Universitat Politècnica de València
Departament de Sistemes Informàtics i Computació

Master Thesis

A recommendation framework based on automated ranking for selecting negotiation agents. Application to a water market

Submitted by: Encarna Maria Durà Garcia

Supervisors: María José Ramírez Quintana
Antonio Bella Sanjuán

València
September 26, 2011

Acknowledgements

First of all, I would like to especially thank María José Ramírez and Antonio Bella for all their advice, support and patience during these months.

Also I would like to truly appreciate the collaboration and dedication of Antonio Garrido and Adriana Giret, and the Agreement Technologies Project (Consolider Ingenio CSD2007-00022).

It is necessary to mention the helpful contribution of the members of DMIP meetings: José Hernández, Cèsar Ferri, Nando Martínez, Jorge Bedoya and Javier Insa. I also wish to remember and thank my colleagues of the laboratory 201 of DSIC, as well as the support of ELP Group and, in extension, everyone who has contributed in some way to this thesis.

Also I would like to mention the friends I have made during the years at ETSINF and at the master's degree. Particularly, my good friend Javi, who has always supported me as I support him.

Outside the academic environment, I would like to express my deep gratitude to my family, especially my parents.
And finally, to Víctor, for being so important and special to me.

Abstract

This thesis presents an approach which relies on automatic learning and data mining techniques in order to search the best group of items from a set, according to the behaviour observed in previous groups.

The approach is applied to a framework of a water market system, which aims to develop negotiation processes, where trading tables are built in order to trade water rights from users. Our task will focus on predicting which agents will show the most appropriate behaviour when are invited to participate in a trading table, with the purpose of achieving the most beneficial agreement.

This way, a model is developed and learns from past transactions occurred in the market. Then, when a new trading table is opened in order to trade a water right, the model predicts, taking into account the individual features of the trading table, the behaviour of all the agents that can be invited to join the negotiation, and thus, becoming potential buyers of the water right.

Once the model has made the predictions for a trading table, the agents are ranked according to their probability (which has been assigned by the model) of becoming a buyer in that negotiation. Two different methods are proposed in the thesis for dealing with the ranked participants. Depending on the method used, from this ranking we can select the desired number of participants for making the group, or choose only the top user of the list and rebuild the model adding some aggregate information in order to throw a more detailed prediction.

Resumen

Esta tesina presenta una aproximación basada en aprendizaje automático y minería de datos con el fin de buscar el mejor grupo de ítems de un conjunto, según el comportamiento observado en grupos previos.

La aproximación se aplica en un marco de un sistema de mercado de agua, el cual tiene como objetivo desarrollar procesos de negociación, donde se crean mesas de negociación para comerciar los derechos de agua de los usuarios. Nuestra tarea se centrará en predecir qué agentes mostrarán el comportamiento más apropiado cuando son invitados a participar en una mesa de negociación, con el propósito de alcanzar el acuerdo más beneficioso.

De este modo, se desarrolla un modelo que aprende de transacciones pasadas ocurridas en el mercado. Cuando se abre una nueva mesa de negociación para vender un derecho de agua, el modelo predice, teniendo en cuenta las características individuales de la mesa de negociación, el comportamiento de todos los agentes que pueden ser invitados a unirse a la negociación, y así, convertirse en potenciales compradores del derecho de agua.

Una vez el modelo ha hecho las predicciones para una mesa de negociación, los agentes son clasificados de acuerdo a su probabilidad (asignada por el modelo) de convertirse en comprador en dicha negociación. Dos métodos diferentes se proponen en esta tesina para tratar a los participantes clasificados. Dependiendo del método utilizado, de este ranking podemos seleccionar el número deseado de participantes para confeccionar el grupo, o elegir sólo el usuario que se encuentra en el primer lugar de la lista y reconstruir el modelo añadiendo información agregada con el fin de lanzar una predicción más detallada.

Resum

Aquesta tesina presenta una aproximació basada en aprenentatge automàtic i mineria de dades amb la finalitat de buscar el millor grup d'ítems d'un conjunt, segons el comportament observat en grups previs.

L'aproximació s'aplica dins un marc d'un sistema de mercat d'aigua, el qual té com a objectiu desenvolupar processos de negociació, on es creen taules de negociació per a comerciar els drets d'aigua dels usuaris. La nostra tasca es centrarà en predir quins agents mostraran el comportament més adequat quan són convidats a participar en una taula de negociació, amb el propòsit d'abastar l'acord més beneficiós.

D'aquesta manera, es desenvolupa un model que aprén de transaccions pasades ocorregudes al mercat. Aleshores, quan una nova taula de negociació s'obri per a vendre un dret d'aigua, el model prediu, tenint en compte les característiques individuals de la taula de negociació, el comportament de tots els agents que poden ser convidats a unir-se a la negociació, i així, convertir-se en potencials compradors del dret d'aigua.

Una vegada el model ha efectuat les prediccions per a una taula de negociació, els agents són classificats d'acord a la seua probabilitat (assignada pel model) de convertir-se en comprador en eixa negociació. Dos mètodes diferents es proposen en aquesta tesina per a tractar els participants classificats. Depenent del mètode utilitzat, d'aquest ranking podem seleccionar el número desitjat de participants per confeccionar el grup, o elegir solament l'usuari que es troba a la primera posició de la llista i reconstruir el model afegint informació agregada amb la finalitat de llançar una predicció més detallada.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Data Mining is the process of extracting useful and comprehensible knowledge from large amounts of data, which is stored in different formats [1]. It is the analysis of large observational sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. The relationships and summaries derived are referred to as models or patterns.

Thus, the goal of this discipline is finding intelligible models from this data, using appropriate techniques to analyze the data and extract new and useful knowledge. Data Mining can be viewed as a result of the natural evolution of information technology.

Data Mining is closely related to the concept of Knowledge Discovery in Databases (KDD). In fact, sometimes both concepts have been used in an indistinguishable way. But there are differences between them. As we will see in the second chapter, KDD refers to a process composed by several stages, where one of these stages is precisely, the Data Mining. The five stages in which is organized the process of KDD are: Integration and Data Collection; Selection, Cleaning and Data Transformation; Data Mining; Evaluation and Interpretation; Diffusion and Use. Data Mining is the most important step in the process, this is why KDD is often confused with the term of Data Mining. Later in this thesis we will go deeper in the connection between these two concepts and in how the KDD process works.

1.1.1 Emergence of Data Mining

Data Mining does not appear because of the development of very new technologies, but because of the emergence of new needs and the discovery of a new potential: the underused value of the high quantity of stored data in the information systems of all kind of organizations. This way, data is no longer a product, but becomes raw material ready to be exploited in order to obtain the main objective: knowledge. Knowledge is specially valuable for decision making in organizations [2].

How did these new needs come up? The amount and variety of information computerized in digital databases has increased spectacularly in last times. This historical information is highly useful to explain the past, understand the present and predict future information. Most of the decisions taken in organizations are based on information extracted from past experiences. Additionally, due to the diverse sources and formats of data, it seems obvious the need of analyze and unify data, in order to be able to extract useful information from it. Thus, the main task of Data Mining will be to solve problems analyzing data stored in databases.

The traditional approach to analyze the data from a database is done by means of SQL queries and using On-Line Transaction Processing (OLTP). However, this process only generates summarized information in a previously established way, and not scalable to large amounts of data. Databases technology has developed a new architecture in order to solve this, *Data Warehousing*. A Data Warehouse is a repository of heterogeneous sources of data, integrated and organized by a unified scheme in order to make easier the analysis and to support decision making. Data Warehousing includes On-Line Analytical Processing (OLAP) operations, like summaries and aggregations. The problem with OLAP tools is that they do not generate rules and patterns. In other words, knowledge that can be applied to other data, so as to inferring new knowledge and use it.

The limitations of traditional approaches that we just saw, have led to come up the need of a new generation of tools and techniques able to support the extraction of useful knowledge from the available information. These tools meet under the designation of Data Mining. Data Mining differs from traditional approaches because it does not obtain extensional information (data), but intensional (knowledge). Data Mining outcome are sets of rules, decision trees, neural networks... that can be used to extract useful and valuable knowledge.

1.1.2 Machine Learning and Data Mining

The field of Machine Learning [9] concerns the design of systems which automatically extract patterns from data. Machine Learning is a scientific discipline based on developing computational methods which can learn and improve from experience [32]. The term experience refers in this definition to training data. This set of data is formed by examples, each one corresponding to a real-world object, who may be accompanied by a label. The label encodes a property, a measurement of this object or a class/group membership [32].

The scenario where all the examples from the training set are labeled is called *Supervised Learning*, while *Unsupervised Learning* refers to techniques dealing with unlabeled data.

In supervised contexts, since Supervised Learning aims to obtain a function $h : X \longrightarrow Y$ (hypothesis or model) which labels unseen examples [32], it corresponds to *Predictive Models.* On the other hand, Unsupervised Learning corresponds to *Descriptive Models*, which are not concerned about predicting future data, but exploring the properties that characterize the studied data.

1.1.3 Negotiation

The field of automated negotiation provides techniques and mechanisms to allow several agents to interact and trade for assets, in exachange for economical compensations. Agents are an essential unit in these trading procedures, due to they and their individual interests are the triggers and the recipient of any comercial transaction that can be proceeded within a negotiation framework. Controlling and predicting the behaviour and proceedings of the agents turns to be vital in order to ensure an optimal performance of a negotiation, and consequently, to ensure also the satisfaction of the agents themselves.

Since the only information that we can handle before a negotiation process is the historical data about past transactions ended in the same market system, it seems clear the usefulness of benefit from this advantage and exploit this information in order to learn from past transactions and predict the behaviour of agents in future negotations. In order to find behaviour in past negotiations, namely, extract knowledge from stored data about transactions, we can make use of data mining techniques.

1.2 Purposes

The main goal of this thesis is to make an approach to a problem which has so far been solved manually, as a data mining problem, covering all the stages integrated in the KDD process. Concretely, we work within a framework which models a water-right market system. Our task will be, given a set of items, search the best group of items according to the behaviour of previous groups.

This is a new approach in the way in which the market systems between real users are considered. The water-right market corresponds to a virtual market system in which water rights will be traded by water users and transfer agreements will be executed by them under specific conditions. The idea of our work is to apply machine learning and data mining in order to provide the best trading tables (the basic mechanism in the framework for trading an asset) that can be formed, by means of predicting the water users that will behave better and, consequently, achieve the best transfer agreement. The quality of a transfer agreement will be determined by some criteria defined by the framework, such as maximum economical benefit, social welfare, individual satisfaction, etc. With the purpose of providing the participants that will show the best behaviour in a certain situation, we aim to develop a model which learns from the past transactions recorded in the system's database and predict future behaviour of users under specific conditions.

Since we work within a framework that implements a virtual market, it seems obvious the relation of our problem with the field of automated negotiation. The problem we face here can be also defined as a negotiation problem approached from a data mining perspective, in order to select a group of negotiating agents, who fulfill certain requirements.

The process is organized along this document as follows:

- First, the historical data of the water market, stored in a database, is studied, analyzing which aspects must be considered and which attributes are relevant for our purposes.

- When the necessary data is extracted from the database, a minable view is generated, and a data mining process is developed, in order to learn a model able to understand the users' historical behaviour and to predict future behaviour in a successful way.

- Then, in order to evaluate the quality of the model to check if it is valid enough, we split data in two parts, assuring that the evaluation of the models is done over a different data set from the training set used for learning those models. As a result of the evaluation, a ranking of users is generated, giving to the market system a recommendation about the best participants that each trading table needs, in order to achieve certain objective or maximize some feature.

1.3 Organization

This thesis is organized in seven chapters, including this introduction:

- Second chapter (Preliminaries): the concept of Knowledge Discovery in Databases is introduced and deeply detailed by means of an explanation of each of its stages. Furthermore, KDD and Data Mining are connected and the differences between both concepts are submitted.

With regard to Data Mining, we analyze its tasks and methods, trying to present them in a clear and understandable way. In this part it is also introduced the concept of negotiation. We take a look to its main characteristics and to its relation with the application developed.

- Third chapter (Working Hypothesis): here is analyzed the working hypothesis, in which we study the supervised learning and some learning algorithms, focusing on decision trees induction. Especially, we review those decision trees that work with probabilities, since this kind of trees will be the technique used for the generation of the model in our application.

- Fourth chapter (Problem Definition): this chapter encompasses the definition of the problem faced in this thesis; in the first section we present the Case Study, paying attention to the question that we are trying to solve, and in the next section we totally focus on the water market system, reviewing the most important issues of its implementation. Also, we take a look to the Requirement Specification of the virtual market system and we summarize the activities developed in the market scenario, grouping them according to the space of the problem where they are located.

- Fifth chapter (Data Description): this chapter details the data used in the application. The first section lists all the tables from the database that are essential for our work, namely, those that are directly related to the market view of the system implementation. Next section focuses on the selection of the data, reviewing the minable views generated and showing an example of a query that extracts information from the tables presented in the previous section and several aggregate functions defined.

- Sixth chapter (Making mWater recommendations based on probabilities): the different approaches developed for the problem are examined in this part. We explain how do they work and the differences and coincidences between them. The explanation is accompanied by several examples in order to visualize more clearly the process followed.

- Seventh chapter (Implementation and Experimental Evaluation): the evaluation of the approaches mentioned in the sixth chapter is presented here. We show and comment the results thrown by the application and we evaluate its quality and goodness, detailing the criteria used for that evaluation.

Finally, the last section corresponds to a conclusion to the present work.

Chapter 2

Preliminaries

2.1 Knowledge Discovery in Databases and Data Mining

Knowledge Discovery in Databases (KDD) is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [7].

As it has been mentioned, usually the concept of KDD is confused with Data Mining. Obviously, they are terms very related to each other, mainly due to the fact that Data Mining is a stage in the process of KDD. While KDD is the global process of discovering useful knowledge from databses, Data Mining refers to the application of learning and statistical methods in order to obtain hypotheses about patterns and models. Because of Data Mining is the stage where hypotheses are generated, usually KDD is referred with the term Data Mining.

KDD is a complex process that includes not only the generation of the models and patterns, but also includes the evaluation and possible interpretation of them.

2.1.1 KDD Stages

KDD is an iterative and interactive process. It is iterative since the output of some stages can lead to previous steps, and sometimes several iterations are needed to extract high quality knowledge. It is interactive because the user should help in activities like data collection, validation of extracted knowledge, etc.

Often, it is added a previous phase about the analysis of the organization needs and problem definition [8], in which the purposes of the Data Mining are established.

KDD process is organized in the following five stages, which are pictured in the Figure 2.1.

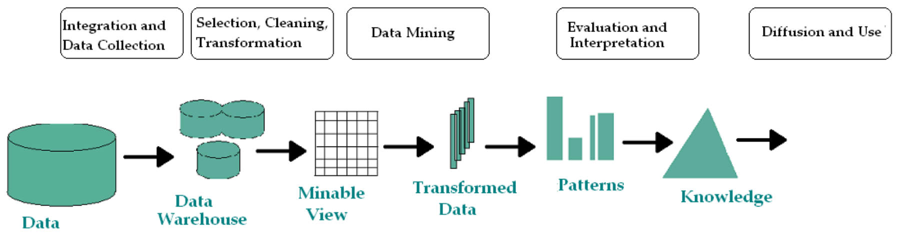
Integration and Data Collection
Selection, Cleaning, Transformation
Data Mining
Evaluation and Interpretation
Diffusion and Use
Data
Data Warehouse
Minable View
Transformed Data
Patterns
Knowledge

Figure 2.1: Stages in the KDD process.

1. Integration and Data Collection

   The need of the activities performed in this stage comes from the fact that databases based on On-Line Transaction Processing (OLTP) are not enough to execute complex functions like analysis, planning and prediction, namely, long-term strategic decisions. In fact, OLTP covers daily needs of an organization but it is not intended to deal with the functions mentioned. In this situations, usually the necessary data for a KDD process comes from different organizations or different and heterogeneous sources. The problem of assemble a dataset that makes possible the knowledge extraction requires the decision of which internal and external sources are going to be used, how are we organizing them, how will be they keep along time and finally, and in which way could we extract information from them. The first step is to integrate this data. The integration is possible thanks to a data warehousing. This term refers to the tendency in the institutions to collect data from several sources, stored by a unified scheme usually located in a single location. Data warehousing is a relatively recent technology that tries to provide methodologies in order to collect and integrate historical data of an organization. Its goals are analysis, summaries and complex reports and knowledge discovery.

   Thus, although data warehouses are not essential for knowledge extraction from data, they are very recommended, because of the several advantages that they provide, when facing large amounts of data, or data providing from heterogeneous sources or randomly combined. Data warehousing makes the task easier to us.

2. Selection, Cleaning and Data Transformation

   The quality of the knowledge discovered not only depends on the mining algorithm used, but also on the quality of the data mined. Next step is about selecting and preparing the subset of data for the mining, which would be part of the minable view. Therefore, data collection needs to be followed by a data cleaning, in order to be ready for the analysis. Related to this, some of the data collected are irrelevant or innecessary for the mining task. Another important problem is the occurrence of values that does not adjust to the general behaviour. These values are called outliers, and they can represent bugs in the data or they can be just correct values that are significatively different. Data mining algorithms does not have a common pattern to treat them, since some algorithms ignore them and others discard them. The reason of this disparity is that we shall be very careful with outliers because there are applications where anomalous events are more interesting than ordinary ones. As we can see, it will depend on the application and on the individual situations.

Missing values is another problematic issue, due to that they can lead to inaccurate results. There are several approaches to deal with missing values. It is important to know the reason why the values are missing. For example, they can represent nonexisting values, incomplete data (as a result of a union of fields from different sources) or maybe missing values can express relevant information. In general, the potential actions to be developed to handle this problem are: ignore them, remove the corresponding attribute, filter the row or replace the value by an estimated value which does not interfere. The replacing solution is one of the most used.

The presence of problems like outliers and missing values makes more obvious the need of assuring the quality of the data and the right selection of the attributes.

If we select as a minable view all the information that could be relevant, it can turn it into a huge minable view with too many columns and rows. It is more interesting to find a subset of instances in order to handle the data more appropriately. Apart from reducing the size of the minable view, the selection helps to improve the outcome. This stage aims to present data in the most suitable way for data mining. The output will be a set of rows and columns known as minable view. This minable view will be a virtual table containing the data prepared to modelization.

3. Data Mining

   The Data Mining stage is the most characteristic of KDD and it is usually used to name the whole process. The goal is to produce new knowledge that the user can apply. This is done by constructing a model based on collected data from previous stages. The model is a description of the patterns and relations among data that can be used for predictions, for a better understanding of data or to explain past situations.

Before starting the process is necessary to take three important decisions:

- which kind of mining task is more appropriate
- choose the type of model
- choose the right algorithm that solves the task and obtains the model we are looking for

Related to the concept of Data Mining, it appears the term of inductive learning. First of all, we present four different definitions of inductive learning:

- the improvement of the performance with the experience [9].
- the ability to predict plausible future observations or to explain past observations.
- the identification of patterns, regularities, existing in the evidence.
- the removal of redundancy, seen as information compression [10].

These four approximations are perfectly combined in the sense that the learning allows us to identify regularities in a set of observations. These regularities are redundancies that can be represented by patterns, which will be used to predict future observations or explain past ones.

The two last definitions correspond to inductive learning. This is a special kind of learning, which comes from particular cases (examples) and generates general cases (models) that abstract the evidence.

All the tasks and methods defined in Data Mining, which will be detailed in further sections, are focused around the idea of inductive learning.

4. Evaluation and Interpretation

   This stage evaluates the quality of the discovered patterns by a Data Mining algorithm. It is not a trivial process, since the criteria involved is very subjective. Ideally, patterns discovered must have three qualities, namely, be accurate, understandable and interesting. Depending on the application, one quality can become more important than other.

   Thus, this phase tries to verify that the patterns produced occur in the wider data set. In other words, we want to know if a certain model obtained in previous stages is valid enough for our purposes. There are several methods to evaluate the quality of a model from the evidence. An optimal option for the evaluation is to test the models over a different data set from the training set. Data is divided in two parts, one for training and one for test. This separation is necessary to guarantee the independence of the measure taken in the evaluation of model accuracy. Usually, partition is made randomly, for example taking 50% of the data for the training and the remaining 50% for the test subset. This percentage varies according to the application needs (80%-20%, 75%-25%...).

A problem originated by a random partition is that it may occur that two different experiments done by the same evidence and the same learning method, can obtain disparate outcomes. Also, when the problem has not enough data, it is not a good idea to reduce even more the data by splitting them.

A solution that reduces the dependency of the results on the way in which the data is partitioned is called cross-validation. This method consists in dividing the set of the evidence into k disjunct subsets of similar size. A hypothesis is learned using the set composed by the union of $k-1$ subsets, and the remaining subset is used to calculate a partial sampling error. This process is repeated k times, each time using a different subset to estimate the sampling error. The final error is calculated as the arithmetic mean of the k final errors. This way, the final result has the mean of the partial expermients with k independent test subsets.

Another technique that is specially indicated in order to estimate the error of a model when there is not enough data, is bootstraping. This method builds a model with the initial data, and then creates several data sets, called bootstrap samples, by sampling the original data with replacement. This means that instances from the original set are selected (same instance can be selected more than one time). When the sets are created, a model is built from each set and we calculate its error rate on the test set. Final estimated error is figured averaging the errors obtained for each sample.

With regard to the interpretation, Data Mining produces several models whose interpretation by the user is essential to the success of the knowledge extraction process. Because of this, some methods to visualize the results of learning stage have been implemented. Visualization of models allows users to identify clearly the more significative patterns that the model has discovered. Additionally, some visualization methods enable users to modifiy the represented models in order to refine them or adapt them according to the circumstances.

5. Diffusion and use

   Once the learned model is built and validated, it can be used with two main goals: by an analyst in order to recommend actions based on the model and its results, or to apply the model to different data sets. Whatever its use is, it is necessary a diffusion of the model, namely, a distribution and communication to the potential users.

   It is also important to measure how the model evolves. We should check periodically the performance, due to the variability of the patterns and external factors, that may lead us to a re-evaluation and reconstruction of the model.

2.1.2 Data Mining. Tasks and Methods

Data Mining aims to analyze data to extract knowledge. This extracted knowledge can take the form of relations, patterns or rules inferred from data and previously unknown, or also can take the form of a concise description (a summary). These relations and summaries represent the model of the analyzed data. Models can be of two types: *Predictive* or *Descriptive*.

As it has been said, Predictive Models represent the tasks that can be developed within a Supervised Learning context. In this supervised scenario, all the examples of the training set have a label that reveals a property of the object represented by the example. The aim is to predict the value of a function $h : X \longrightarrow Y$ for an unseen example, after having studied a number of training examples. For that purpose, the learner generalizes from the presented data to unseen situations. Thereby, predictive models try to estimate future or unknown values of certain variables (target variable) using other fields of the database (predictive variables). This kind of tasks always has an output. Within the schema of Supervised Learning, we can differentiate between *classification* and *regression*. When the set Y is without order, we talk about classification, and when Y is a continuous set of ordered values, it is called a regression task.

Below, we will see a review of the data mining tasks and then we will study several methods that can be applied to solve these tasks. Further in this section is pictured in the Table 2.1 the correspondence between the tasks and the methods that we are about to examine.

Tasks

Classification is the task of generalizing known structure to apply it to new data. The set Y has nominal values, so that it can take a set of values known as classes. A learned function will be able to determine the class for a new unlabeled example: it will assign a value from Y for each value from X. When there are only two classes, the task is called *binary classification*.

Another variation of classification is *soft classification*. Besides from the function h, it learns an extra function that represents the certainty of the prediction produced by h. This kind of extension enables other applications, like a ranking of the predicitions or a selection of the best k examples.

With regard to Regression, it attempts to learn a function that represents the correspondence between the examples, for each value of X it assigns to it a single value for Y. Unlike classification, here the set Y is numeric, and can be integer or real. In this case, the goal is to minimize the error between the predicted value and the actual value.

Descriptive Models, on the other hand, meet those tasks that can be developed within a Unsupervised Learning context, namely, where the training data set given to the learner contains only unlabeled examples. This kind of tasks attempt to identify patterns that explain the data. That is, descriptive models explore the studied data and provide information about its relationships and propierties, but do not predict new data. As a result, descriptive tasks do not have an output attribute. Examples are presented as an unlabeled and unordered set. The major challenging here consists in discovering hidden groups or unknown classes in data [32].

The data mining tasks that result in descriptive models are *clustering*, *association rules* and *correlations*. Clustering is the task of discovering groups of individuals among the elements of a set h, such that the elements assigned to the same group are similar. It differs from classification in the fact that, the groups and the groups membership are what we want to clarify, and besides, classes and its number are unknown, in order to create different groups. Sometimes, the number of desired groups is determined by the clustering algorithm. The function to obtain is identical to the function mentioned in supervised contexts, with the difference that the values on Y and the members of the set will be created along the learning process. The main utility of clustering is to decide the behaviour of a new instance according to which group it belongs, due to the certainty that all the instances of a group act in a very similar manner.

In regard to correlation, this task analyzes relationships between numerical attributes exclusively, so that we are able to detect if several attributes from a set are linearly correlated or related in some other way. By exploring relationships, we can determine hypotheses and discover associations in the data set.

The last task that we are talking about in the unsupervised learning context is association rules. Its purpose is similar to correlation but it applies to nominal attributes. It searches for relationships between variables, so that we get an association between two attributes when the frequency of having certain values of each one together in a certain moment is quite high. Usually, we look for more than one association rule, namely, a set of association rules that work good together.

Methods

In the data mining context, a type of task refers to a type of data mining problem. *Methods* are the techniques that let us solve the data mining tasks. One single task can be solved by several methods, and also one method is able to solve many different tasks. The type of knowledge that we want to extract determines the possible techniques that can be used. In the Table 2.1 we present a correspondence between the tasks we have seen and several techniques.

Algebraic and statistical methods are techniques based on expressing models and patterns by means of algebraic functions, linear and no linear functions, distributions or statistical aggregate values. Usually, a pattern is obtained from a predetermined model, from which some parameters are estimated. The most known algorithms within this group of techniques are the linear regression and the logistic regression. The simplest regression function is the linear, in which each explanatory variable participates in a constant and additive way for all the observed domain of the answer [2]. Meanwhile, logistic regression is used to solve soft classification. This kind of task obtains a probability estimation for categorical variables. Thereby, logistic regression models a probability. In this case, the output variable has two or more possibilities, each one with its respective probability.

Moreover, one of the most used techniques in data mining problems are the *Bayesian Methods*, which are based on the estimation of the probability of belonging to a certain class. According to [9], the two main reasons of the relevance of bayesian methods in data mining and automatic learning are, on the one hand, because they are practical methods to make inferences from data, inducing probabilistic models that will be used to formulate hypotheses about new examples. And also because they calculate explicitly the probability associated to each hypothesis. On the other hand, bayesian methods enable a useful framework for the comprehension and analysis of multiple learning techniques and data mining techniques which do not work directly with probabilities. Regarding to its inconvenients, the main problem that bayesian methods have to deal with is the computational cost that they require. It is usual to restrict the models by some kind of simplifications. Despite these simplifications, these models turn to be very competitive. Here we can find algorithms like Naive Bayes Classifier and the EM Algorithm. Before discussing these algorithms, we need to introduce the concept of Bayesian Networks. They are a formalism that represents the qualitative knowledge of the model by means of a directed acyclic graph. This knowledge articulates around the definition of independence/dependence relations between the variables that compose the model [2]. Bayesian networks constitute a very attractive tool in its usage as knowledge representation, because they use graphical representation of the model. Naive Bayes is the simplest classification model in this area. In this case, the structure of the networks is fixed and we only have to learn the parameters. The *Naive Bayes Classifier* [56, 57] assumes that all the attributes are independent when the value of the class variable is known. Despite of this assumption, Naive Bayes Classifier is one of the most used classifiers, and it has been proved [58] that its results are very compelling.

Other classifiers based on bayesian networks are TAN and BAN. TAN (Tree Augmented Naive Bayes) [61] is an extension of Naive Bayes Classifier that tries to improve the hit rate during the classification by means of the specification of several dependencies between the attributes, by assuming that the attributes form a bayesian network in a tree-structure. BAN (Bayesian Network Augmented Naive Bayes) learns a bayesian network for the attributes (except the class) and then increases the model by adding the class variable and edges from it to all the attributes.

On the other hand, EM Algorithm *(Expectation Maximization)* [59, 60] can be used in problems where we need to estimate the distribution tables associated to each node of a bayesian network. The algorithm is organized in two iterative stages, where the first one calculates the expected frequencies from the model, and the second phase deals with the reestimation of the paramaters from those expected frequencies. The process stops when reaches the convergence.

Another common technique are the methods based on *decision trees and rules learning.* This kind of techniques will be deepen in its corresponding section, but here we are going to see a brief review of them. Both perspectives are techniques that can be represented by a collection of rules. In the case of decision trees, these rules are structured like a tree. These methods turn out to be very comprehensible, in the sense that they are expressed in a symbolic way, by building a set of conditions, and so, the models generated are extremely intelligible and usable for humans. In the chapter dedicated to decision trees induction and rules learning we will have a more intensive review.

Neural networks are another interesting approach. *Artificial neural networks* are a learning method that aims to emulate biological processors of information. They use artificial supports similar to those that exist in human brains. These techniques learn a model by training the weights that connect a set of nodes. The network topology and the connection weights define the learned pattern. Basically, an artificial neural network consists of units called neurons. Each neuron receives several inputs through interconnections (synapsis), and releases an output. This means that in the network, the outputs of some neurons connect with other neurons inputs. An output is characterized by a propagation function that has the effect of excitation when the weight between two neurons is positive. Instead, if the weight is negative, the effect is inhibitory.

Artificial neural networks can be divided by those that use supervised learning and those that use unsupervised learning. Supervised learning provides the network with an input dataset and with the right answer. The input dataset is propagated forward until the activation reaches the neurons of the output layer. In that moment, we can compare the calculated output with the target output. Then the weights are adjusted in order to assure a correct answer if the same input pattern is received again [2]. One of the most used neural netwokrs in this context is the *Multilayer Perceptron*. This consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. The activation propagates through the weights from the input layer to the intermediate layer, where a function of activation is applied to the inputs received. Then the activation propagates till the output layer. Multilayer perceptron overcomes the limitations of the previous *Perceptron*, since due to its multiple layers it can solve problems not linearly separable. The *Backpropagation* algorithm [62] is the most known technique that implements this kind of neural network. Its learning algorithm works as follows: The weights are initialized to random values, as well as the input dataset. The activation is propagated forward through the weights until the activation reaches the output layer. Then, the network calculates the error of each inputs (by comparing the output generated with the desired output) and also of each hidden layer. The weights are now updated by back-propagating the error backwards to the previous layers until the input layer.

Meanwhile, in the unsupervised learning, the network is provided only with an input dataset. This means that models are discovered only from the input data, and does not exist an ouput value that we can use to compare. The network must organize itself by means of reacting to some kind of structure existing in the dataset. The two methods mostly used for unsupervised learning with neural netoworks are the Hebbian Learning and the Competitive Learning. *Hebbian Learning* [63] is used to obtain projections or optimal compressions of data sets. The conjecture described by Hebb says that when a modification is produced between two neurons, the influence of one over the another is increased. In a network with forward propagation this statement means that the weight between an input neuron and an output neuron strengthens when the activation of the input neuron reaches the output neuron and activates it. This connection between two neurons is called synapsis. On the other hand, in the *Competitive Learning* takes a place a competition between the neurons of the output layer for the right to respond or activate. The goal is to find groups within data (clusters). One of the most important variations of competitive learning are the *Self-Organizing Maps (SOM or Kohonen Maps)* [64]. This type of neural network produces a two-dimensional discretized representation of the input space of the training samples, which is called a map. The input layer contains the examples and in the competitive layer each cell represents a prototype. The objective of the training is to cause the prototypes to capture similar examples (in order to ensure that different parts of the network respond similarly to certain input patterns). The Kohonen algorithm states that there is a competition between the neurons of the output layer. The difference is that not only the weights of the winning neuron are updated, but also the weights of the neighbour neurons. This is defined by a neighbourhood function, which preserves the topological properties of the input space. However, in the training the number of neurons affected by this function decreases with time.

Another variation of competitive learning is the *Learning Vector Quantisation* (LVQ) [65]. It is based on Kohonen neural networks, although it is a supervised technique employed in classification problems. The main difference from Kohonen Maps is that in this technique the number of classes is known previously, so that in the competitive layer is introduced this number of cells (or a bigger number). These cells represent the prototypes, which are distributed randomly along the input space. Once the network is trained, the solution to the classification problem will be the disposal of those prototypes with the rule of the nearest neighbour. The prototypes are allocated according to the values of the weights of the connections between the input layer and the competitive layer. The rule of nearest neighbour works this way: the new unclassified example is introduced. It is calculated the distance from this example to all the existing prototypes. The example is labeled with the class of the prototype that has the lowest distance [2]. When comparing LVQ to SOM, we can observe that in LVQ does not exist the concept of neighbourhood, and only is modified the winning cell (and not always in the same direction).

Otherwise, the *Support Vector Machines (SVM)* try to maximize the margin between several formed groups, by means of transformations (denominated kernels) that can increase the dimensionality. The possible variations depend on the kernel and the margin used. SVM are based on a simple linear classifier. The standard SVM takes an input data set and predicts for each given input, which of two possible classes the input is a member of. Given a set of training examples, a SVM training algorithm builds a model that assigns new examples into one category or the other. A SVM model is a representation of the examples as points in space. The simplest model of SVM is the maximal margin linear SVM. Assuming that the data set is linearly separable in the input space, the examples can be divided by an hyperplane such that in each side of it, there are only examples belonging to a single class.

Last method we are seeing are the Case-based Reasoning techniques. This type of methods aim to solve a problem by using information extracted from a set of previous examples. *K means clustering* is the most popular method of cluster analysis in this area. This method starts with several prototypes and a group of unlabeled examples, and it tries to place the prototypes in the space, such that data belonging to the same prototype have similar features [66]. When the prototypes have been placed, each new example is compared with them and associated to the nearest one.

Another important technique based on distances is the *K-NN (k-nearest neighbour)* [67]. This method assigns the majority class among the k nearest neighbours. This way, an object is classified to the class most common amongst its k nearest neighbours. The main issue within this context is to determine the optimal value for k.

Technique	Predictive		Descriptive		
	Classification	Regression	Clustering	Association rules	Correlations
Neural Networks	✓	✓	✓		
Decision Trees	✓(C4.5)	✓(CART)	✓		
Kohonen Maps			✓		
Linear Regression		✓			
Logistic Regression	✓				
Kmeans			✓		
Apriori				✓	
Factorial Analysis					✓
CN2 rules	✓			✓	
K-NN	✓	✓	✓		
Naive Bayes	✓				
SVM	✓	✓	✓		

Table 2.1: Correspondence between tasks and techniques.

2.2 Negotiation

In the context of the application implemented in this dissertation, negotiation arises as an important issue that we must pay attention to. Since the water market system is developed within an environment of a Multi-Agent System, where the trading mechanism constitutes a crucial stage in the framework, we need to define some of the main characteristics of the negotiation process.

As it is defined in [34], the concept of negotiation involves determining a contract under certain terms and conditions. Automated negotiations form an important type of interaction in agent-based systems for e-commerce. As it happens in our virtual water market, automated negotiation provides a mechanism that allows sellers and buyers to exchange offers iteratively to reach acceptable agreements. A good automated negotiation can both save time and find better deals in the current complex and uncertain e-commerce environment [35].

A decisive aspect involved in a negotiation context and that is very important in our application, is experience. Good negotiation skills in humans seem to come from the experience [34]. That is because, when we face a negotiation situation, we make use of past negotiation experience and strategies as guidance in suggesting suitable ways to proceed.

Collecting and analyzing all behaviour data obtained during the negotiation process and about the negotiation result will improve our understanding of the actual behaviour of negotiators during a negotiation process [36]. For finding behaviour in negotiations, in this thesis we propose to use Data Mining techniques.

When we talk about a Multi-Agent System, we should stop on the concept of *agent*, and think about it. [38] and [39] say that agents are being advocated as a next generation model for engineering complex, distributed systems. Meanwhile, [40] and [41] mention that agents are being used as an overarching framework for bringing together the component AI subdisciplines that are necessary to design and build intelligents entities. A description that seems to be mostly accepted claims that an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives [39].

Negotiation is seen in this environment as the process by which a group of agents come to a mutually acceptable agreement on some manner [37].

Following the distinction appeared in [37], and more detailed in [42], automated negotiation research deals with three broad topics:

- Negotiation Protocols: the set of rules that govern the transaction. This refers to the permissible types of participants, the negotiation states, the events that change negotiation states and the valid actions of the participants.

- Negotiation Objects: the range of issues over which agreements must be reached.

- Agents' Decision Making Models: the system by which the participants act in line with the negotiation protocol in order to achieve their individual objectives.

Even so, there is no universally best approach for automated negotiation. In [37] is described a generic framework for automated negotiation, where a negotiation is viewed as a distributed search through a space of potential agreements and where participants are the active components that determine the direction of the search. Each agent who gets involved in the negotiation process, has at the beginning a portion of the space in which it is willing to make agreements. That space may change during the negotiation process, and the search comes to an end when the required number of participants find a mutually acceptable point in the agreement space or when the protocol dictates that the search should be terminated without making an agreement.

Within the context of a very basic negotiation, the minimal negotiation capabilities are to propose some part of the agreement space as being acceptable, and to respond to such a proposal indicating whether it is acceptable [37]. If we take as an example the simplest kind of negotiation, that is, a Dutch auction [43], we can see that the process develops as follows [37]. The auctioneer calls out prices. When there is no signal of acceptance from the other parties in the auction, the auctioneer makes a new offer which it believes will be more acceptable (by reducing the price). In this protocol, a lack of response is sufficient feedback to infer a lack of acceptance. In more complex protocols, the minimal feedback requirement is that other agents indicate dissatisfaction with proposals that they find unacceptable.

A negotiation where agents can only accept or reject others' proposals can become very inefficient and a time wasting. In order to improve the efficiency of the negotiation process, the recipient needs to be able to provide more useful feedback on the proposals it receives. This feedback can take the form of a *critique* (comments on which parts of the proposal the agent likes or dislikes) or a *counter-proposal* (an alternative proposal generated in response to a proposal) [37]. Obviously, the more information placed in the critique, the easier it is for the original agent to determine the boundaries of the other agent's agreement space. In contrast, counter-proposals differ from critiques in that the feedback is less explicit but generally more detailed. This way, by using critiques and counter-proposals, according to the feedback received, the proposer should be in a position to formulate a new proposal that is more likely to be in concordance with the other users' wishes and, in consequence, more likely to lead to a satisfactory agreement.

Within the framework proposed in [37] are discussed three negotiation techniques, which we are going to review below. But before discussing these techniques, we need to comment the main kinds of negotiation protocols that are implemented in our application. The aforementioned dutch auction is a type of auction where the price of the selling item is lowered until it gets a bid. When the auction starts, there is a defined price for the auctioned item, and the buyer can bid that price. If no buyer places a bid, the item price is decreased by a percentage of its initial price. In the case that a bid for the actual price is placed by a buyer, the auctioneer decides whether to accept or to refuse the bid. Finally, the winning participant pays the last announced price. The basic paramaters to start a dutch auction in our implementation are the initial price, the minimum price and the waiting time between rounds [16].

Another possible negotiation protocol is the English Auction. In this auction, bids can be placed when the auctioneer is calling for bids. Usually, the auction is opened with a suggested opening bid, and the bids must be progressively higher in value. Each round terminates with a temporary allocation of the good being auctioned to the prospective buyer that meets the auctioneer call [42]. The parameters needed to start a english auction in our application are the initial price and the length of the timeout [16].

A variation of the english auction is the Japanese Auction. When the bidding starts, no new bidders can join, and all the bidders participating are considered in the auction with a continuously rising price. Each bidder must continue to bid each round or drop out.

A basic and useful protocol which is also implemented is the Face-to-face strategy. In it, the seller who is trading with the item negotiates individually with each buyer interested in the purchase. This is a very extended protocol thanks to its simplicity and its intention to facilitate the rapprochement between parties involved.

Finally, the last protocol implemented in the market is called Blind Double Auction. As it is defined in [16], all the members participating are matched by the auctioneer with another member that has submitted the same price. If a bid from an agent matches with another agent's bid, the auctioneer informs the agent about the agreement that was just created in the double auction, so each agent may finally know who is the other agent he will trade with. If after several counter-offers all the participants of the blind double auction are not matched, the round is restarted with this call and all the members must make a new offer.

After seeing the negotiation protocols used in the framework, in next sections we are presenting several negotiation techniques.

2.2.1 Game Theoretic Models

Game theory is a branch of economics that studies interaction between self-interested agents [37]. Techniques and results of game theory can equally be applied to all interactions that occur between self-interested agents. Game theory is relevant to the study of automated negotiation because the participants in such negotiations can reasonably be assumed to be self-interested.

If we imagine a particular negotiation scenario involving several automated agents, two key problems arise on which the techniques of game theory can be applied:

- The design of an appropriate protocol in order to govern the possible interactions that may occur between the participants of the negotiation. The protocol defines the "rules of encounter" between agents [46].
- The design of a particular strategy that can be used by the individual agents during the negotiation process. It seems obvious that an agent will aim to use a strategy that maximizes its own individual welfare [37], in terms of economical compensation or other kind of profits, which depend on the particular case of each participant.

There are several problems associated with the use of game theory when applied to automated negotiation. The first problem is that game theory assumes that it is possible to characterize an agent's preferences with respect to possible outcomes but, in real world it is difficult to define human preferences in a simple way, especially in complex situations that involve multi-issue preferences. Second disadvantage is that this theory has failed to generate a general model governing rational choice in interdependent situations [47]. Some very specialized models have been produced, instead of a generic one that encompasses all of them. These separate models created are applicable to specific types of interdependent decision making. Last problem that has emerged is that game theory models often assume perfect computational rationality meaning that no computation is required to find mutually acceptable solutions within a feasible range of outcomes [37]. Nevertheless, in real world cases, agents typically know their own information space but totally ignore that of their opponent.

In conclusion, despite of the problems highlighted, game theory is extremely compelling as a tool for automated negotiation [37]. Especially in those scenarios where the preferences and strategies of the participating agents are known. This approach can turn out to be very useful in those cases.

2.2.2 Heuristic Approaches

Heuristic methods overcome the aforementioned limitations of game theory models. These techniques acknowledge that there is a cost associated with computation and decision making and so seek to search the negotiation space in a non-exhaustive fashion. This has the effect that heuristic methods aim to produce good, rather than optimal solutions [37].

The main advantages that this approach offers are the following. The models produced are based on realistic assumptions and, because of that, they are capable to provide a more suitable basis for automation. Besides, the designers of agents can use alternative models of rationality to develop different agent architectures.

This line of work is concerned about modelling the agent's decision making heuristically during the course of the negotiation, so that, the chosen protocol does not prescribe an optimal course of action.

[37] has developed a set of deliberation mechanisms that work within a fairly free negotiation protocol [48, 49, 50]. Within this framework, the space of possible agreements is quantitatively represented by contracts having different values of each issue. The participants have to rate these points in the space of possible outcomes, so that proposals and counter-proposals become offers over these points. The termination of the search is determined by the expiration of the time agreed or by the formulation of a mutually acceptable solution. This corresponds to a point of intersection of the agents' acceptable outcome. An agent architecture that models the decisions included in this search has been developed in [49].

Decision making within this mechanism is based on a linear combination of simple functions called tactics, which manipulate the utility of contracts [48].

Although heuristic approaches solve the disadvantages of the game theoretic models, they also charge with their own problems. According to [37], the two main drawbacks of these methods are that the models often select outcomes that are sub-optimal and that they need extensive evaluation.

2.2.3 Argumentation-Based Approaches

The basic idea of the argumentation-based approach is to allow additional information to be exchanged, over and above proposals. This way the approach aims to remove the limitations showed in the negotiation systems aforementioned.

The information to exchange can be of a number of different forms, all of which are arguments which explain explicitly the opinion of the agent about the proposal received or sent. Thus, in addition to rejecting a proposal, an agent can offer a critique of the proposal, explaining why he finds it unacceptable. Similarly, in reverse, an agent can accompany a proposal with an argument which says why the other agent should accept it, specifying the benefits that the other agent could obtain in the transaction. This procedure makes it possible to change ocasionally the other agent's region of acceptability (by altering its preferences), and also provides a means of changing the negotiation space itself.

As it happens in human argumentation, agents may not be truthful in the arguments that they elaborate. Thus when evaluating a received argument, the recipient needs to asses it on its own merits and then modify this by its own perception of the argument's degree of credibility in order to work out how to respond.

The argumentation mechanism implemented in [51] builds on work in argumentation as an approach to handling defeasible reasoning. This way, agents are able to handling the frequent contradictory statements, allowing the resolution of conflicting arguments that may lead the negotiation process to failure. Using agumentation in real agents (as opposed to simple collections of logical statements) means handling the complexities of the agents' mental attitudes, communication between agents, and the integration of the argumentation mechanisms into a complex agent architecture.

In [52] was showed how to augment a standard model of argumentation to work for agents which reason using beliefs, desires and intentions.

2.2.4 Machine learning techniques in negotiation

The summary of the negotiation techniques that we have just seen reveals that automated negotiation has not faced the problem of making a selection of the most appropriate agents to participate in a particular negotiation scenario. Nowadays, this selection is done manually by a determinate person responsible of assembling the group of invited agents. Depending on the particular case, this person will take certain specific criteria as guidance to select the agents.

The distinction that we want to introduce in this thesis is that this group of agents who participate in a negotiation process will be prepared following the criteria set by the historical information of past transactions. Based on the behaviour observed in historical negotiations, we want to predict the behaviour of the agents in future negotiations, in order to select the most suitable group of agents for a given negotiation scenario.

Since the only certain information available before starting a negotiation process is the historical data stored about past transactions, our application aims to learn from this information, concretely that relative to the behaviour of the agents, paying special attention to its role in a transaction (whether if they have achieved an agreement or not). We attempt to introduce the capabilities of machine learning and data mining techniques by following the stages dictated by the KDD process in order to apply a model that learns from historical behaviour and predict the future behaviour of each participant in a certain negotiation scenario, which in our case will correspond to a trading table.

Focusing in the system implemented, our virtual market incorporates and defines its own negotiation mechanisms in order to govern and control all the trading process. More concretely, within this system we are interested in the market view, which is the view where it is implemented the environment of trading tables and agents. Here, during a negotiation process, a group of agents come (or not) to an agreement by using the negotiation mechanisms available and satisfying the rules established in the market scenario.

An approach that also has used data mining techniques in negotiation is [34]. This work has proposed a negotiation model which tries to learn past behaviour of buyers in order to provide the seller with information that he can use to adopt an effective interaction strategy, given a set of registered strategies. In that work, the knowledge extracted from previous negotiations is used to decide strategies during the negotiation, according to the behaviour and the proposals from the other agent. The question they want to answer is which strategy should a seller choose for negotiating with a new buyer. This is solved using experience from previous negotiations and analyzing them. This way, when a seller needs to decide a strategy, he just has to find a cluster of buyers with similar behaviour to the behaviour demonstrated by the buyer with he is dealing in the current negotiation.

Chapter 3

Working Hypothesis

As it has been said, the problem we face here consists in finding certain set of appropriate elements, specifically participants, who give the best performance according to a criterion based in their ease to reach a satisfactory transfer agreement.

To prepare this group of best participants, in this work we have approached the problem by setting up a ranking to classify them, from the best to the worst user who can be invited and participate in a certain trading table. In order to create such a ranking, we use the classification algorithm known as probability estimation trees. In next sections, we start commenting supervised learning and explaining decision trees and then we focus in probability estimation.

Among all the aproximations to the supervised learning, which has been studied in prior sections, the so called methods based on models represent the learned knowledge (the model) in some richer language than that used to represent evidence from which the learning is done. These methods build explicit generalizations from the training examples, and then they can be applied to classify other non seen cases. On the other hand, methods based on instances represent the learned knowledge as a set of prototypes described in the same language used to represent the evidence. The prototype can be one of the examples or can be obtained from one or more examples [54].

An advantage of the methods based on models is that once the model is built there is no need to maintain the training examples to classify new ones. Also, models can obviate some attributes if they are not relevant for the classification.

Now are going to study some learning algorithms. According to the way in which the rules are generated, the algorithms are classified in:

- Splitting Algorithms: consist in recursively divide into subsets the initial data according to the value of one attribute until each subset contains cases belonging to a single class. This is the strategy followed by the decision trees induction.

- Algorithms based on Covering: this system consists in finding conditions of the rules that cover as many examples of a class and the least amount of the other classes.

In the next sections of this chapter, these algorithms and other approaches will be reviewed.

3.1 Decision Trees Induction

Decision Tree Induction is one of the simpliest learning methods. Decision trees are used in different tasks related to data mining, which were studied in prior chapters, like classification [20], regression [24] and clustering [25].

A decision tree is a graphical representation of a procedure to classify or evaluate a concept. They are an excellent support tool to help us to make appropriate choices among many possibilities.

Formally, a decision tree is a tree structure where each inner node represents a condition or test about an attribute and each branch that comes out of the node corresponds to a boolean condition over a possible value of the attribute considered in the node. Each branch leads to another decision node or to a leaf node. Leaves represent the class value for all the instances that reach that leaf. An unseen case is classified starting in the root node and applying the test of the attribute specified by this node and following the branch corresponding to the value that the attribute has in the example. The process is recursively repeated, and this way, a path is traced from the root to the appropriate leaf, and so the instance is classified with the label indicated in that leaf, determined by the set of training cases associated with it. If the class attribute is discrete, the decision tree is called classification tree, while if the class takes values in a continuous rank is called regression tree.

In order to understand this more clearly, we are going to see an example of the classification of *iris* plants (Iris dataset from UCI repository [26]). This problem consists in determine the kind of an iris plant (iris setosa, iris versicolor, iris virginica), according to the width and length of its petal and/or sepal. In the Figure 3.1 is pictured a classification decision tree for this problem.

Every path from the root node to a leaf correspond to a conjunction of tests about the attributes, while all the tree corresponds to a disjunction of those conjunctions. This can be expressed with a set of *if-then-else* rules, where the conditions are the conjunction of the tests and the leaves are the consequents.

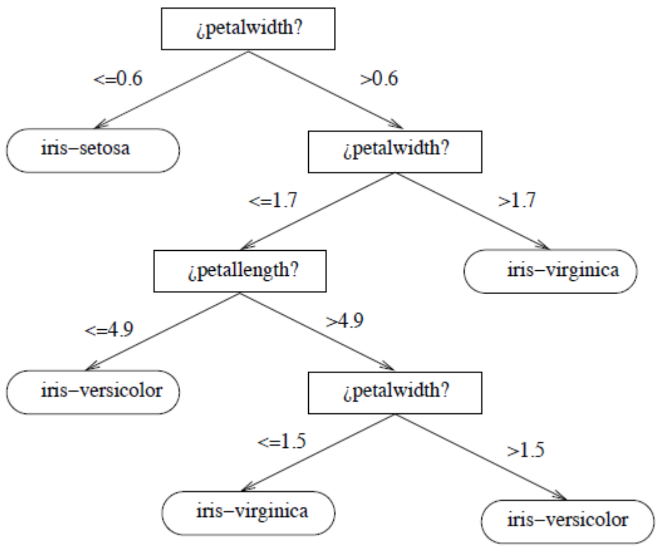

Figure 3.1: Classification decision tree for iris plants.

Below is represented the set of rules corresponding to the previous decision tree of the Figure 3.1.

```
IF petalwidth <= 0.6 THEN Iris-setosa
ELSE petalwidth > 0.6
| IF petalwidth <= 1.7
| | IF petallength <= 4.9 THEN Iris-versicolor
| | ELSE petallength > 4.9
| | | IF petalwidth <= 1.5 THEN Iris-virginica
| | | ELSE Iris-versicolor // petalwidth > 1.5
| ELSE Iris-virginica // petalwidth > 1.7
```

Most of the induction decision tree algorithms build the tree following the *top-down* system, from root to the leaves [20]. This approach applies a Divide-and-Conquer strategy. Initially we have the training set, at every turn the best attribute is selected, according to a certain criterion, as the test of the node and the examples are distributed among the descendant nodes. The process repeats until all the examples in a node are from the same class. This node becomes a leaf, assigned to that class.

Algorithms that split the instances from the top to the bottom are called partition algorithms. The two main issues in the decision trees induction algorithms are the generation of possible partitions and the criterion of selection of the best partition. These functions are, in fact, what distinguishes the existing partition algorithms, like CART [24], ID3 [20], C4.5 [23] or ASSISTANT [53]. The importance of these points is due to the fact that once a partition is chosen it can not be changed even though we realize that it was a wrong choice. Furthermore, more number of partitions could generate a more expressive decision tree, but also will increase the complexity of the algorithm. Because of that, most decision tree algorithms allow a very limited number of partitions. For instance, the C4.5 algorithm, developed from the previous algorithm ID3, only contains one type of partition for nominal attributes and one type for numeric attributes. These types of partitions are described as follows [54]:

- Partitions on nominal attributes: if a nominal attribute x_i has the values $\{v_1, v_2, ..., v_k\}$, can generate partitions like $xi = v_j$, with $1 \leq j \leq k$.
- Partitions on numeric attributes: the rank of a numeric attribute x_i splits into intervals, such that the partitions are $(x_i \leq v_c, x_i > v_c)$. v_c is some intermediate value between the maximum and the minimum value, and it is called cutoff point.

When facing the situation of choosing the best partition for a node, the partitions are ranked by heuristic functions and the first in the ranking is chosen. Most of the heuristics suggested are based on minimize the *entropy*.

According to [54], entropy is a measure used to characterize the degree of impurity of the examples E. In a classification problem with c classes, the entropy is defined as:

$$Entropy(E) = \sum_{i=1}^{c} -p_i \cdot \log p_i \tag{3.1}$$

where p_i is the probability of an example of belonging to a class i.

Another way to observe the effect of splitting the data by a particular attribute is the criterion called *Information Gain*, which calculates the expected reduction of the entropy by splitting the examples according to that attribute [54]. The $Gain(E, x)$ of a set of examples E in relation to the attribute x is:

$$Gain(E, x) = Entropy(E) - \sum_{v \in V(x)} p_v \cdot Entropy(E_v) \tag{3.2}$$

where $V(x)$ corresponds to the set of values of the attribute x, p_v is the probability of the attribute of taking $v(|E_v|/|E|)$ and E_v is the subset of E whose attribute x has the value v.

As it is said in [54], by maximizing the gain, it is expected to obtain smaller trees, by selecting in a given node the attribute which allows to distribute the examples in groups that are as homogeneous as possible. Nevertheless, gain does not reflect whether if the attribute x distributes the examples in a uniform way or not. In order to include this information, it has been proposed the criterion of *gain ratio* [20], which corresponds to a modification of the gain, defined as:

$$GainRatio(E,x) = \frac{Gain(E,x)}{SplitInformation(E,x)} \tag{3.3}$$

where E corresponds to a set of examples, x represents the selected attribute, and $SplitInformation(E,x)$ is the entropy of E with respect to x.

Another criterion is the *Gini* heuristic [24], which is similar to the gain but it replaces the entropy of a set of examples E by:

$$g(E) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3.4}$$

Therefore, the quality of the partitions inducted by an attribute x is settled as:

$$Gini(E,x) = g(E) - \sum_{v \in V(x)} p_v \cdot g(E_v) \tag{3.5}$$

3.2 Rules learning based on Covering

This rules perspective is somehow related to the previous one, since, as we have shown, decision trees can be viewed as a collection of propositional rules organized into a tree structure. In systems based on covering, the algorithms build the rules following a covering strategy, consisting in the generation of the rules one by one, and removing the examples covered by the new rule. So that, the process starts again with the remaining uncovered examples until it arrives to the final set of rules. These covering algorithms are based in the family of AQ algorithms [27, 28]. Another example of algorithm based on covering is the CN2 algorithm [44], [45]. CN2 redesigns AQ including pre-processes and post-processes to generalize the rules. FOIL [21] and FFOIL [22] are also based on covering.

There are two main differences between these algorithms and the decision trees recently explained. In a decision tree, the rules are mutually exclusive because of they come from different conditions. However, covering algorithms may overlap rules, since rules are generated independently. The second difference is that a new example will be classified on a single class in a decision tree but can be classifed on several distinct classes in a covering algorithm. Splitting algorithms like decision trees tend to be more efficient due to that each partition generates two or more rules. Even so, the capability of representation of both approaches is very similar.

3.3 Other applications for Decision Trees

Classification task is the best known application of decision trees, although it is not the only one. There are other variations of the decision trees in order to use them for tasks like regression or clustering. The main difference of a regression tree in front of a classification tree is that leaf nodes are labeled with real values. Even so, regression trees are built similarly to classification ones.

CART Algorithm [24] was one of the firsts decision tree learning systems, and implements both learning models for classification (predict nominal values) and for regression (predict numeric values). This way, the quality of a regression tree is measured in terms of a certain magnitude that we want to maximize. CART makes binary partitions in the atrributes, by assigning values of average and variance to each node, and then trying to select the partitions that lower the variance of child nodes.

As we have said, apart from regression, there are also algorithms for decision trees learning applied to clustering problems. The idea is to generate partitions in order to split into dense zones and sparse zones. The process continues until we get zones with very high density or very low density, which constitute the tree nodes. The method developed by [25] refines this idea, by considering all the examples from a class E (existing) and by adding fictional examples N (nonexistent) uniformly distributed in the space. Then, it is used a method of decision trees learning in order to classify the examples, having as a result, rules obtained for class E, which become the clusters.

3.4 Probability Estimation Trees

Finally, another area where decision trees can be applied is the probability estimation. As it happens in a classification problem, the examples have a discrete label, called class. The main difference is that probability estimators are based on assigning to each new case a probability distribution for all the classes at the leaf nodes, instead of determine which is the class of the case. They count the proportions from each class present at the leaf nodes, based on the training data, and generate a local maximum likelihood estimate.

A probability estimator is specially useful when we need a certain reliability in the prediction, when we want to combine predictions from several classifiers or when we are interested in making a ranking of the predictions, which is the case of our mWater problem. As it happens in mWater, there are problems in which ranking samples by the class probability are more useful than class predictions.

It is simple to adapt a classification tree to turn it into a probability estimator, by means of using class absolute frequencies of each leaf. For example, as seen in [2], if we have three classes: a, b, c. Each leaf node with a cardinality n (the total number of training examples that arrive to that node) has a certain number of examples corresponding to each class: n_a, n_b and n_c. The probability estimation is obtained by dividing each of those values by the total cardinality, namely: $p_a = n_a/n, p_b = n_b/n$ and $p_c = n_c/n$. This kind of decision trees is called *Probability Estimation Trees (PETs)* [30, 29].

Despite of the easiness of the conversion, the probability estimates obtained are poor when comparing to other probability estimators. However, there are some ways to change these results. [29] has improved the quality of PETs by applying frequency smoothing like Laplace correction, which has significatively improved the estimations. Similarly, at [30] the probabilities were improved using a ponderation of all the probabilities calculated through each branch, which was called *m-branch smoothing*. Furthermore, it has been observed that the use of transformations techniques like pruning usually harms the probability estimation. Consequently, trees unpruned produce the best results.

Thus, considering these particularities, decision trees are a good tool for the probability estimation.

Back into mWater problematique and focusing briefly in it, PETs appear to be a valid solution in order to create a best group of participants, due to the fact that we can leverage from the probabilities estimated by the decision tree and rank them. Thereby, we set a group of water users which show the best behaviour according to certain quality factor.

A recent work that have tried to exploit as well the use of PETs in order to generate rankings is [31]. This work faces a mailing campaign application, where a probability estimation model is learned and used to rank customers according to their probability of buying certain target product. From that customers ranking is selected the optimal cutoff that maximizes the overall benefits of the mailing campaign.

Moreover, an extension of PETs recently developed is [55], where it is presented a redefinition of PETs based on stochastic understanding of decision trees that follows the principle of attraction. The probability estimation tree learning method proposed there, is based on computing prototypes and applying an Inverse Square Law that uses the distance to the prototype and its mass in order to derive an attraction force which is then converted into a probability.

Chapter 4

Problem Definition

4.1 Case study

The problem we tackle in this thesis consists in, given a set of items, choose the best group of items according to the behaviour of previous groups. The historical data we have is about items and collections, both of them with several features. Each collection may have different number of items, and each item may have appeared in several collections. This way, we can talk about many-to-many relation between items and collections. The goodness of the collection not only depends on the goodness of each individual recommendation, but on the whole set, since in many cases there might be relations between the products.

Given this historical data, we are able to formalize the problem. For a new collection and a set of items, we need to select a number of items such that a certain utility function is maximized.

The case study consists in modelling a virtual water-rights market system, following the process described above. This system will be explained in the next sections. Within this water market context, participants will be the items and trading tables will correspond to the collections. This way, we have to choose the best k users that will participate in a certain trading table, which is characterized with some particular features. These best participants will be inferred according to a utility function that we want to maximize, in order to increase the benefits of the transaction. This group of best participants can be formed in different ways, approaches that we are going to detail deeply in next sections.

4.2 mWater

mWater is an application of a regulated open Multi-Agent System (MAS) [16, 18]. It uses intelligent agents to simulate a flexible water-rights market, including the model and simulation of the water-rights market itself, the basin, users, protocols, norms and grievance situations [17, 18].

In hydrological terms, a water market can be defined as an institutional, decentralized framework where users with water rights are allowed to voluntarily trade them, always fulfilling some pre-established norms, to other water users in exchange of some kind of compensation, economic or not [11].

The motivation of a system like mWater is due to the fact that water scarcity is becoming a major concern in most countries [16, 17, 18]. Particularly, it is specially problematic in dry climates which suffer from severe water shortages, such as the Mediterranean coast of Spain [33], where there is a high degree of public awareness of the main consequences of the scarcity and the need of fostering efficient use of water resources.

Water scarcity threatens the economic viability of current agricultural practices and also is likely to alter an already precarious balance among its different types of use. Countries in arid climates need to find better ways to manage their water. Factors like inneficiency and waste are luxuries that water scarce regions can no longer afford.

This way, more efficient uses of water may be achieved within an institutional framework where water rights may be exchanged more freely [12, 13, 11]. Big problem appears due to the fact that if farmers cannot sell their extra water allotment, and in consequence, receive an economical compensation, they have no incentive to use the allotment efficiently, and it may become wasteful [33]. Initially, the willingness of irrigators to buy or sell water depends on economic motivations, such as the difference between the price of water and net revenue each farmer expects to earn by irrigating.

However, it is not always a matter of price expectations, but also of regulation. Because of water's unique characteristics, it is essential to design appropiate water laws and regulate the users' actions, interactions and trades. This emphasis on regulatory aspects is due to the fact that the main objective of policy-makers is to achieve an adequate behaviour of users to ensure the success of the market [16, 18], and the best way to control this is by means of regulation.

Additionally, it is important to consider also the social perspective. The idea is not only to consider hydraulic factors, such as river basins, water demands or pumping flows, etc., but also different norms typology, human (mis)conducts, trust criteria and users willingness to agree on water-right trading, which may lead to a win-win situation in a more efficient use of water. This requires the use of intelligent agent technology, including trust, cooperation, argumentation and, in general, agreement technologies [18].

The idea is to implement such a market with a regulated open multi-agent system, which focuses on demand and, in particularly, on the type of regulatory and market mechanisms that foster an efficient use of water while preventing conflicts among parties [16, 18]. Also, one of the main goals of mWater is to be used as a simulator to assist in decision-taking processes of policy-makers. This way, it plays a vital role defining norms, agents behaviour and roles, and assess their impact in the market, thus enhancing the quality and applicability of its results as a decision support tool.

In such institutional framework we shall profit from agreement technologies in order to understand the behaviour of participating agents and the collective effects of their behaviour. Taking control of these agents actions and attempting to predict their future behaviour seems to be a useful tool to perform more efficient and beneficial transactions.

4.2.1 mWater Requirement Specification

As it is declared in [16], mWater will be a virtual market system [14, 15, 19], in which water right transfer agreements will be executed by autonomous normative entities. In this environment, different entities will get in contact with water right holders that are willing to transfer their rights. By means of mWater system they will be able to negotiate the terms and conditions of the transfer agreement, following the spanish National Hydrologic Plan laws. The correct execution of the water balanced distribution and usage will be assured by normative entities that will represent the Basin Administration in the mWater system.

The two main scenarios mWater will support are: water markets among water users of the same basin and water markets among users of different basins. Both are quite simple, and comprise the temporary transfer of water-right from one right-holder to another after reaching an agreement in terms of the rights to be transferred and the economic compensation associated [14, 15, 19]. These two scenarios are well detailed in [16]. In the first case, the stakeholders are formed by all the members of the basin together with the administrative organisms of the basin. Equally, when water users come from different basins, stakeholders are members and administrative organisms of the basins involved.

The two right-holders that finally come to an agreement are allowed by law to establish the economic compensation by means of a private agreement process. Also, when water requieres third parties' infrastructure, its use must be freely agreed by the parties: seller, buyer and infrastructure owners.

On the other hand, mWater system implements a MySQL database, which will be seen later. According to the mWater implementation [18], there are three views that comprise the basin, market and grievance structure. In the first view all the information about the nodes, connections, users, norms and water-right definition is modeled. The second view models the information related to the entire market, including trading tables and their protocols, the water rights to be traded, participants, agreements and contracts that can be signed. Finally, in the third view, is modeled the information about the legislation and conflicts that may appear after an agreement or contract and the mechanisms for solving them. Anyway, in this work we only focus in the market view, and its related tables.

4.2.2 Activities in the Market scenario

Here are described the different activities required for any of the aforementioned market scenarios, as it is defined by mWater documentation. For more information, see [16]. For our purposes, the activities have been grouped by their scope, and the scopes are ordered according to their appearance in the scenario. The scopes that have been considered are the following four: Registration, Offers and Demands, Transfer Agreements and Grievances.

- Registration

  The initial group of activities is related to the registration of all the necessary elements in the system. These are the beginning steps in order to fulfill the conditions required to create the scenario.

  The activities included are:

  - Admission and Registration of Water Users
  - Registration of a Buyer
  - Registration of a Seller
  - Group Formation

- Offers and Demands

  This scope combines the activities related to the offer and demand of water rights, before they are negotiated.

  - Publish Water-Right Offer
  - Publish Water-Right Demand
  - Withdraw Water-Right Offer
  - Withdraw Water-Right Demand
  - Query Water-Right offers
  - Query Water-Right demands

- Transfer Agreements

  Once the Water-Right offers and demands are public, the negotiation can start, and so, all the activities included in it. This scope groups the actions around a transfer agreement.

  - Negotiate Water-Right transfer agreement
  - Registration and publication of Water-Right transfer agreement
  - Query Water-Right transfer agreement
  - Authorization of Water-Right transfer agreement
  - Execution of Water-Right transfer agreement

- Grievances

  Last scope refers to the grievances mechanism. Water users are allowed to initiate a conflict resolution procedure about a transfer agreement. The activities in this group correspond to the typical stages on a conflict.

  - Allegation against Water-Right transfer agreement
  - Hearing of water dispute
  - Sanctioning offenses in a Water-Right transfer agreement
  - Expel Water user

Despite of the existence of these four groups of activities in the market scenario, this work only takes into consideration the two of them directly related to the trading mechanism, namely, Offers and Demands and Transfer Agreements.

If we focus in Offers and Demands scope and take a look to the activities that it involves, we can extract that they are all actions taken by a Water user, who wants to do something with a water right that owns, or a water right that he or she is willing to buy. The activities concerning to the publication of a water-right offer or demand constitute the trigger of a trading process, which will be included as an instance in the corresponding table of the database and, if it ends successfully (with an agreement), it will appear also in the table that contains all the transfer agreements.

But before this happens, it is necessary to develop the activities included in the Transfer Agreements scope. Here are defined the actions that can be done about a transfer agreement, and correspond to the subphases which an agreement can pass through. First step is the negotiation between the seller and the potential buyers (and perhaps, a third party affected). In a trading table is defined a certain number of invitations, which are sent to the wished participants to join the negotiation. Anyway, this does not mean that all the participants invited actually accept the invitation. The water users are free to decline an invitation to a trading table if they are not interested or if they consider that they can not extract anything positive from the trading process. During the negotiation procedure, and according to the settled protocol, the seller can hear the economic proposals of the different users and freely negotiate with them in order to achieve the best compensation in accordance to his individual needs. It is possible that a potential agreement requires the intervention of another water user, besides from seller and buyer. This user is called third party and it is due to the fact that to carry out the transfer agreement is needed his infrastructure or his authorization to use some area in his property. In this case, seller and buyer negotiate together with the third party to give him a compensation according to the help he has provided. Once the seller has accepted the offer of a buyer and both have established the conditions (and, if it is appropriate, with the third party), a transfer agreement is signed and remains pending of the authorization from the Basin Administration. If the transfer agreement conforms the normative regulation and there is no allegation (which would be administered in the Grievance area), the transfer agreement is authorized and will be executed in due course.

Chapter 5

Data description

5.1 mWater Database

Once we have seen a summary of the mWater characteristics, in this section we focus in the implementation of the system. The MySQL database implemented by mWater system is filled with 63 relational tables which store the historical data of the water market. As we have said, in this work we only care about the tables concerning the market view of the implementation.

In relation to this view of the database, the main tables we use for the application are the following:

- **TradingTable:** represents the tables opened, each one having an opening_user and a certain protocol_type, and with an opening and closing date. Its primary key is formed by a combination of configuration_id, trading_table_id and mwater_market_id. A market facilitator is able to open a new trading table whenever a new auction period starts or whenever a right-holder requests to trade a right [18]. In such a case, the right-holder (opening_user) chooses the negotiation protocol from a set of available ones. mWater system defines several scenes through which can pass a trading table, namely, Registration, Negotiation and Validation. This three-scene performative structure is handled by the system.

  It is important to note that the trading tables are stored here whether the trading agreement has been reached or not. Instead, only the trading tables where the seller has successfully transferred the water right will appear in the table explained below.

- **TransferAgreement:** represents an agreement of a transaction on a water right made by a buyer in a TradingTable, with a price accorded with the seller by means of the negotiation protocol set in the TradingTable. If the water-right transfer agreement conforms to the National Hydrological Plan (current Spanish law on water rights) normative regulation and there is no allegation, the agreement is authorized by the Basin Administration. Once a trading table has been succesfully validated reaching an agreement, a new instance appears in TransferAgreement table. Furthermore, if in the transfer agreement has participated a third party, he or she receives the agreed compensation.

- **User:** contains all the water users included in the database, whatever its function. Every user has a user_type which indicates his priority_value, and other individual properties, concerning to their behaviour in the negotiation. The table UserType adds more information about these properties.

  Although this table only stores users with water rights, mWater prototype specifies the following stakeholders, where the first point corresponds to the different types of user reflected in the database:

  - Water users: A water user is a water right holder. It could be:
    - Human water suppliers: a company that supplies water for human consumption.
    - Irrigator: owner of a property that uses water for agriculture.
    - Power companies: a company that produces electricity by means of hydraulic power.
    - Industries: other industrial uses.
    - Aquiculture Users: water use for growing plants in water dissolved nutrients.
    - Leisure Users: sport and leisure uses of water.
    - Navigation and Aquatic Transportation

    Water users can adopt four different roles in the market scenarios:
    - Water User: a water right-holder of the basin.
    - Buyer: a Water User that wants to transfer its right and or buy a transportation resource.
    - Seller: a Water User that wants to purchase rights and or sell a transportation resource.
    - Third Party: a Water User that can be affected by a water-right transfer agreement.

    mWater specification defines several attributes that describe a water user, which are: name of the farm, area of the farm, delimitation and boundary localization, district where it is settled, owner's name, participation in the assembly expenses, water volume and type of water.
  - Basin Adminsitration: assures an adequate management of the water requests, in order to promote saving and socio-economical efficiency in the different water uses.

  - Ministry of the Environment and Rural and Marine Affair: is the major authority on water uses regulation.
  - The Irrigators and other Uses Jury: is acquainted of the possible problems that may appear among the User Assembly members.

- **RecruitedParticipant:** represents all the water users who have participated in a TradingTable, whether they are the final buyer or not. For each participant there is an invitation_date and the number of his participations within the negotiation is recorded in the table.

  A water right-holder becomes a recruited participant when he or she accepts an invitation for a trading table. During the negotiation, every participant is allowed to interact and discuss offers for the water right. If he arrives to an agreement with the seller, the participant turns into buyer.

  There is another possibility for a recruited participant in a trading table to be a part of the transfer process, and it happens when the participant becomes a third party, as it is listed in User part. This may occur if the seller and the buyer need others' infrastructure to carry out the transfer.

- **WaterRight:** represents the water rights that will be traded in the trading tables, owned by a certain water user from the system. The water right is defined by: basin, water user, water volume, district where it is settled, time period and type of water. The type of water is determined by the type of user that owns it. The types of water defined in this table are: Aquiculture, Human, Energy, Industrial and Irrigation.

- **Resource:** represents resources that will be used in a TransferAgreement (RequiredResource) between an initial and final date and with an agreed price. The main resource in the system is obviously water, but there are different water sources which are also resources of the system, and the transportation infrastructure that connects different sources and allows water flow. These resources are the following: river, dam, underground water and transportation infrastructure. Nevertheless, in our implementation we only consider water as a resource.

5.2 Selection of Data

We have seen in the previous section the most important tables in the database for our purposes. Following the steps dictated by the KDD process, all these tables correspond to the first stage, the one dedicated to Integration and Data Collection. The mWater database collects all the information about past transactions performed within the framework. All this data needs to be analyzed in order to select only the information that is useful to our purposes. In this section we study how the minable view has been built, which corresponds to the second stage of the KDD process, which precisely outputs a minable view, ready to begin the process of data mining. The minable view generated will be used in the third stage of KDD to build a model that can make predictions and discover patterns in the data.

In order to selecting data to build the minable view, we design several SQL queries that help us to extract the information that we need from all the historical data stored in the water market system. The idea is to prepare three different datasets, containing the training set, the group of trading tables and the group of water users. For creating each dataset, it has been modeled a query, which extracts the required attributes and instances for each dataset.

Below it is presented as an example one of these three queries, specifically the one used for obtaining the trading tables dataset.

```
SELECT TradingTable.configuration_id, TransferAgreement.trading_table_id,
TransferAgreement.mwater_market, ‘opening_user‘, ‘protocol_type‘ ,
‘authorized_extraction_flow‘, ‘type_of_water‘,
DATEDIFF(WaterRight.initial_date_for_extraction, ’2000-01-01’) AS dias_inicio,
DATEDIFF( WaterRight.final_date_for_extraction, ’2000-01-01’) AS dias_fin

FROM  ‘TransferAgreement‘ ,  ‘TradingTable‘, ‘WaterRight‘

WHERE TransferAgreement.trading_table_id = TradingTable.trading_table_id
AND TransferAgreement.mwater_market = TradingTable.mwater_market
AND TradingTable.configuration_id = TransferAgreement.configuration_id
AND WaterRight.id=TransferAgreement.waterright_id

ORDER BY ‘trading_table_id‘, ‘configuration_id‘, ‘mwater_market‘
```

This SQL query extracts from the tables TransferAgreement, TradingTable and WaterRight the attributes that are needed in the minable view, such as the identifier of the trading table (composed necessarily of three elements), the water user who has opened the trading table, the type of the protocol used in the negotiation, the type and quantity of the water to be traded and the initial and final dates of the extraction right.

Next, we are taking a deeper look to the attributes extracted in the query. Regarding to the trading table identifier, as we just said, it is necessarily composed of three attributes. These attributes are the identifier of the trading table, the identifier of the configuration and the identifier of the water market which the trading table belongs to. This three-set will unequivocally stand for a single trading table.

The water right holder who requests to trade a right is the user who opens the table (actually, this task is developed by a market facilitator, but for our purposes and with the aim of a better understanding, the right-holder will receive the designation of opening user).

The type and the quantity of the water to be traded also appear in the minable view, as well as the type of negotiation protocol that is used in the trading process. Finally, the initial and final dates for the extraction right, which are settled by the seller, are also an aspect to be considered.

With regards to the other two SQL queries mentioned, the aspect of both queries is very similar to the showed prior. The one intended to contain the training set extracts the information about the ended training tables, wherever an agreement has been reached or not. This data set will be split in two parts, in order to use one of these parts for the training and one for the test. By making this partition, we guarantee the independence of the measure taken in the validation of the model obtained in the training.

The other data set obtained by a query is the list of all the water users included in the database who are able to participate in trading procedures. Each water user is accompanied by the statistics of its participations in the historical trade, that is, the number of times that he has participated in a trading table, what role he took and the number of succesful trades in which he has been involved, either as seller or buyer. These statistics have been extracted from the database by means of the definition of several aggregate functions, one for each value that we want to obtain. In a SQL query, the aggregate functions return a single value by performing a mathematical calculation on a set of some values of the table. All the aggregate functions used in the users query are quite simple, since they just need to make use of the COUNT function. This function returns the number of items in a group, in our case the number of rows in a certain table that satisfy the criteria specified in the WHERE clause. For instance, if we require the number of final transfer agreements that each water user has reached in the history of the market system, the query and its corresponding aggregate function would have the following aspect:

```
SELECT user AS user_id, count(*)

FROM  `TransferAgreement` ,  `TradingTable`, `RecruitedParticipant`

WHERE TransferAgreement.trading_table_id = TradingTable.trading_table_id
AND TransferAgreement.trading_table_id = RecruitedParticipant.trading_table
AND RecruitedParticipant.trading_table = TradingTable.trading_table_id
AND TransferAgreement.mwater_market = TradingTable.mwater_market
AND TransferAgreement.mwater_market = RecruitedParticipant.mwater_market
AND RecruitedParticipant.mwater_market = TradingTable.mwater_market
AND TradingTable.configuration_id = TransferAgreement.configuration_id
AND RecruitedParticipant.configuration_id = TransferAgreement.configuration_id
AND TradingTable.configuration_id = RecruitedParticipant.configuration_id
AND RecruitedParticipant.user=buyer_id

GROUP BY RecruitedParticipant.user
```

In this simple case, the COUNT function only cares about the number of rows that fulfill the conditions required in the WHERE clause, namely, if the identifier of a participant in the trading table corresponds to the identifier of the final buyer.

In the users query, the functions are a bit more complex. Below are presented one by one the four aggregate functions defined:

```
count(case when RecruitedParticipant.user=opening_user then 1 end)
as seller_participations
```

This aggregate function returns the total number of times each water user has opened a trading table in order to trade a water right he was willing to sell. We must highlight that the fact that a trading table has been opened does not mean that an agreement is reached in that table. Because of this, it has been created another aggregate function that shows the total number of sales that each opening user has achieved. Obviously, this number must be obligatorily less or equal to the number thrown by the previous function. The following query corresponds to that total number of sales. It makes an intersection between the tables TradingTable and TransferAgreement, as a result we get the quantity of trading tables that are not included in the TransferAgreement table, because there was no succesful agreement reached in those trading tables. Since this intersection shows the number of unsuccesful trading tables opened by each water user, we just need to substract this quantity to the total number of opened tables by each one, namely, the quantity obtained in the previous query.

```
SELECT count(*)

FROM TradingTable
LEFT OUTER JOIN TransferAgreement
ON TradingTable.configuration_id = TransferAgreement.configuration_id
AND TradingTable.mwater_market = TransferAgreement.mwater_market
AND TradingTable.trading_table_id = TransferAgreement.trading_table_id

WHERE TransferAgreement.configuration_id IS null
AND TransferAgreement.mwater_market IS null
AND TransferAgreement.trading_table_id IS null
```

On the other hand, the next two aggregate functions focus on the buying statistics. The function presented below makes the count of the total number of times each water user has participated in a trading table playing the role of a potential buyer (this does not mean that finally becomes the buyer). To count the participations of a water user as a potential buyer, the function calculates the difference between the total number of appearances of each user in a trading table and the number of times that he appears as a seller.

```
count(*) - count(case when RecruitedParticipant.user=opening_user then 1 end)
as buyer_participations
```

Similarly, in order to compute the number of final transfer agreements that each water user has obtained as a buyer in all the trading tables, the function pays attention to the correspondence between the identifier of the participant and the identifier of the user that has bought the water right in each table.

```
count(case when RecruitedParticipant.user=buyer_id then 1 end)
as num_agreements
```

Another remarkable function defined in the queries is DATEDIFF. This function returns the amount of time between two given dates. In the context of the system's database, the function is used to denote the difference between the initial and the final date specified for the right of the water extraction. The idea is to mark a previous reference date, from which we can place any day, so that the starting day for the extraction (and also the final date for extraction) appears as the interval between the reference date and that day, calculated in days. The functions are defined as following:

```
DATEDIFF(WaterRight.initial_date_for_extraction, '2000-01-01') AS dias_inicio,

DATEDIFF( WaterRight.final_date_for_extraction, '2000-01-01') AS dias_fin
```

Chapter 6

Making mWater recommendations based on probabilities

In this chapter, the approaches proposed for solving the problem are presented. Our application deals with the problem of assembling an optimal group of water users that achieve the best performance in a particular trading table. The basic idea is to observe the historical information stored in the market's database in order to study the past behaviour of the participants to successfully predict future behaviour under specific conditions. As we are saying along this thesis, the predictions of the future behaviour will be obtained by using the probabilities that can estimate a probability estimation tree. By means of taking the value thrown by the tree for each participant about the probability of becoming the final buyer, a ranking is generated. A ranking is a great tool in order to visualize in a clear way which are the best items of a list. In this particular scenario, the best items shape the ideal group of participants for a trading table.

This issue has been faced in different ways. In this chapter we present the best approaches that we have developed, those that produced the best performance.

All the implementations make use of the three datasets mentioned in the section 5.2. They coincide in the fact that they all split data in two parts, one for training and one for the test. This way, the evaluation of the model can be done over a set of data different from the set in which the model is learned.

Below are presented these approaches dealing with the problem of making the group of the best participants for a trading table.

6.1 Direct Ranking approach

In this simple approach, for each trading table considered we will obtain a ranking of the water users that can be invited as participants. These users are ranked according to their probability to get a transfer agreement during the negotiation with the seller who is trading his water-right. From this ranking, we offer to the water market system a recommended group of agents, formed by the k first classified users. k will correspond to the number of invited participants required by the mWater system.

How can we perform that? We start from the idea of having, on the one hand, a dataset with trading tables, and on the other hand, one dataset with all the existing participants and their attributes. The 75% of the trading tables dataset is taken for the training, and the remaining 25% is used for the test. The trading tables stored for the training include, besides of the main attributes that characterize a trading table and explained in the prior chapter, the information about the participants in each one, and an output attribute indicating whether the user has got a transfer agreement in the trading table or not. In the Table 6.1 is represented an extract of these trading tables used for the training. In order to better visualization of the data, we omit some columns of the table that have no relevance to the example. Each row stands for a trading table stored in the system's database and a water user who has participated in it. For each trading table it is indicated which was the agent who achieved the transfer agreement, in case that it is necessary (can exist trading tables without buyer).

configuration_id	trading_table_id	mwater_market	opening_user		user (id)		buyer
572	1	571	7	...	3	...	NO
572	1	571	7	...	4	...	NO
572	1	571	7	...	9	...	YES
587	1	586	5	...	9	...	NO
587	1	586	5	...	10	...	YES
1537	1	1536	4	...	3	...	YES
1537	1	1536	4	...	7	...	NO
1537	1	1536	4	...	8	...	NO
1537	1	1536	4	...	10	...	NO
1537	1	1536	4	...	11	...	NO
1952	1	1952	10	...	4	...	YES
1952	1	1952	10	...	7	...	NO
1952	1	1952	10	...	8	...	NO
1952	1	1952	10	...	11	...	NO
2251	3	2250	8	...	2	...	NO
2251	3	2250	8	...	9	...	NO
2251	3	2250	8	...	10	...	NO

Table 6.1: Trading tables in training dataset.

From this minable view, we would learn a model for the utility variable. The model is learned by means of a probability estimation tree, specifically the application works with a version of the classifier C4.5, called J4.8. In the next chapter are reviewed the details of the communication between our application and the environment used. This way, a C4.5 decision tree is generated from the training dataset. This model will be used to evaluate the test dataset and predict future buyers on the trading tables included in that dataset.

With respect to the test, for each trading table we analyze its performance with each single water user, namely, we create as many new table+participants as trading tables we have in the test dataset. As a result, for each resulting dataset (each individual trading table combined with all the potential participants), is generated a ranking of the participants that better performance offer in the trading table. This performance is determined by the probability of each participant of having an agreement in the conditions settled in the table. The probabilities are calculated by a probability estimation tree mentioned, the J4.8 classifier, which assigns to each unclassified instance (trading table + participant) a probability distribution for the output classes, yes/no in our application ('yes' stands for a user has reached an agreement on the trading table, if not, the label is 'no'). Because of we care about the possibility of a participant becoming a final buyer, we are interested in the 'yes' probability, so each instance is ranked according to it. It is generated a ranking for each trading table in the test dataset, this way each single trading table will have different appropriate participants according to its particular features. From each ranking of probabilities, a certain number of water users is chosen, according to the number of participants required in the trading table associated to the ranking. This group of selected participants will constitute the recommendation given to the water market system by our implementation.

6.2 Incremental Ranking approach

This approach is conceived as a redefinition and improvement of the previous one. It also aims to create a ranking of participants according to their 'yes' probabilities, but it takes into consideration more elements. We include some aggregate information about the historical behaviour of the participants in each trading table, with the purpose of refining the prediction of the model.

Following the steps detailed for the previous approach, the model is learned similarly, data splits in two parts and the same datasets are used. Up to this point, the steps followed are the same. The main difference arises when the ranking is generated, because we re-evaluate the model introducing several modifications. The first improvement is produced when the ranking for each trading table is available, so for each one we take the first classified of the ranking, namely, the instance who has obtained the highest probability for the 'yes' class, and keep this user apart as the first of the final group of recommended participants, that we will offer to the mWater system. This user is called the *winner*. In order to choose the next participant, we rebuild the classifier, but before rebuilding we introduce slight changes in the datasets that we are using. This means that this time the winner is removed from the users' dataset (this way that participant can not "win" again) and also we add an attribute in the training dataset which contains the information associated to the winner statistics. For each trading table of the training set, this new attribute indicates whether the winner had participated in it or not. The model is re-evaluated and a new probability estimation tree is generated. The test is evaluated similarly to the prior approach, and a new ranking is generated for each trading table of the test dataset. In the new rankings is obviously not included the winner of the previous iteration, and each ranking tends to be different from the ranking generated previously for the same instance. Choosing the top of the list again, we have the second participant for each trading table recommendation. Then we repeat the process iteratively until we obtain the required number of participants for the trading table.

Let us illustrate the process with an example. Suppose we have a test dataset containing all the unclassified trading tables combined with all the potential water users that can be invited to the negotiation process. Suppose also that the required number of invitations has been fixed on $k = 4$, so that in the incremental approach we will run the application until we get the 4 recommended participants. This means 4 iterations. For example, an instance of one of these trading tables is represented in the Table 6.2.

configuration_id	trading_table_id	mwater_market	opening_user	protocol	flow	type_of_water	dias_inicio	dias_fin
2217	1	2216	5	1	97	human	3653	5478

Table 6.2: Test instance of a trading table.

user	user_type	priority_value	seller_participations	buyer_participations	num_sales	num_agreements
2	5	6	232	645	133	143
3	4	7	210	660	109	162
4	6	5	170	678	80	247
6	6	5	169	700	93	260
7	6	5	208	635	121	125
8	6	5	187	696	92	203
9	7	4	150	663	101	122
10	5	6	134	744	50	241
11	4	7	151	730	66	154

Table 6.3: Test dataset of water users.

The potential participants are represented in the Table 6.3. Each user is accompanied by information about the past transactions in which he has participated, both as seller and as buyer. The model would predict, for each user, the probability of becoming a buyer in the trading table of Table 6.2, by means of achieving an agreement with the opening user of the trading table after a negotiation process. Then, these probabilities are ranked in order to choose the participant who occupies the top position of the ranking.

The classification obtained for the trading table after ranking the 'yes' probabilities takes the form pictured below. Assuming that the number of invitations required by the system is $k = 4$, the ranking only shows the top four participants.

Water User	Probability
10	60%
6	48,24%
4	48,24%
3	42,42%

Table 6.4: Ranking associated to the trading table (1st iteration).

Water User	Probability
6	85,81%
11	72,16%
8	50,77%
3	34,32 %

Table 6.5: Ranking associated to the trading table (2nd iteration).

Following the scheme described before, from this ranking we choose only the top of the list (that is, user with $id = 10$ in our example) and rebuild the model, introducing the modifications already mentioned. Now, the training set includes a new attribute indicating, for each trading table, whether the winner of the previous iteration had participated in it or not. The winner is also removed from the pool of water users (pictured in Table 6.3) that can be invited to the trading table. After the second iteration, the ranking generated is showed in Table 6.5. We should emphasize that the ranking after the second iteration has changed significantly from the first ranking showed in Table 6.4, due to the introduction of aggregate information which reflects the possible relationships and influences between the agents. It seems obvious that if the iterations after the first one do not change the results of it, it would be not necessary to develop this incremental approach.

Then, the process repeats until we obtain the required number for the group of recommended participants (in this particular example, until we make four iterations).

In order to understand this more clearly, let us see one more example of the execution of the application. In the Table 6.6 there are pictured ten trading tables predicted, showing the prediction made by the model and the buyer who achieved the transfer agreement in the real world.

configuration_id	trading_table_id	mwater_market	1st ranked (id)	Real case buyer (id)
757	2	756	10	3
311	3	310	4	4
573	2	572	8	6
704	2	703	6	8
505	2	504	3	8
1986	1	1985	10	10
1193	1	1192	8	8
858	1	857	6	10
1052	1	1051	4	2
236	3	235	2	10

Table 6.6: Example of the prediction of 10 trading tables.

In this incremental ranking approach, the results of this table would correspond to the first iteration. We can observe that each trading table has an independent prediction, namely, the user predicted as the winner in each one is not affected by the predictions of other trading tables. In the next chapter, we will study the different ways of how this results have been evaluated.

This method would improve over the previous one as more appropriate aggregate attributes are included, in such a way that this information captures global information about the trading tables and not only local information.

It is important to highlight also that in the previous approach the values of the probabilities obtained by the decision tree for each instance are independent from the other water users considered for the same trading table. Meanwhile, in the incremental approach the other users information is used to calculate the probability of each participant. This occurs after the first iteration, when we start to take into consideration who was the participant with the highest probability in the previous iterations and also it is included the information of the participation of that user in the trading tables of the training dataset. This way, the estimation of the future behaviour of a water user is conditioned by its relationship with the other users of the pool. The social aspect of a negotiation is an important issue to consider, since the relationships between real users can facilitate or lead to failure a trading operation. In the incremental method, we take into account the agents that worked together in past negotiations and then appreciate this information in order to predict the most suitable group of participants for each new trading table.

Related to that, our incremental approach has resemblances with the concept of *Structured Prediction*. This kind of problems deals with the task of predicting structured objects by means of learning a function $h : X \longrightarrow Y$ where Y corresponds to complex structured outputs. This output space Y decomposes over some smaller substructures and the different output values are related to each other. As it is commented in [68], these dependencies between the output units suggest that in each case it makes more sense to solve a single task of predicting structured outputs rather than a collection of independent classification problems.

The incremental approach defined is halfway between the structured prediction and the idea of simply rank the probabilities. As we have said, the resulting value of the probability of getting an agreement is affected by the dependence of the water user to the other users. In our particular case, the output space is not so much a structured prediction as a prediction depending on the relationships between the individual agents.

Chapter 7

Implementation and Experimental Evaluation

7.1 Communication with Weka

The classifiers generated in the approaches studied in the previous chapter have been learned by means of WEKA. It is a free distribution data mining tool, Java implemented. In this section it is presented a brief review of the main points of how the application interacts with Weka.

WEKA is a data mining environment capable of accessing to data. Because of that, we can load the data from files in a '.arff' format. This corresponds to a plane file structured in rows and columns. So that, we need to settle the datasets extracted from the database in that format, in order to become understandable by WEKA. This way, the application developed can read the datasets as external files and work with them by using WEKA libraries in a successful way.

WEKA provides us with many algorithms that we can apply to datasets. As it has been studied, the application works with probability estimation trees, in order to create a ranking of participants according to their probabilities. WEKA implements a version of C4.5 classifier, mentioned in the section 3.1 when talking about decision trees induction, called J4.8. This method builds a model from the training dataset with the form of a decision tree, which will be used in the test, for classifying new instances.

As regards to the test, WEKA evaluates the test dataset according to the previously trained set of instances. By means of the probability decision tree mentioned, for a new instance (a trading table combined with a potential participant) it will assign a probability of the participant of belonging to each class, 'yes' or 'no', according to his probability to become a buyer in the trading table.

7.2 Experimental Evaluation

Once the models are built, we test the validity of their predictions in the evaluation part. As it has been examined in the previous chapter, we analyze the performance of each new trading table combined with all the potential users that are able to join the negotiation that is about to start. For this purpose, we have used a test dataset filled with 382 instances, each one corresponding to an unlabeled trading table. This number of instances has been extracted from the total training dataset, consisting of 1562 trading tables. From this dataset we have taken the 25% (382 instances) for the test, and the remaining 1144 were used for the training. The results reviewed in this section are taken from the average of ten executions of each of the two approaches explained in the prior chapter.

This way, for each dataset trading table + users, it is calculated, by means of the probability estimation tree, the probability of each instance to belong to the class 'yes' and to the class 'no'. Then, a ranking is generated in order to observe which are the participants who have obtained the highest 'yes' probabilities. After generating the rankings for each trading table, and for each iteration in the case of the incremental ranking approach, we test their hit rate. Before studying the results, we are going to see how the evaluation is performed.

It have been used two ways of measuring the models. In both of them, the hit rate is obtained by observing how the trading table predicted was developed in the real world. The difference comes because in the first way, if the participant that was actually the final buyer of the water right in a specific trading table is the same as the one that our model has ranked on the top of the list, we conclude that the model has achieved a succesful prediction. Instead, the second way aims to see further than the user placed on the top of the list. This way to evaluate the predictions is to form, for each trading table, the pool of k participants required by the mWater system and to observe if the real world buyer is in that group of users recommended by the model. In the case of the incremental ranking approach, whenever an iteration is produced, one participant is picked for the group associated to each trading table. Meanwhile, in the direct ranking approach, the group is extracted directly from the ranking obtained for each trading table.

In the Table 6.6 of section 6.2 was showed an example of an execution of the incremental ranking approach, considering the first type of evaluation just commented. The total hit rate of the model corresponding to that example was 23.82%. This hit rate corresponds to the possibility of "guessing" the real buyer in the first iteration of the execution, but since the mWater system usually asks our application for a recommendation of a group of best agents, it seems more appropriate to measure the quality of the predictions in terms of a group of several users, as it is evaluated in the second way

configuration_id	trading_table_id	mwater_market	1st	2nd	3rd	4th	Real case buyer (id)
757	2	756	10	8	6	4	3
311	3	310	4	10	9	3	4
573	2	572	8	3	10	6	6
704	1	703	6	10	4	8	8
505	1	504	3	8	6	10	8
1986	1	1985	10	6	5	8	10
1193	1	1192	8	3	10	2	8
858	1	857	6	10	4	7	10
236	3	235	2	10	3	11	10
1052	1	1051	4	7	8	3	2

Table 7.1: Example of recommended groups of participants (Incremental ranking approach).

explained above. This way, considering $k = 4$, Table 7.1 completes the Table 6.6 by picturing an example of several groups of participants made for different trading tables. The right column contains the real buyer in each trading table. In this particular case, the total hit rate, considering a hit when the real buyer is contained in the group, reaches the 62.56%.

Meanwhile, the corresponding results of the direct ranking approach for the same trading tables of the previous table are represented in the Table 7.2. In this approach, the group of selected participants for a new trading table is taken directly from the top k users of the ranking associated to the trading table. The total hit rate in this example, also considering a hit when the real buyer is contained in the top k ranked agents, reaches the 54.45% (208 hits over 382 trading tables in the test dataset).

configuration_id	trading_table_id	mwater_market	1st	2nd	3rd	4th	Real case buyer (id)
757	2	756	10	9	4	2	3
311	3	310	8	10	3	11	4
573	2	572	2	8	10	6	6
704	1	703	8	10	6	4	8
505	1	504	8	6	3	11	8
1986	1	1985	10	6	4	8	10
1193	1	1192	8	10	6	7	8
858	1	857	4	10	6	3	10
236	3	235	5	9	11	8	10
1052	1	1051	8	4	3	10	2

Table 7.2: Example of groups of participants (Direct ranking approach).

In order to understand how the hit rate is calculated, the Table 7.3 shows the number of hits achieved by an execution example in each iteration of the incremental ranking approach. We need to keep in mind that in each iteracion, a hit is produced when the top agent of the ranking corresponds to the buyer in the real world case. Obviously, when a hit is produced in a certain trading table in the iteration n, the next iterations $(n+1, n+2...)$ of that particular table will not be considered as potential hits, because the top user (who coincides with the real case buyer) has been already removed from the pool of potential participants.

Iteration	1st	2nd	3rd	4th	Total
Hits	76	64	67	32	239

Table 7.3: Hits in each iteration (with $k = 4$).

In this case, the total number of hits of the table is 239. As the test dataset used for the example contained 382 trading tables, the total hit rate is 62.56%.

The next table summarizes the average hit rates of each approach after several executions of the application, considering the two ways mentioned of evaluating them.

Approach	First classified	Group of 4 agents
Direct Ranking Approach	19.11%	54.97%
Incremental Ranking Approach	23.82%	61.78%

Table 7.4: Summary of the resulting hit rates.

As we can observe, in both examples of Tables 7.1 and 7.2, and in the summary of Table 7.4 (and therefore, in the rest of the execution, since the examples pictured are a representative sample of the results), usually the real buyer is contained in the group of participants selected by the application. This way of evaluating, in contrast to the first one, which only cares about the first classified, seems to be a fairer measure of the validity of the prediction, since when a new trading table is opened within the framework, what we offer to the system is the recommendation of this group of agents, and if it is considered appropriate, those will be the participants invited to the negotiation process. The hit rates of 62% (at incremental ranking approach) and 55% (at direct ranking approach), constitute a good measure of the goodness of the application developed.

The Table 7.5 presents a compilation of the differences of the hit rates obtained according to the number of agents picked to form the recommended group (namely, the possible values of k).

Approach	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
Direct Ranking	19.11%	29.32%	43.2%	54.97%	66.23%	74.61%
Incremental Ranking	23.82%	38.22%	50.26%	61.78%	73.04%	83.78%

Table 7.5: Hit rates according to the value of k

We can observe the solid performance of the incremental method when the number of invited participants is increased to five or six, reaching a hit rate of nearly 84%, which could be considered a good estimation of the trading table development.

As we commented in the presentation of incremental ranking approach in section 6.2, this method is supposed to improve the direct ranking approach by capturing aggregate information about the agents who participate in the negotation process, as the relationships between the users that can affect the trading. Thanks to the execution of several iterations, the probabilities of each participant of getting an agreement in a trading table are recalculated in the $n > 1$ iterations, based on the relationships and interactions between individual users. After seeing the results, we can conclude that this improvement actually happens, since most of the times the execution of the incremental ranking approach throws a higher hit rate than the other approach.

Conclusions

In this thesis we have presented an approach based on automatic learning in order to elaborate trading tables within the framework of a negotiation process. We have applied the methods developed to a particular case: a water market system. It have been proposed two methods based on computing the probabilities of each agent of buying a water right, by means of taking into consideration the information about their trading behaviour in past negotiation processes.

The process that leads to the need of the proposed methods works as it follows: the virtual market receives the demand of opening a new trading table in order to trade a water right by a water user who is interested in temporarily transfering its right. Then, the system requires to our application of a recommendation of the best group of agents to join the negotiation in order to achieve the most satisfactory agreement. At this point, the methods developed prepare that recommended group according to the different characteristics of each trading table, the agents and the behaviour observed in past negotiations within the same framework.

The groups are formed using the probabilities thrown by a classification algorithm of a probability estimation tree. Each probability stands for the likelihood of an agent of becoming the final buyer of the water right traded. The agents with the highest probabilities are picked up to concoct the recommended group.

After studying the results reviewed in section 7.2, it can be highlighted that the incremental ranking approach throws nearly a 75% of hits, when the trading tables are composed of five agents, and nearly 84% when six agents are required. This hit rate overcomes the outcomes of the direct ranking approach. This improvement is due to the fact that the incremental method aims to include aggregate information about the historical behaviour of the agents, and also incorporate the social aspect of the negotiation, by means of considering the relationships between the individual participants. The inclusion of such aspects makes the incremental ranking approach more sensitive to the details of interactions between agents that could decant the negotiation towards a specific user.

On the other hand, in a case of a trading table where the application predicts a group of participants in which the real case buyer is missing, it would be a very interesting issue the possibility of checking how this group handles the negotiation and if they achieve a transfer agreement that is more beneficial for the parties involved than the original agreement that the real buyer reached. This kind of evaluation of the predictor could not be performed because of several difficulties (beyond our control) in the simulation tool used to simulate the whole water market system. However, it is a very interesting aspect to consider in possible future improvements of the work developed in this thesis.

Bibliography

[1] Clark, P.; Boswell, R.: Data Mining. *Practical Machine Learning Tools and Techniques with Java Implementations*, Morgan Kaufmann Publishers, 2000.

[2] Hernández, J., Ramírez, M.J., Ferri, C.: *Introducción a la Minería de Datos*, Pearson Prentice Hall, 2004.

[3] BakIr, G., Hofmann, T., Scholkopf, B., Smola, A.J., Taskar, B.: *Predicting structured data*, The MIT Press, 2007.

[4] Ricci, E., De Bie, T., Cristianini, N.: *Magic moments for structured output prediction*, Journal of Machine Learning Research, 9:2803-2846, 2008.

[5] Joachims, T., Hofmann, T., Yisong Yue, Chun-Nam Yu: *Predicting structured objects with support vector machines*, Communications of the ACM, Research Highlight, 52(11):97-104, November 2009.

[6] Vembu, S.: *Learning to Predict Combinatorial Structures*, PhD thesis, Department of Computer Science, University of Bonn, 2009.

[7] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: *From Data Mining to Knowledge Discovery: An Overview*, Advances in Knowledge Discovery and Data Mining, 1-34, AAAI/MIT Press 1996.

[8] Two Crows Corporation: *Introduction to Data Mining and Knowledge Discovery*, 1999.

[9] Mitchell, T. M.: *Machine Learning*, McGraw-Hill, 1997.

[10] Solomonoff, R.: *A Formal Theory of Inductive Inference, Part I*, Information and Control, Part I: Vol 7, No. 1, 1-22, March 1964.

[11] Thobani, M.: *Formal water markets: Why, when and how to introduce tradable water rights*, The World Bank Research Observer, 12(2):161179, 1997.

[12] Calatrava, J.: *Mercados y bancos de agua en España*, Agricultura Familiar en España, 99105, 2006.

[13] Riesgo, L., Gómez Limón, J.A.: *Mercados del agua. análisis de las opciones elegidas para su aplicación en España*, IV Congreso Nacional de Economía Agraria, 2004.

[14] Botti, V., Garrido, A., Giret, A., Noriega, P.: *Managing water demand as a regulated open MAS*, Proceedings of the MALLOW Workshop on Coordination, Organization, Institutions and Norms in agent systems in on-line communities (COIN@MALLOW09), 108109, 2009.

[15] Botti, V., Garrido, A., Giret, A., Igual, F., Noriega, P.: *On the design of mWater: a case study for agreement technologies*, Proceedings of the 7th European Workshop on Multi-Agent Systems (EUMAS 2009), 2009.

[16] Botti, V., Garrido, A., Giret, A., Igual, F., Noriega, P.: *mWater Analysis and Design*, AT/2008/Deliverable D8.2.1/v0.7, 2009.

[17] Botti, V., Criado, N., Garrido, A., Garrido, J.A., Giret, A., Noriega, P.: *mWater prototype review #2 analysis and design*, AT/2008/Deliverable D8.2.1.P2/v0.5, 2010.

[18] Botti, V., Garrido, A., Gimeno, J.A., Giret, A., Noriega, P.: *mWater Analysis and Design*, AT/2008/Deliverable D8.2.3/v0.1, 2010.

[19] Garrido, A., Giret, A., Noriega, P.: *mWater: a sandbox for Agreement Technologies*, Proceedings of the XII Congreso Catalán de Inteligencia Artificial (CCIA09), 252261, 2009.

[20] Quinlan, J.R.: *Induction of decision trees*, Machine Learning, 1(1):81-106, 1986.

[21] Quinlan, J.R.: *Learning logical definitions from Relations*, Machine Learning, 5:239-266, 1990.

[22] Quinlan, J.R.: *Learning first-order definitions from relations*, Machine Learning, 5(3), 239-266, 1996.

[23] Quinlan, J.R.: *C4.5. Programs for Machine Learning*, San Francisco, Morgan Kaufmann, 1993.

[24] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*, Wadsworth and Brooks, Monterey, CA, 1984.

[25] Liu, B., Xia, Y., Yu, P.S.: *Clustering through decision tree construction*, Proceedings of the ninth international conference on Information and knowledge management, 20-29, 2000.

[26] Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: *UCI repository of machine learning databases*, 1998.

[27] Michalski, R., Larson, J.: *Incremental generation of VL1 hypotheses: The underlying methodology and the description of program AQ11*, ISG 83-5, Computer Science Department, University of Illinois at Urbana-Champaign, 1980.

[28] Michalski, R., Mozetic, I., Hong, J., Lavrac, N.: *The AQ15 inductive learning system: an overview and experiments*, Proceedings of IMAL 1986, Orsay 1986. Université de Paris-Sud.

[29] Provost, F.J., Domingos, P.: *Tree induction for probability-based ranking*, Machine Learning, 52(3):199-215, 2003.

[30] Ferri, C., Flach, P.A., Hernández, J.: *Improving the AUC of probabilistic estimation trees*, Machine learning: ECML 2003, 14th European Conference on Machine Learning, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings 121-132, 2003.

[31] Bella, A., Ferri, C., Hernández, J., Ramírez, M.J.: *Joint Cutoff Probabilistic Estimation Using Simulation: A Mailing Campaign Application*, Intelligent Data Engineering and Automated Learning - IDEAL 2007, Lecture Notes in Computer Science 4881, 609-619, 2007.

[32] Estruch, V.: *Bridging the Gap between Distance and Generalisation: Symbolic Learning in Metric Spaces*, PhD thesis, Departament de Sistemes Informàtics i Computació, Universitat Politècnica de València, 2008.

[33] Honey-Roses, J.: *Assessing the potential of water trading in Spain*, ENR 319 Advanced International Environmental Economics, Professor Theo Panayotou at Harvards John F. Kennedy School of Government, 2007.

[34] Mashayekhy, L., Nematbakhsh, M.A., Ladani, B.T.: *E-negotiation model based on Data Mining*, IADIS International Conference e-Commerce, 2006.

[35] Lai, H., Doong, H., Kao, C., Kersten, G.E.: *Understanding Behavior and Perception of Negotiators from Their Strategies*, Hawaii International Conference on System Science, Hawaii, 2006.

[36] Coehoorn, R.M., Jennings, N.R.: *Learning an opponents preferences to make effective multi-issue negotiation tradeoffs*, Proceedings of the 6th International Conference on Electronic Commerce (ICEC2004), Delft, The Netherlands, 5968, 2004.

[37] Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M.: *Automated Negotiation: Prospects, Methods and Challenges*, Int Journal of Group Decision and Negotiation, 2000.

[38] Jennings, N.R.: *On Agent-Based Software Engineering*, Artificial Intelligence, 117 (2) 277-296, 2000.

[39] Wooldridge, M.: *Agent-based software engineering*, IEE Proc Software Engineering 144 (1) 26-37, 1997.

[40] Nilsson, N.J.: *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, 1998.

[41] Sandholm, T.W.: *Distributed Rational Decision Making*, Multiagent Systems, MIT Press, 201-258, 1999.

[42] Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: *A classification scheme for negotiation in electronic commerce*, Agent-Mediated Electronic Commerce: A European Perspective, Springer Verlag, 19-33, 2000.

[43] Rodríguez-Aguilar, J.A., Martín, F.J., Noriega, P., García, P, Sierra, C.: *Towards a tesbed for trading agents in electronic auction markets*, AI Communications 11 5-19, 1998.

[44] Clark, P., Niblett, T.: *The CN2 induction algorithm*, Machine Learning, 3(4):261-283, 1989.

[45] Clark, P., Boswell, R.: *Rule induction with CN2: Some recent improvements*, Proceedings of the Fifth European Working Session on Learning, 151-163, Springer, Berlin, 1991.

[46] Rosenschein, J.S., Zlotkin, G.: *Rules of encounter*, MIT Press, 1994.

[47] Zeng, D., Sycara, K.: *How can an agent learn to negotiate?*, in J. Mller, M. Wooldridge and N. R. Jennings, Intelligent Agents III, 233-244, Springer Verlag, 1997.

[48] Faratin, P., Sierra, C., Jennings, N.R.: *Negotiation Decision Functions for Autonomous Agents*, Int. Journal of Robotics and Autonomous Systems 24 (3-4) 159-182, 1998.

[49] Faratin, P., Sierra, C., Jennings, N.R., Buckle, P.: *Designing Responsive and Deliberative Automated Negotiators*, Proceedings AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, Orlando, FL, 12-18, 1999.

[50] Faratin, P., Sierra, C., Jennings, N.R.: *Using Similarity Criteria to Make Negotiation Trade-Offs*, Proceedings 4th Int. Conf on Multi-Agent Systems, Boston, USA, 119-126, 2000.

[51] Parsons, S., Jenning, N.R.: *Negotiation Through ArgumentationA Preliminary Report*, Proceedings 2nd Int. Conf. on Multi-Agent Systems, Kyoto, Japan, 267-274, 1996.

[52] Parsons, S., Sierra, C., Jennings, N.R.: *Agents that reason and negotiate by arguing*, Journal of Logic and Computation 8 (3) 261-292, 1998.

[53] Cestnik, G., Kononenko, I., Bratko, I.: *Assistant 86: A knowledge acquisition tool for sophisticated users*, Progress in Machine Learning, Sigma Press, 1987.

[54] Palma, J.T., Marín, R.: *Inteligencia Atificial: Técnicas, métodos y aplicaciones*, McGraw Hll, 2008.

[55] Martínez, F., Estruch, V., Ferri, C., Hernández, J., Ramírez, M.J.: Newton Trees, AI 2010: Advances in Artificial Intelligence, Proceedings 23rd Australasian Joint Conference, Adelaide, Australia, 2010.

[56] Duda, R., Hart, P.E.: Pattern classification and scene analysis, John Wiley and Sons, 1973.

[57] Langley, P., Iba, W., Thompson, K.: *An analysis of bayesian Classifiers*, Proceedings of the 10th National Conference on Artificial Intelligence, 223–223, 1992.

[58] Michie, D., Spiegelhalter, D., Taylor, C.C.: *Machine learning, neural and statistical classification*, Ellis Horwood, 1994.

[59] Dempster, A., Laird, N., Rubin, D.: *Maximum Likelihood from incomplete data via EM Algorithm*, Journal of the Royal Statistical Society. 9:1-38, 1977.

[60] McLachan, G.J., Krishnan, T.: *The EM Algorithm and Extensions*, Wiley, 1996.

[61] Friedman, N., Geiger, D., Goldszmidt, M.: *Bayesian netwroks classifiers*, Machine Learning, 29:131-163, 1997.

[62] Rumelhart, D.E., Hintont, G.E., Williams, R.J.: *Learning representations by back-propagating errors*, Nature vol 323, 533–536, 1986.

[63] Hebb, D.O.: *The Organisation of Behaviour*, Wiley, 1949.

[64] Kohonen, T.: *Self-Organising Maps*, Springer, 1995.

[65] Kohonen, T.: *Learning vector quantization for pattern recognition*, Helsinki university of technology, department of technical physics, Technical report, TKK-F-A601, 1986.

[66] Moody, J., Darken, C.: *Fast learning in networks of locally tuned processing units*, Neural computation, 1, 281-294, 1989.

[67] Cover, T.M., Hart, P.E.: *Nearest neighbour pattern classification*, IEEE Trans. Info. Theory, IT-13, 21-27, January 1967.

[68] Lampert, C., Blaschko, M.: *Structured prediction by joint kernel support estimation*, Machine Learning 77: 249269, 2009.