training dataset, which is a collection of historical measurements of station status, prepared in previews preprocessing phases.

(ii) *Prediction phase.* Once obtained the station classification model, it's possible to predict the upcoming station status (Critical or Non-critical label of class attribute) of new *test records.*

As anticipated in Chapter 2, the classification model can be potentially generated by many different techniques available in the literature (e.g. Decision trees, Support Vector Machines, Bayesian classifiers, associative classifiers). The framework process described so far can integrate many classification algorithms, but to perform the experiments and to compare the three different approaches described in Sections 3.2.1, 3.2.2, and 3.2.3 it necessary to define which classification algorithm is better to adopt. To take this decision an analysis of the current literature about the subject has been made. Especially to a paper of *Cagliero et al.* [10]. The study describes the experiments conducted with data acquired from a real bicycle system and comparing the most currently used state-of-the-art classification algorithms. The algorithms analyzed has been ranked considering two complementary aspects:

(i) The ability of the classifier to *accurately* predict the station status. Because the prediction must be as much accurate as possible to minimize the bias due to classification errors.

(ii) The *interpretability* of the generated model, which implies the readability by humans (domain experts).

The results demonstrated that the Averaged One-Dependence Estimators with Subsumption resolution (AODEsr), described in Section 2.6, performed best according to performance indices. Therefore it has been selected for this thesis framework.

Chapter 4

Experimental Validation

In this section, are have been conducted extensive experiments to evaluate different approaches, and to determine based on the effectiveness which should be integrated int the framework implementation. An introduction to the data and settings involved in the preparation process is given, then the model's evaluation is described, and finally the results obtained are discussed.

4.1 Data preparation

In this work two cities has been selected to conduct the experiments on real occupancy data. One is *New York*, a large and the most populous city in the United States (8.4 million inhabitants). The second one is *Turin*, a large Italian city (around 870.702 inhabitants). The New York bike sharing system scheme, CitiBike [16], is composed of 600 stations across Manhattan, Brooklyn, Queens and Jersey City with and 10,000 bikes. While the Turin one, [TO]Bike [17], is composed of 116 stations.

Considered the large amounts of data that has to be analyzed for a network like the New York one, it has been necessary a selection in order to reduce sizes. To achieve a size reduction that doesn't influence heavily the prediction models, the analysis has been conducted on the largest stations, which correspond to the mostly used ones. This choice has been taken because these stations are more likely to be in critical situations, hence are the most interesting for the analysis.

Since the use of a bike sharing system varies based on temporal factors. The data collected shall be enough to cover a long period of time. But if that period is too long the data amount increases consequently, threatening to make meaningless the previous selection of stations. Therefore, a trade-off is necessary. To conduct the experiments three different time periods of the year 2015 has been considered for the New York scheme:

(i) April - May

(ii) June - July

(iii) September - October

These periods not only contain information about different moments of the day, week and month but they also seasonal difference. Hence, they represent a quite good trade-off between representativeness and data size.

As described in the previous chapter, to collect the bike stations occupancy a measurement has to be done with a certain frequency. This frequency depends on the time period separating two consecutive measurements, that it has been called *Time Between Samples*. Before to start the data collection process this parameter value has to be defined. In the experiments done a 5 minutes period has been chosen.

At the end of data preparation phase, a dataset for each station and in each period considered has been obtained. The structure of these initial datasets is the one described in Table 4.1. Since each dataset is related to a single station some information like the name and location are redundant, and they are summarized by the *id* attribute.

Table 4.1: Raw data structure example for a single station

idx	ID	name	bikes	free	timestamp	lat	lng
1	79	W Broadway...	19	14	2015-04-17T13:30:24.129Z	40.71911552	-74.00666661
2	79	W Broadway...	21	12	2015-04-17T13:35:24.184Z	40.71911552	-74.00666661
3	79	W Broadway...	21	12	2015-04-17T13:40:24.210Z	40.71911552	-74.00666661

4.2 Data preprocessing

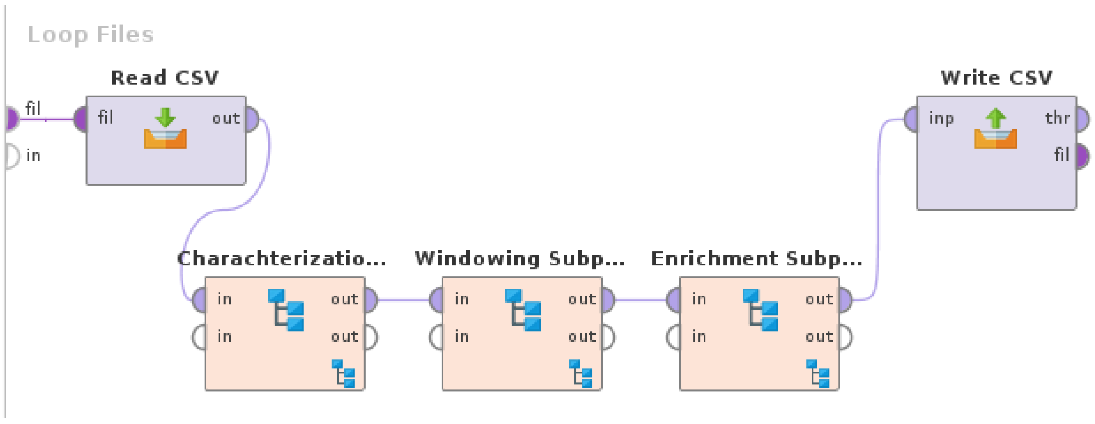
The raw data collected are not yet in the suitable form for a model training. It is indeed necessary to perform the step described formerly in Section 3.5, which are the characterization, the windowing and finally the enrichment.

The characterization consists of the application of a class label (Critical/Non-Critical) to the measurements based on the occupancy level. To do this, it has been necessary to define the *threshold* value below which the occupancy, in terms of occupied docks, is considered critical. For the experiments conducted this value was established at 10%.

The windowing task creates for each measurement a "window" containing the preceding values. Therefore, it has been defined a window of size $WL = 7$. Since the time period between samples has been fixed at 5 minutes, the window contains data relative to the preceding 40 minutes. The windowing task also assigns to each sample the class label that station assumes at an upcoming instant, which is constant and defined by the *prediction horizon* γ parameter. For that purpose, it has been selected a prediction horizon of one hour. In this way a time for an eventual rebalancing intervention is guaranteed and the correlation with station status in the past is maintained. The last step of this phase is the enrichment, which didn't need any particular parameter value definition.

In order to make faster and efficient the preprocessing, it has been created a *RapidMiner* process. Which as illustrated in Figure 4.1, it has been structured in three different subprocesses. Each one contains a definition of the corresponding task described above.

Figure 4.1: RapidMiner process for data transformation



The *characterization subprocess* (Figure 4.2(a)) is composed of two operators: the *Generate Attribute* that receives the dataset and adds a new categoric attribute. This new attribute represents the station criticality. Hence, the value is assigned according to the *bikes* attribute value. The *Set Role* instead assigns the *id* special role to the newly created class attribute, preparing the dataset to the following windowing subprocess.

The *windowing subprocess* depicted in Figure 4.2(c) uses a *Select Attribute* operator to get a set with the only *timestamp* and *available bikes* attributes, that is passed to the *Windowing* operator. The result is then united to the original dataset in order to have all the attributes by mean of a *Join* operator, which uses the measurement identifier as the key attribute.

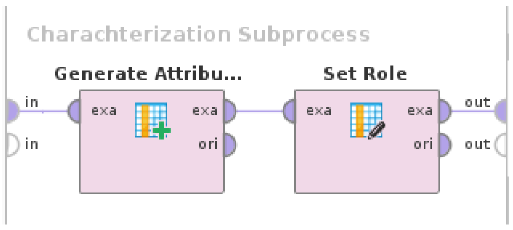
Finally, the *enrichment subprocess* (Figure 4.2(b)) extracts the temporal information starting from the timestamp again by mean of a *Generate Attribute* operator. In Table 4.2 are described the adopted generation functions.

Table 4.2: Attribute generation functions used for the Enrichment

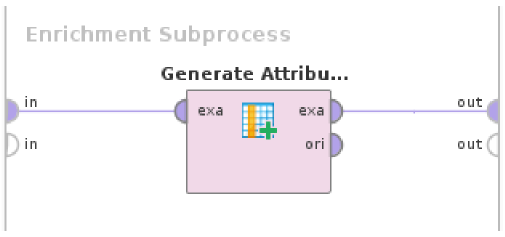
attribute name	**function expression**
timeslot_4h	*concat("4hslot", str(floor(date_get(timestamp, DATE_UNIT_HOUR)/4) + 1))*
timeslot_2h	*concat("2hslot", str(floor(date_get(timestamp, DATE_UNIT_HOUR)/2) + 1))*
month	*date_get(timestamp, DATE_UNIT_MONTH)+1*
hour	*date_get(timestamp, DATE_UNIT_HOUR)*
dayofweek	*parse(date_str_custom(timestamp, "u"))*
weekday_weekend	*if(date_str_custom(timestamp, "u")==7 || date_str_custom(timestamp, "u")==6, "weekend", "weekday")*

Figure 4.2: Subprocesses details.

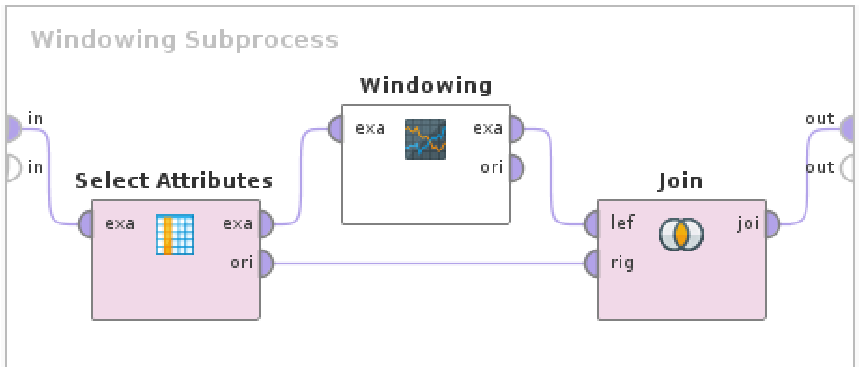
(a) Characterization subprocess content



(b) Enrichment subprocess content



(c) Windowing subprocess content



4.3 Models training

The previous steps have allowed obtaining a dataset for each station, in the three different two-month periods, ready for model training. Now, to obtain the models it has been created another RapidMiner process, which purpose is to apply the

AODEsr algorithm on a given dataset.

To train the models it has been decided to use, for each two-month period, the first month. While the second one for the testing purpose. Therefore, April data have been used to train per-station classification models and May to test them and measure performance, and so on for the other periods.

Before to start models generation, it needs to be underlined a difference between the three approaches that has to be investigated. In facts, while in the case of per-station model, the training datasets so far obtained are ready, for the approach considering data of a station and data of it's closer stations, a further step is needed. To obtain training datasets suitable for this approach, it has been developed a program. That for each station seeks the closer stations given a radius and then unifies these stations dataset with the current one data. This operation has been performed for each two-month period considered and using a radius of $2km$. A similar step has to be done for the last approach, unifying the data of all stations.

For all the three approaches, it has been used the same RapidMiner process (Figure 4.3), to train, test and measure performances.

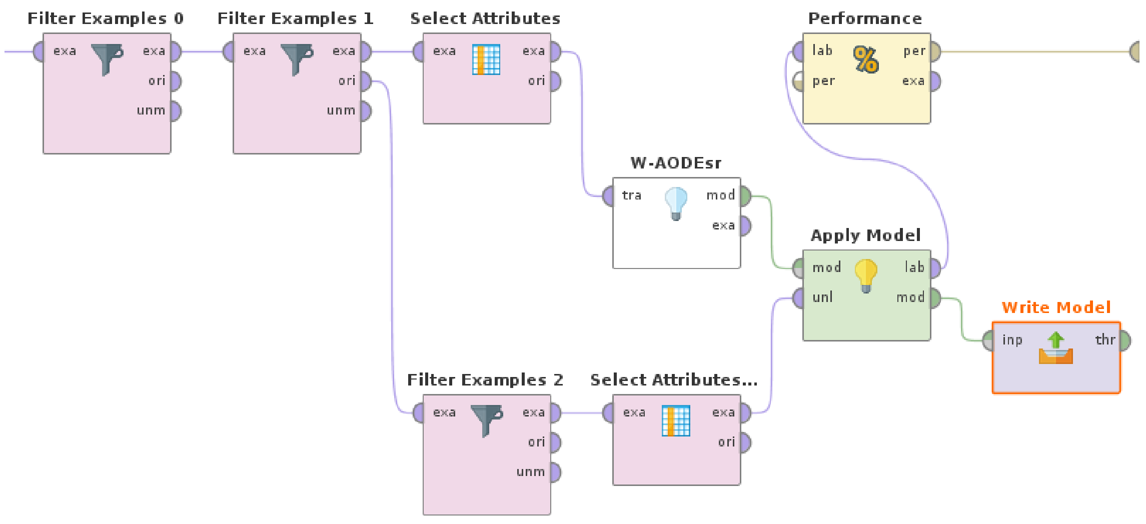


Figure 4.3: RapidMiner process for training testing and performance

The process receives a training dataset (containing two month period), and first of all, applies a *Filter Examples* operator that discards records having the *totalslots* attribute value equals to zero. Then other two filtering tools are used to split data. They select respectively, data of the first month for the model training and the second one for the following test phase. The classification model is generated by the mean of a W-AODEsr operator, which applies the AODEsr algorithm made available by the *Weka* [18] extension. The generated model is then applied to the testing data (the second month) using an *Apply Model* operator. The applied model quality is then measured through the *Performance* block and returned as output.

4.4 Models evaluation

To evaluate the results of the classification process have been considered established quality measures [1]. For each model, prediction quality has been assessed according to Precision, Recall and F1-measure. In the experiments conducted the statistics have evaluated considering the Critical class. Because in the analyzed context to plan re-balancing operations predicting critical situations is definitely more important than normal conditions.

$Recall(critical)$ gives an indication of how many critical situations (lake of parked bicycles) the model is able to predict compared with the total number of critical situation that actually occurred. And vice versa, $Precision(critical)$ gives the indication of critical situations that actually revealed themselves as critical. Therefor, the objective is to obtain a $Recall(critical)$ as much high as possible, but also a $Precision(critical)$ is fundamental because non-critical situations wrongly classified as critical could cause a waste of time and resources to the bike sharing system manager. The $F1 - measure$ represents a harmonic mean between recall and precision. The harmonic mean of two numbers x and y tends to be closer to the smaller of the two numbers. Hence, a hight value of $F1 - misure$ ensures that both precision and recall are reasonably hight.

4.5 Results

All the three model creation approaches investigated has achieved an appreciable performance in terms of Precision(critical), Recall(critical) and F1-measure (critical).

The approach that considers a single model for a station (trained using only the station data) reported a mean Precision(recall) varying between 73.17% of April-May period and 80.24% of September-October one. The Recall(recall), instead, reported a mean varying between 69.1% (the worst among al the three approaches) of April-May period and 80.24% of September-October. Another characteristic that could be noticed looking the bar chart (Figure 4.4) is a variation between the considered year periods. This approach, despite the acceptable values, has demonstrated to be the less accurate one.

The second approach, in which the per-station model is trained using data of station itself and of closer stations in a 2km ray, reported the best values for all periods and for both Precision(critical) and Recall(critical). The highest mean precision has been measured for the April-May period, with a value of 82,16%. While the lowest has been measured for the September-October period, where reached 79.45%. The F1-measure reported values around the 80%. The variation between two-month periods examined is less marked (Figure 4.5).

The last approach in which the model is generate based on data from all stations (Figure 4.6), reported lower Precision(critical) respect to the previous one but with higher Recall(critical) values. In fact, for the June-July time period, it has been registered the highest Recall(critical), with 84.55%. Given these results, it's possible to say that a good approach could be represented by a middle way between the two last one. This could be achieved using a per-station model but increasing the considered radius around the stations. This point can be a matter of future development of this work.

Figure 4.4: Single station model prediction quality

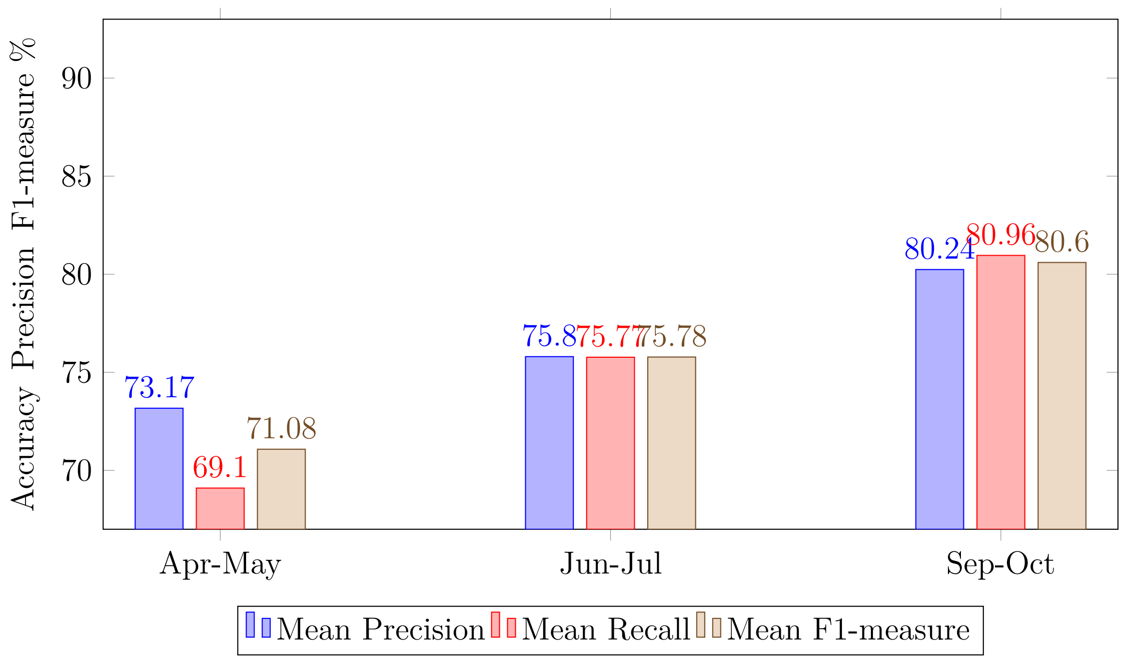


Figure 4.5: Single station with neighbors (2km ray) model prediction quality

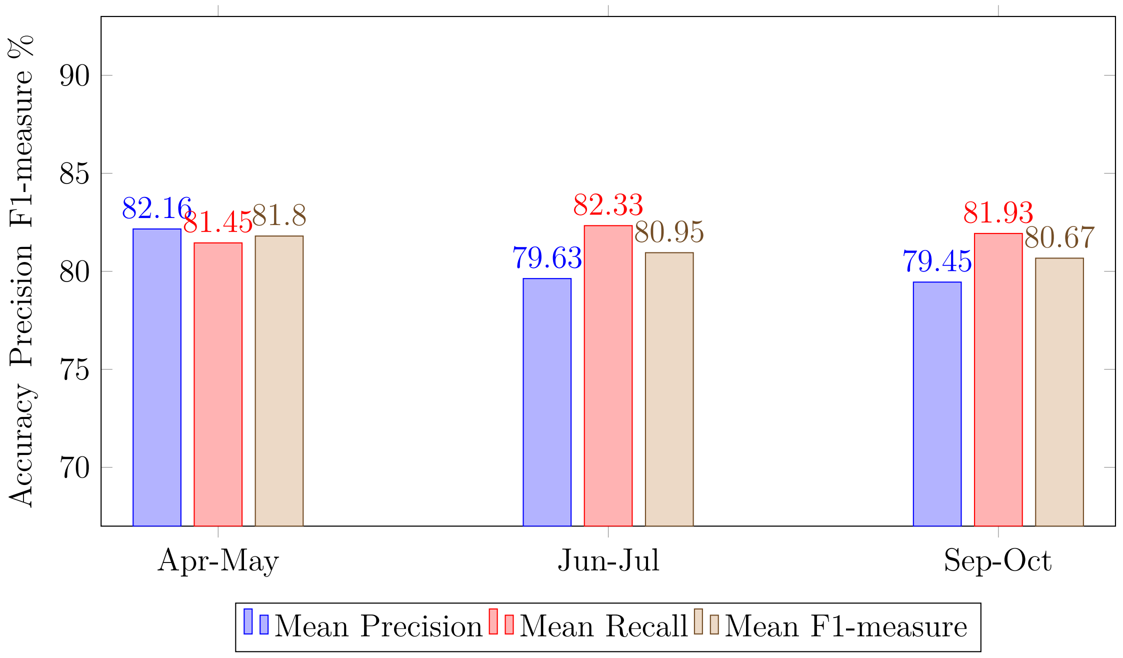
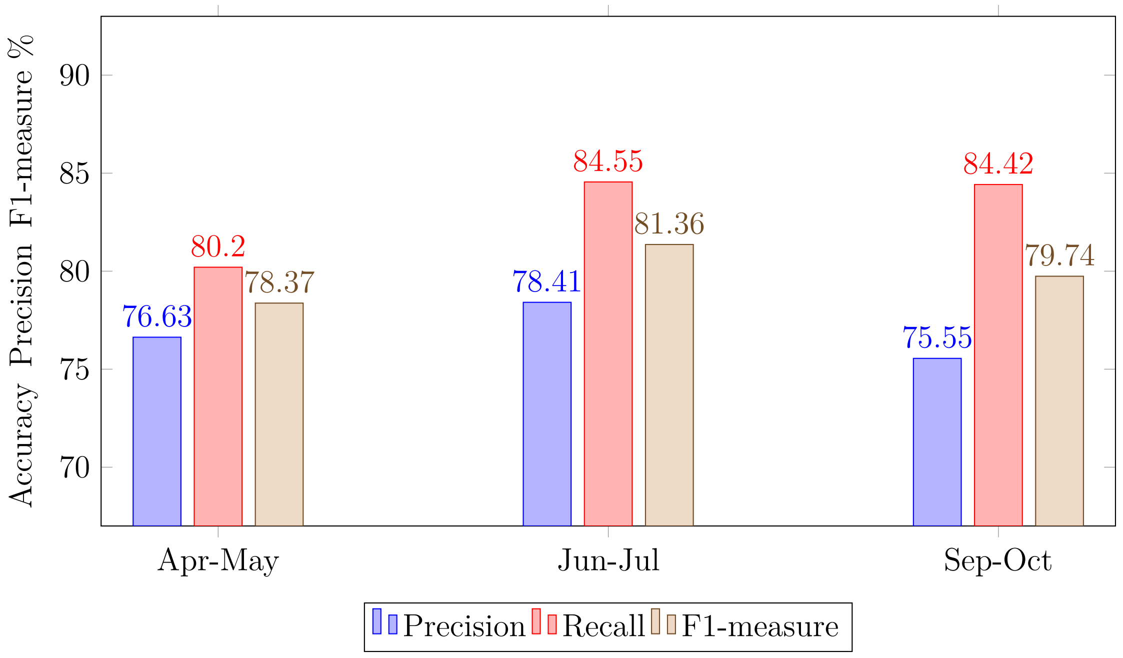


Figure 4.6: Global scheme model prediction quality



Chapter 5

The Prediction Framework Implementation

The goal of this chapter is to describe the structure of the prediction framework, developed in this thesis work, and some details about its components. Then are presented the technologies adopted for the actual implementation. To conclude, are presented some possible future developments and improvements.

5.1 The Framework implementation purpose

The idea behind this thesis project is to define a good model that is able to predict the future situations in a Bike Sharing System. Once obtained a model with acceptable results in terms of accuracies, it's time to deploy it in a real scenario and make it available to a potential final user, that could be a system manager or even the bikers using the system.

Facing this aspect has been realized that the model produced for a BSS scenario, could be extended to similar problems in a smart city environment. For instance, a similar approach could be used to predict the availability of places in a car parking, or electric car charging stations. Therefore, it has been adopted as a strategy the creation of a modular framework, which can be easily extended or adapted for similar situations.

5.2 Components

The preliminary framework design process,has lead to opt for a modular structure (Figure 5.1). This presents great advantages in term of flexibility and opens to future extensions and upgrades. The framework is composed of a set of modules, each of them has its own tasks. The following is a description of them, defining tasks performed and interactions among them and with the outside.

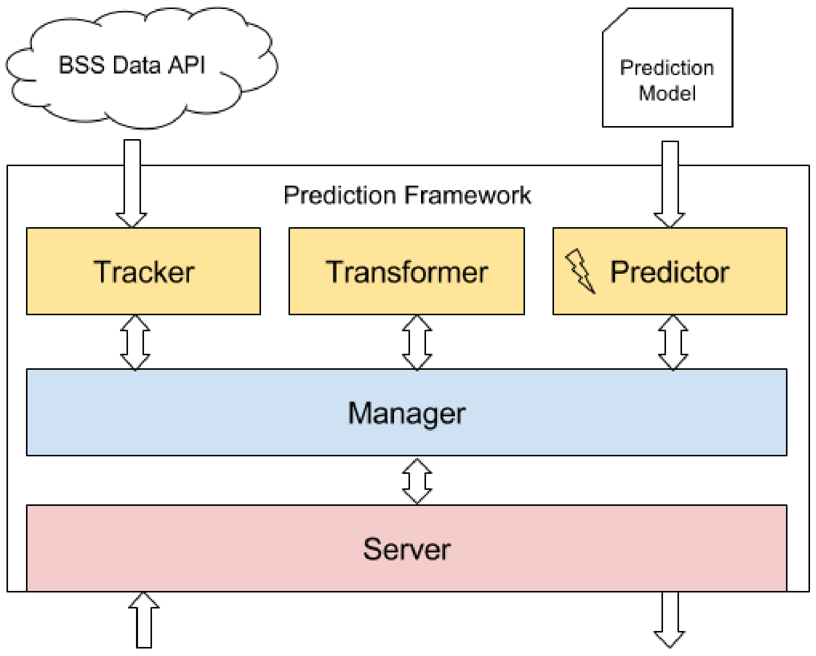


Figure 5.1: The prediction framework architecture

5.2.1 Server

One of the challenges, faced in the design process, was the definition of a suitable architecture. The choice affects all aspects of software design and engineering. An inappropriate architectural choice or a vitiated implementation would result in a bad response time. The choice also affects the development time, the future flexibility and maintenance. Considering all those factors, it has been decided to implement the framework based on two-tier client-server architecture.

As the framework has been intended to be used in a client-server architecture, it's necessary to have an interface able to perform the following task: receive requests, interpret them in a parsing process, and if well formed, send them to the Manager module in order to perform the requested operation. In the realize implementation are managed two type of requests:

(i) One that queries the prediction, specifying the station identifier and prediction horizon.

(ii) One that triggers the Tracker, specifying the delta time between samples (as described in previous chapters).

Once performed the requested task, the Server module reply with the result (expressed as a JSON or XML string) or eventually the error message in case of exception during the process.

5.2.2 Manager

It is the module that initializes and maintains the framework status information and parameters. It makes available the tasks that the Server module can execute. In our case, it starts and stop the Tracker module and triggers the Prediction one, but further tasks could be integrated if necessary.

5.2.3 Tracker

The framework to work needs the historical data, with the status in the past time. In this work, the BSS data are provided by a public API, and they have the form shown in Table 4.1. Through that API it's possible to obtain the current information about the occupancy of stations, but not in a past moment. The Tracker module aim is then to query the API with a certain frequency. This frequency depends on the temporal distance between samples which is given as input by the Manager. The gathered samples are then saved in a storage file for future elaboration.

From this dataset are extracted the samples belonging to a time window. The time window starts from the moment in which the prediction request has been received, and goes backward. The time window size is defined by an input parameter received by the Manager. Those samples correlated with the temporal data at the moment of prediction, are saved and will represent the unseen dataset that will be labeled by the Predictor.

The Tacker module is controlled by the Manager module, which is able to start and stop it when requested.

5.2.4 Transformer

This module handles the data preparation and transformation stage that precede the prediction. In particular, this module retrieves the latest dataset among all data collected by the Tracker module.On this dataset are then applied the required transformations (Characterization, Windowing, Enrichment). which had been previously described.

5.2.5 Predictor

It could be considered as the Framework's engine. In facts, it's the component that receives a data set containing the station status information, at the precise moment in which the prediction has been triggered, correlated with the station occupancy during a period of time preceding that moment (time window). On this dataset is then applied the prediction model given as a second parameter. This module has the particularity to be flexible in facts it performs de prediction with the given model, which can be a classification Naive Bayes based or other possible predictive algorithm.

5.3 Implementation

As has been anticipated, the framework has been designed to work in a client-server architecture. In this section will be discussed the implementation and technologies adopted.

5.3.1 Adopted server technology

The server application that implements the framework has been entirely developed in Java programming language, more precisely as a Java Enterprise Edition (J2EE) web application. A Java EE application is represented by a set of classes hosted on a remote server. A software called container runs on the server and monitors the entering connections. Each time a request is received the server instantiate one or more objects responsible for the elaboration and transmission of the generated response. There are different containers available on the market: for this project has been used Apache Tomcat. The container manages HTTP protocol and uses a standard template to describe web applications, which are distributed as .war archive.

Web applications are composed of different elementary components, and they are instantiated and controlled in their entire life cycle by the container in order to elaborate entering requests and produce responses to the user. Below are exposed the main components and their characteristics.

(i) Servlet: the most important components are servlets, Java classes responsible for the management of requests coming from the container, elaborate them and produce a response. The container creates a unique instance of servlet and calls its initialization method. At each request, the container invokes the servlet's service method using a different thread.

(ii) Filters: they are the Java components that transform the requests sent to a servlet or responses generated adding a functionalities layer which is dynamically inserted in the application's context. In this way it's possible to describe cross-application behaviors out of the application domain, defining functionalities common to multiple objects in only one spot. A classic example could be the authentication management, that without the use these components should be repeated in different points of the application. But in this way, it's not repeated and the other classes implement their functionalities only.

(iii) JSP Pages: other important components are the JSP pages, which are used to create web contents dynamically when responses are mostly textual because they make possible to use Java code inside an HTML structure. The JSP files are compiled and clients receive just the result of this operation.

(iv) Listener: the last main components of a Java EE application are the listeners, that are used for listening to events in a web container, such as a session creation.

Listeners are triggered for an actual physical request that can be attached to events in the application server. With listeners, it's possible to track application-level, session-level, life-cycle changes, attribute changes etc. In a servlet's life cycle, it's possible to monitor and react to events by defining listener objects whose methods are invoked when life-cycle events occur.

5.3.2 Representational State Transfer - REST

The internet evolution impose the realization of scalable and distributed systems, that are able to guarantee the interoperability among different platforms and terminals. In this thesis project, the system to be developed has to meet those requirements. For the project's main part the server offers a RESTful service with which the users are able to communicate. REST (Representational state transfer) is an open source architectural style for distributed hypermedia systems that can also be seen as a protocol for exchanging structured information in the implementation of web service. The concept was introduced in the year 2000 by Roy Fielding in his Ph.D. thesis [5]. In a REST architecture, data and functionality are seen as resources that are accessed through URIs. REST in the market component was used to interact with the resource in the database by specifying the location and the representation of the data. Most often the representations are HTML mainly for human consumption and JSON or XML for machines.

The HTTP protocol standard methods are mapped by REST on CRUD requests type: each request consist of an action (verb) and a URL (name), as shown in the Table 5.1. The web service implemented on this thesis uses POST requests to transfer messages between client ad server. As the search criteria of the services get pretty complex, to make things a little more convenient POST was the best choice to allow you to send more complex search criteria to the API. Working over HTTP, REST has stateless communication which means servers do not keep application states of the clients, instead clients should send all necessary state information in requests. The only state in the servers are the resources. Stateless nature is one of the main reasons under the scalability of Web. In simplest terms, RESTful Web services mean to apply REST design principles to develop Web services. REST is the underlying architecture style of the Web, so applying REST principles means direct integration to the Web. Rather than focusing on functions, RESTful Web services use Web resources as the main abstraction [6].

Table 5.1: HTTP verbs mapping according to REST principles

Operation	Method	Object
Create (single voice)	POST	URL of the collection
Read (voices set)	GET	URL of the collection
Read (single voice)	GET	URL of the element
Update (single voice)	PUT	URL of the element
Delete (single voice)	DELETE	URL of the element

5.3.3 Model application using RapidMiner

As it has been described before in the framework design process has been planned a module called Predictor, which should be able to perform the forecast requested by the user. Given the model previously produced through the process documented in Chapter 5, saved in the form of a file, and the unseen dataset to be labeled, it should proceed to perform the classification. In the implementation to perform this task, it has been used the RapidMiner suite [12].

In this thesis work, it has been created a RapidMiner (in Figure 5.2)process which goal is to perform the prediction task, and it has been exported to an XML file. In this way, the Prediction module Java class of the framework can use the RapidMiner Java libraries to interpret the process and run it when requested. The model application RapidMiner process has been constructed using the following operators:

(i) Read CSV: as the name implies, is the operator that extracts a dataset from a CSV file given as input and delivers the CSV file in tabular form along with the meta data. It's characterized by several parameters, but just csv_file is of interest. In fact the CSV file path is specified here and is modifiable by a Java class that implements the Predictor module.

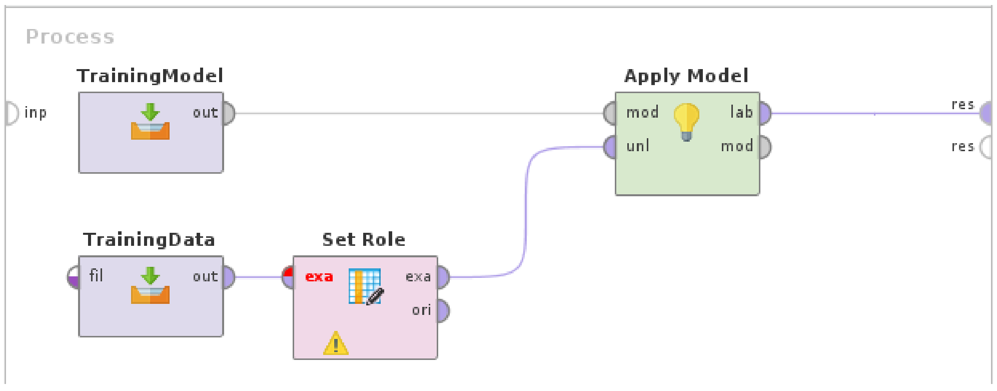


Figure 5.2: RapidMiner process that applies the model to unseen datasets

(ii) Set Role: the Role of an attribute reflects the part played by that attribute in an ExampleSet. Changing the role of an attribute may change the part played by that attribute in a process. One attribute can have exactly one role. This operator is used to change the role of one or more attributes of the input ExampleSet. This is a very simple operator, all you have to do is to select an attribute and select a new role for it. Different learning operators require attributes with different roles. This operator is frequently used to set the right roles for attributes before applying the desired operator. The change in role is only for the current process, i.e. the role of the attribute is not changed permanently in the ExampleSet.

Broadly roles are classified into two types i.e. regular and special. Regular attributes simply describe the examples. Regular attributes are usually used during learning processes. One ExampleSet can have numerous regular attributes. Special attributes are those which identify the examples separately. Special attributes have some specific task. Special roles are: label, id, prediction, cluster, weight, and batch. An ExampleSet can have numerous special attributes but one special role cannot be repeated. If one special role is assigned to more than one attribute in an ExampleSet, all these attributes will change their role to regular except the last one. The Set Role operator receives as input a data set and is essential that meta data should be attached with the those data for the input because the role of an attribute is specified in the meta data of the ExampleSet. The Retrieve operator provides meta data along with the data. The ExampleSet with modified role(s) is the output. This operator is characterized by the following parameters:

(a) name:The name of the attribute whose role should be changed is specified through this parameter.

(b) target_role The target role of the selected attribute is the new role assigned to it.

In our case this operator has been used to set the Label special role on the StationStatus attribute. An attribute with Label role acts as a target attribute for learning operators or others. Thanks to RapidMiner Java API it's possible to change this configuration on fly once loaded and interpreted the process file. This is useful in case of future changes in the dataset or model.

(iii) Apply Model: as the name itself suggests, it applies the model received as input to the dataset received as input too (unseen data). The models usually contain informations about the data on which has been performed the learning process, that will be used to forecast the class label (or the value in case of regression), to reproduce some transformations made during the learning and/or to execute other changes. In the moment that a model is applied, the number of attributes, their order, type and role played must be identical for the training dataset and for the unseen one. This operator owns two parameters only:

  (a) application parameters: is a list of parameters of the model, but it's not used typically;
  (b) create view: indicates that the model has to create a new view on the data where is possible, so that the results are calculated on the fly if necessary. It can assume a boolean value and it's false as default.

5.4 Improvements

After the implementation phase, a working web application running the prediction framework has been obtained. Despite a careful design process, a few critical issues have been noticed. The actual framework obtained presents some small robustness and prediction latency issues. While the first is simply solvable with tighter checks and best coding practices the second is more complex and critical from the final user point of view. The prediction latency is partly explained by two main factors:

(i) The data preparation phase, where the records belonging to the prediction time window are searched in the file containing al the records gathered by the Tracker module. This file size grow as new records are obtained from the CityBike API making the unseen dataset preparation latency always larger.

(ii) The second factor is due prediction model application and depends on the adopted algorithm, the number of variables and RapidMiner Java API.

While on the second is difficult to intervene, for the first one it has been proposed and implemented a solution based on a sliding windows algorithm. The idea is to spread the gathered records on different containers (in this framework have been used CSV files). The containers are associated with a time period that could be of different dimensions, in facts, it's possible ho have containers representing a day, or week, or month etc. The sliding window is intended to contain the last two containers, and it slides each time a new container is created. In this way, the Transformer module of the framework has to search in just the last two containers which represent the last period of time. A new container is created each time a new record belonging to a more recent time period (a new week or a new month etc.) is collected.

Chapter 6

Visualization and Mobile Application

In this chapter are explained the reasons and the importance of Information Visualization. Follows a descriptions of the choice maid in this thesis work to make available in the best way the results computed by the framework. It's also presented the mobile application that has been developed in this thesis, and the technologies adopted for it's actual implementation.

6.1 Information visualization

One of the challenges to deal with after a model creation and knowledge extraction is how to expose this last one. *Information visualization* is a scientific discipline that focuses on how visual data representations should assist people to understand better and gain insights into large and complex datasets. The main objective of Visual Analytics, is not just drawing nice graphics, but is to amplify cognition. Since it's inception,several definitions and descriptions of the field were given. One of the widely cited definition of information visualization is:

> "*Information visualization is the use of computer-supported, interactive, visual representations of abstract data to amplify cognition.*" [11]

The key elements of this definition that ar implied in this thesis final challenge are:

Computer-supported, the framework resulting knowledge is visualized by the mean of a computer or mobile device.

Interactive, the user should have the opportunity to interact the visualization elements. This element is represented by the possibility to navigate the map, and the selection of a particular station among the entire network, and once it's done to trigger the prediction process.

Visual representations of abstract data, the current station occupancy level is an abstract information that has to be graphically represented. This gives a more direct idea of the current situation. A second element to visualize are the time window samples considered in the prediction, being a bigger quantity of data, a visual representation may be more effective and direct.

Now that a correlation between the knowledge that has to be transmitted and the information visualization definition is verified. It's possible to follow the process modeled by the information visualization to obtain the views that framework user will see.

6.1.1 Visualization Reference Model

A model called the visualization reference model was suggested by [16] to convert raw data into a visual representation. Figure 6.1 illustrate the model and shows the major steps which are as follows:

(i) *Data Transformation*: The first step is to convert the raw information into a well-organized data format. In this step, missing values can be interpolated, the data can be filtered, or invalid information can be corrected. Additionally, data mining can be applied at this stage to extract meaningful information from the data collection. In the case of bike sharing system framework part of this process is performed on the server. Hence in this step current stations data (location, name and status) obtained by the API are organized.

(ii) *Visual Mapping*: The second step is mapping the dataset to a visual form. The stations current occupation is represented with circular rings and the numerical value. While the stations location is visualized on a map. The prediction information instead, are graphically shown with an histogram chart.

(iii) View Transformation: In the third step the visual forms are integrated into views which are shown on the screen and which can provide view transformations such as user navigation. After the raw data has been transformed into visual views, the user can interpret and reconstruct information by looking at the views.

6.1.2 Icon placement on map

This section deals with the problem of visualizing a large number of icons, each one representing a station, on the map. The visualization of many icons on the same screen often leads to cluttering issues, especially when users perform zoom-out operations and icons begin overlapping each other. Under high icon density

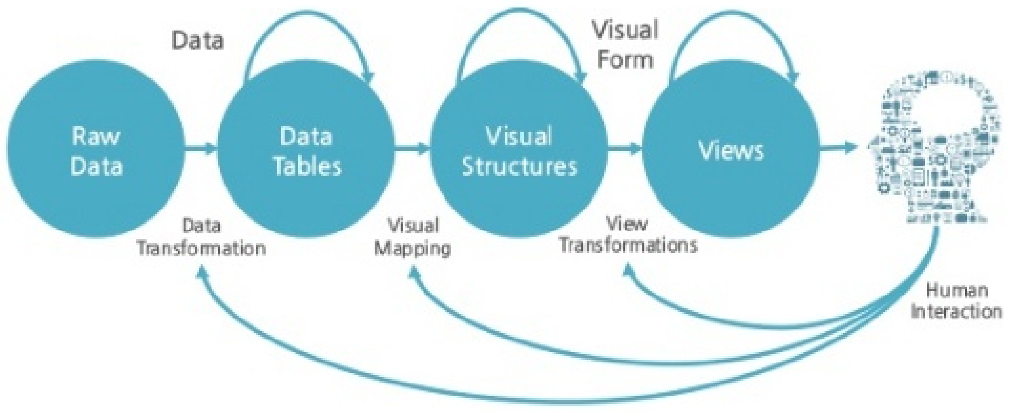


Figure 6.1: Data Visualization Reference Model pipeline

conditions, cluttering may degrade the effectiveness of even a close-up view of a map and it may severely mask other important map informations such as roads. Since user's requirements are not known a priori, one cannot pre-compute and store all possible maps that may be needed but must generate them in real-time.

Many solutions address problems pertaining to the design of icons or the encoding of values with icons but they are not applicable. However, a possible strategy to tackle the problem is to adapt methods and algorithms that have been developed for the closely related problem of label placement on maps. A solution is a techniques based on *aggregation* that aims at handling high icon density. The approach is based on applying an icon aggregation algorithm to reduce the total number of icons that need to be displayed and on designing icons that could function both as individuals and aggregates, in such a way users can convey critical information on the associated stations and possibly provide access to less critical information by tapping on them.

6.2 Mobile application

As already mentioned, for the actual implementation of the framework developed in this thesis a client-server architecture has been selected. In the previews Chapter 5 has been described the server side implementation, which role is to perform all the data acquisition, transformation, prediction model application and finally return the knowledge. Now it's necessary to discuss the second element of the architecture chosen, which is the client side.

Few decades ago client-server architecture was optimized to used with PC work stations and similar devices having large screen. Nowadays, the mobile revolution, and the technological level that has reached, pushes users to prefer mobile devices. This is happening thanks to the possibility offered by these devices to interact with services independently from time and location. This is a key aspect because final users of the framework are bike sharing system administration operators, which main task is to predict criticism and consequently schedule the re-balancing operations. This operation may be conducted while moving from a station to another, hence the mobility is a key aspect. Another advantage the could be considered is the affordable cost of mobile devices. Therefor a mobile application for smart-phones is more suitable then a PCs software for the client side implementation.

Another element to be considered are the limitations of portable devices, in terms of computational power and energy supply. These limitations, that are strictly correlated, are less restrictive. Thanks to the fact that operations requiring the majority of computational power are performed on the server side. While the principal task of client side, is the information visualization. A task that has less stringent requirements.

6.2.1 Mobile platform choice

Once defined the typology of client side, there was still a decision to be made, which mobile platform for a smart-phone is most suited. Three platforms were taken into consideration:

- Android
- IPhone OS (IOS)
- Windows Mobile

Windows Mobile had lots of different versions, which introduced compatibility issues. At the time, there are also rumors that the platform was dying, and Microsoft releasing a completely new platform - Windows Phone 10 - confirmed this not long ago, which is not going to be backward compatible at all. But despite this novelty, the market data of first period of 2016 confirmed the collapse trend. In march 2016 Microsoft has acquired Xamarin, a framework for cross-platform applications development. After this Xamarin has become officially Open Source and free for developers. In this work this technology has not been considered because of the lack of elements that can indicate a future success.

One of the main reasons not to consider developing for the iPhone is that the iPhone SDK did not support many features that our client application needed. Also at that time, Apple's developer's agreement did not allow any open source applications, which make it difficult to look for helpful material. Another limitation is that to develop the application is strictly necessary a Mac computer with the

XCode IDE. The development on other operative systems is discouraged by the incompatibilities. The last reason was Apple's tight control on what is released on their App Store, which sometimes takes even several weeks or months to get an application approved.

Therefor Android has been the final platform choice. There are several reasons that support this choice. One of them is the open nature of Android, being an open source project licensed mostly under Apache Software License and based upon the Linux kernel too. Other reasons are the large supply of materials and source examples and that publication of an application in the Android Market is easy and fast. Native Android applications, except some special cases, are developed using the Java programming language, and they can be easily ported to other mobile operating systems like Blackberry, Symbian and Ubuntu. In addition, Android apps can also be ported easily to Chrome OS. Market share, this is the number one reason that pushed for developing on Android first. According to IDC (International Data Corporation), Android absolutely dominated the market of smart-phones since today. To get an idea, in the second quarter of 2016 it was the first with 86.8% of market share.

Table 6.1: Worldwide smartphones OS market share (Source: IDC, Nov 2016)

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

6.2.2 Android application implementation

The application developed is composed by only two *activities* in order to make it simple and lighter as possible. The activities are called respectively *MainActivity* and *SettingsActivity*. The MainActivity is the first activity visualized just after a splash screen, and shows a map with the current user position obtained thanks to geolocation service.

The map, instead, is visualized by the means of a *MapFragment* which is a particular extension of a *Fragment* class specialized in map visualization and management. This fragment is the only element contained in the *MainActivity*, and inside it are performed all the operations. For the visualization of the map it has been adopted the *Google Maps SDK*, which is easily integrable in android applications and well supported.

Listing 6.1: A JSON response sample from CityBikes API

```
{
  "network": {
        "company": ["Comunicare S.r.l."],
        "href": "/v2/networks/to-bike",
        "id": "to-bike",
        "location": {
          "city": "Torino",
          "country": "IT",
          "latitude": 45.07098200000001,
          "longitude": 7.685676},
        "name": "[TO]BIKE",
        "stations": [
        {
          "empty_slots": 1,
          "extra": {"description": "Corso Castelfidardo 3
            9"},
          "free_bikes": 3,
          "id": "d5d84b0de5b967198d467baeb5b02c65",
          "latitude": 45.064101,
          "longitude": 7.659946,
          "name": "Politecnico 4",
          "timestamp": "2017-03-23T22:22:05.391000Z"
        }, ...
    ]
  }
}
```

Once loaded the application and the map is ready, an implemented method downloads the bike sharing system stations. To avoid the user interface slowdown that could cause usability issues, the operation is performed in a secondary thread. The thread sends a request to the same API service used to collect data on the framework server. The response obtained is a JSON containing an array of JSON Objects describing for each station the current status, location and name. The response, which resume is depicted in the Listing 6.1, is then parsed and each station data saved in objects called *Station*. For each of this stations, a google maps marker object at the corresponding coordinates (longitude and latitude) is created and visualized on the map. To avoid the map clutter issue described formerly it has been used the Google Maps MarkerClusterer library. In this way, when the user zooms out markers are clustered and the geographical information remains visible. The resulting view is depicted in Figures 6.2(a) and 6.2(b). The view contains also panel on the bottom, which goes up when a station is selected on the map. This panel shows the station name and the current status in terms of free bikes and empty slots, both in graphical and numerical mode. To give a more complete information the last update date and time are also shown. The panel is depicted in Figure 6.2(c), and as can be seen, a forecast button is shown. When pushed an http request indicating the station id is sent to the prediction framework server. As soon as the response is received, a string indicating if the station will be in a critical state or not and the samples considered for the prediction are shown, as in Figure 6.2(d).

6.2.3 Current application requirements

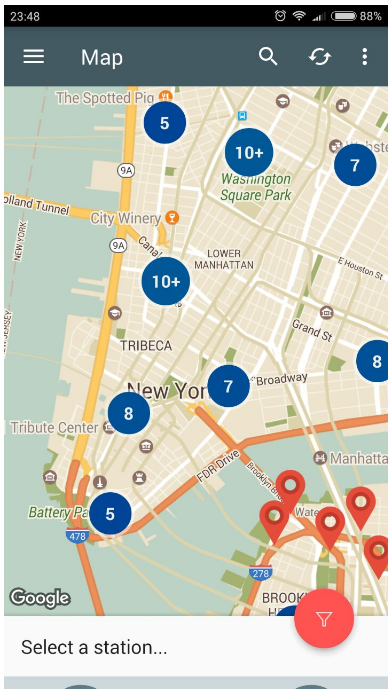
There are a few hardware and software requirements that has to be considered in order to successfully install and use the prediction application developed. The most important one is obviously that the mobile device is running the Android operating system. The minimum version of the operating system installed should be Android 4.0.3 (Ice Cream Sandwich), which has been released on October 19, 2011. As Android 6.0 is the most commonly installed version as of February 2017, this requirement should not cause any difficulties.

Platform	Codename	API	Distribution
Android 2.3.x	Gingerbread	10	1.0%
Android 4.0.3 Android 4.0.4	Ice Cream Sandwich	15	1.0%
Android 4.1.x	Jelly Bean	16	3.7%
Android 4.2.x		17	5.4%
Android 4.3		18	1.5%
Android 4.4	KitKat	19	20.8%
Android 5.0	Lollipop	21	9.4%
Android 5.1		22	23.1%
Android 6.0	Marshmallow	23	31.3%
Android 7.0	Nougat	24	2.4%
Android 7.1		25	0.4%

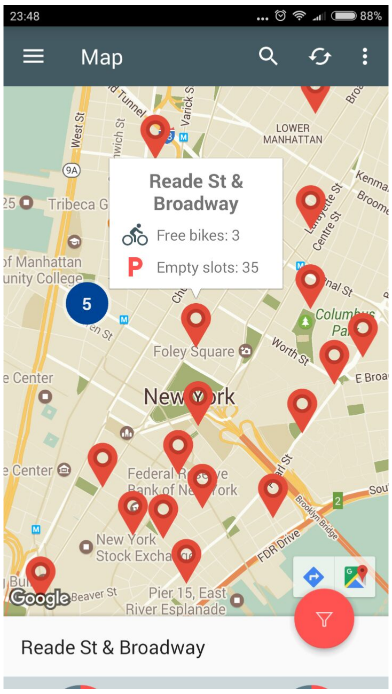
Table 6.2: Global Android version distribution as of February 2017.

An important requirement is that the mobile device should have GPS functionalities so that the application can determine the user's exact location. If this technology is not available in the device, the location could still be determined using the Wi-Fi or network connections, but a GPS receiver is capable of providing more accurate estimates.

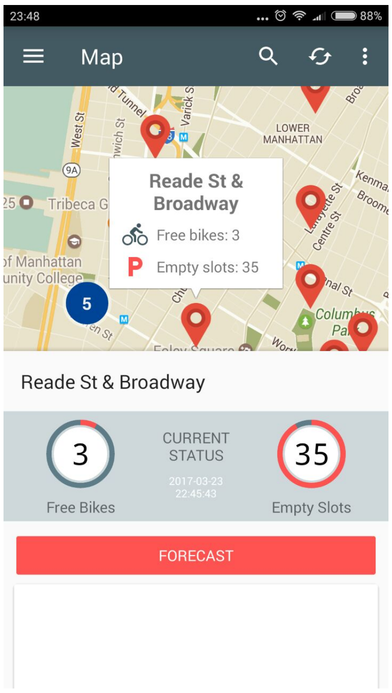
Lastly the mobile device should have an internet connection available at all times. The recommended one is a 3G connection such as UMTS or EDGE, as these have the highest data rates (384 Kbit/sec or more). Others, such as GPRS (144 Kbit/sec), might suffice for the current use as well, as the internet connection is solely being used to download JSON and XML data.



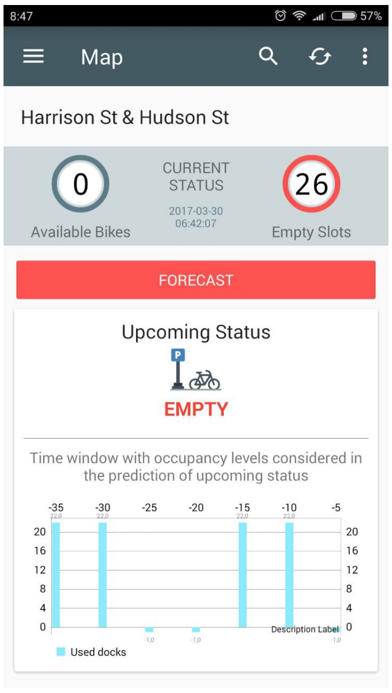
(a) Map with clustered markers



(b) Station selected



(c) Station current status



(d) Prediction result

Figure 6.2: Android application main Activity

Chapter 7

Conclusion

In urban areas, bicycle sharing systems are eco-friendly transportation systems that are becoming more and more popular and widely used. With the help of this work, we tried to bring the Information System, in particular, the Data Mining technology, to an upcoming growing area of sustainable mobility in smart cities. The main contributions and developments of this thesis were threefold.

The fist was the formalization of a process for automatically forecasting critical situations, due to low occupancy levels, in a bike sharing scheme context. Especially, considering predictive data mining techniques based on classification.

The second was an analysis of three different approaches used to generate classification models. Respectively: (i) a creation of unique model used for all stations making up the network, (ii) a model for each station including information about the neighbors (within a radius of 2 km) and finally (iii) a model for each station including only information about itself. By conducting experiments on real data adopting a Naive Bayes algorithm (AODEsr), we proposed that a tradeoff between the model based on the overall network, considering all stations together and the one based on each station and the neighbors can lead to a more precise prediction.

Lastly, a software implementation of the prediction framework has been developed. The software made available as a web application has been developed adopting a modular structure that allows flexibility and possible future integration of new features. The framework implementation is composed of an interface element that receives prediction requests from users. And inside by a set of modules that carry out the wall predictive data mining process. In order to make the prediction framework easily accessible in a mobility context, an application mobile devices has then developed.

The work done in this thesis can provide useful information to improve the quality of provided services and fulfill the needs of BSSs users. Future developments of this work, can address the classification problem adopting alternative algorithms and verify the model's creation approaches analyzed here on different schemes and cities. Moreover, the impact on prediction quality can be investigated considering spatial knowledge, like the weather (temperature, precipitations, etc.) or traffic. Another possible future work can be the extension and improvement of the produced framework implementation, in order to make it a more general tool that can be adopted by other services in a smart city environment (e.g. power consumption, public transit systems).

Bibliography

[1] P.Tan, M.Steinbach, and V.Kumar. “Introduction to Data Mining”. Addison Wesley, 2005.

[2] R. Kohavi et al. “A study of cross-validation and bootstrap for accuracy estimation and model selection“, in IJCAI, vol. 14, pp. 1137–1145, 1995.

[3] G.I. Webb, J. Boughton, Z. Wang “Not So Naive Bayes: Aggregating One-Dependence Estimators”, in Mach Learn, vol. 58, pp. 5-24, 2005.

[4] F. Zheng, G.I. Webb, P. Suraweera, L. Zhu “Subsumption resolution: an efficient and effective technique for semi-naive Bayesian learning”

[5] Roy Thomas. “Architectural Styles and the Design of Network-based Software Architectures”. diploma thesis, University of California, 2000.

[6] Sam Ruby Leonard Richardson. “RESTful Web Services”. O’Reilly Media, 2007.

[7] Elliot Fishman, “Bikeshare: A Review of Recent Literature”. Transport Reviews, volume 36, pages 92-113, 2016.

[8] Pelechrinis, Konstantinos and Li, Beibei and Qian, “Sean, Bike Sharing and Car Trips in the City: The Case of Healthy Ride Pittsburgh”. 2016.

[9] Le Gruenwald, Hanqing Yang, Md. Shiblee Sadik, Rahul Shukla. “Using data mining to handle missing data in multi-hop sensor network applications”, MobiDE ’10 , Pages: 9-16, 2010.

[10] L. Cagliero, T. Cerquitelli, S. Chiusano, P. Garza, X. Xiao. “Prediction critical conditions in bicycle sharing systems”. Computing, volume 99, pages 39–57, 2017.

[11] Card, K. S., Mackinlay, J. D., and Shneiderman, B. “Readings in Information Visualization, using vision to think". Morgan Kaufmann, Cal. USA. 1999.

[12] RapidMiner, `https://rapidminer.com/`

[13] FOSS, “Bike sharing networks arround the world”, CityBikes API, 2010, `https://citybik.es`

[14] The bike-sharing blog, `http://bike-sharing.blogspot.it/`

[15] The Bike-sharing World Map, `http://www.bikesharingmap.com/`

[16] CitiBike, `http://www.citibikenyc.com/`

[17] TO]BIKE, `http://www.tobike.com/`

[18] Weka 3: Data Mining Software, `http://www.cs.waikato.ac.nz/ml/weka/`