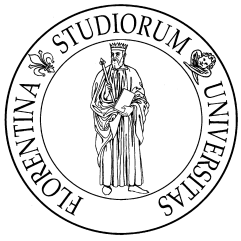
UNIVERSITÀ DEGLI STUDI DI FIRENZE
Dipartimento di Sistemi e Informatica
Dottorato di Ricerca in Informatica e Applicazioni
XXV Ciclo



DATA MINING MODELS FOR STUDENT DATABASES

RENZA CAMPAGNI

Supervisor: *prof. Donatella Merlini*

PhD Coordinator: *prof. Rosario Pugliese*

December, 2012

To my family:
Maurizio, Laura and Gloria

ACKNOWLEDGEMENTS

During the last three years I have been very lucky; I had a job that I really liked but mostly I met people who gave me so much, from whom I have learned so many things.

First, I would like to thank people of my thesis committee, Bruno, Dino, Donatella, Renzo and the PhD coordinators Rocco and Rosario.

A particular thanks to Cecilia and Pierluigi: this thesis would not have been possible without their presence, such as colleagues and friends.

Thanks in advance to Professors Paolo Mancarella and Toon Calders, who kindly accepted to read this thesis.

Thanks to past and current PhD students, in particular to Leonardo and Andrea; thanks to all teachers of department, in particular Betti, Michele B., Michele L., Pierluigi and Rosario.

A great thanks to Cecilia, Donatella and Renzo who welcomed me like an old friend and colleague.

Dino and Fosca have taught me all of the subject of this thesis; a particular thanks to them who have been for me also real friends.

I have to thank my parents who are gone: I owe to them the nature that made me making this wonderful experience at my no longer young age!

CONTENTS

1 Introduction 5
2 Data organization 11
2.1 Data warehouse 12
2.2 Types of data 15
3 Data Mining techniques 19
3.1 Supervised and unsupervised learning 21
3.2 Data mining tasks 22
3.3 Clustering analysis 23
3.3.1 `K-means` clustering 24
3.3.2 `DBSCAN` clustering 28
3.4 Analysis based on classification 29
3.4.1 Decision trees 30
3.4.2 Hunt's Algorithm 30
3.5 Association analysis 32
3.5.1 Association rules 32
3.6 Frequent pattern or sequential pattern analysis 35
3.6.1 SPAM 38
3.6.2 CloSpan 39
4 Data mining on student careers 41
4.1 Clustering model 41
4.2 Sequential pattern model 46
4.2.1 The methodology 46
4.2.2 Clustering on sequential patterns 50
4.3 Classification model 51
5 The case study 53
5.1 Context of analysis 53
5.2 Data preprocessing 56
5.3 Applying the clustering model 62
5.3.1 Extending clustering by applying classification model 77
5.4 Applying sequential pattern model 80
5.4.1 Tests and results on students enrolled in 2001-2004 of Curriculum 1 83
5.4.2 Tests and results on all students enrolled in 2001-2007 87
5.5 Analysis of delays distributions 90
5.5.1 Analysis of the real case 91
6 Conclusions and future analysis 99
6.1 The evolution of the C. S. degree at the University of Florence 102
7 Appendix A 105

8 Appendix B 111

1

INTRODUCTION

Rapid advances in data collection and storage technology have enabled organizations to accumulate a vast amount of data and often tools for traditional data analysis are not sufficiently efficient to produce significant results. Data mining is a data analysis methodology used to identify hidden patterns in a large data set. This technology blends traditional data analysis methods with sophisticated algorithms for processing large volumes of data; it has been successfully used in different areas including the educational environment.

Educational data mining (EDM) [9, 43, 59, 62] is an emerging and interesting research area that produces useful, previously unknown regularities from educational databases for better understanding and improving the educational performance and assessment of the student learning process (see [78] for a detailed description of the state of the art in this context). It exploits statistical, machine learning and data mining algorithms over the different types of educational data. Its main objective is to analyze these types of data in order to resolve educational research issues. EDM is concerned with developing methods to explore the unique type of data in educational settings and, using these methods, to better understand students and the settings in which they learn [8]. On one hand, the increase both in instrumental educational software as well as state databases of students information have created large repositories of data reflecting how students learn [57]. On the other hand, the use of Internet in education has created a new context, known as e-learning or web-based education, in which large amounts of information about teaching-learning interaction are endlessly generated and ubiquitously available [25]. All this information provides a gold mine of educational data [66]. EDM seeks to use these data repositories to better understand learners and learning and to develop computational approaches that combine data and theory to transform practice to benefit learners. EDM is a research area that involves researchers all over the world from different and related research areas, which are:

1. Offline education, based on face-to-face contact, that try to transmit knowledge and skills and also study psychologically on how humans learn. Psychometrics and statistical techniques have been applied to data, like student's behavior/performance, curriculum, etc., that was gathered in classroom environments.

2. E-learning and learning management system (LMS). E-learning provides online instruction, and LMS also provides communication, collaboration, administration, and reporting tools. Web mining (WM) techniques have

been applied to students data stored by these systems in log files and databases.

3. Intelligent tutoring system (ITS) and adaptive educational hypermedia system (AEHS) are an alternative to the just-put-it-on-the-web approach, trying to adapt teaching to the needs of each particular student. Data mining has been applied to data picked up by these systems, such as log files, user models, and so on.

The EDM process converts raw data coming from educational systems into useful information that could potentially have a great impact on educational research and practice. This process does not differ much from other application areas of data mining, like business, genetics, medicine, etc., because it follows the same steps as the general DM process [79]: preprocessing, data mining and postprocessing, such as we present in Chapter 3 by introducing the knowledge discovery in databases process (KDD).

In this thesis we deal with traditional universities data, that is, record data containing information about students which are usually only treated individually and for official purposes. However, if properly analyzed, these data could be used by the academic organizations to understand the behavior of students, how it is illustrated in [21].

Over the years, several data mining models have been proposed and implemented to analyze the performance of students. For example, in [30, 31], a model is proposed which presents the advantages of data mining technology in higher educational systems; the authors give a sort of road map to assist the institutions to identify the ways to improve their processes. In [28], the authors illustrate a classification model to investigate the profile of students which most likely leave university without ending their career. In particular, they use some classification algorithms implemented in the `WEKA` system [94]. In [29] a framework is proposed for mining educational data using association rules and, in [42], in order to explore the factors having impact on the success of university students, a system based on the decision tree classification technique is presented.

Data mining techniques have also been applied in computer-based and web-based educational systems (see, e.g., [77, 80]). The existing literature about the use of data mining in educational systems is mainly concerned with techniques such as clustering, classification and association rules (see, e.g., [29, 92, 94]).

In particular in this thesis we deal with data about graduated students. To this purpose we introduce the concept of *ideal career* corresponding to the career of a student who has given every examination just after the end of the corresponding course, without delay, and propose a data mining methodology to study the student behavior by comparing student careers with the ideal one.

- In a first approach by using the *clustering* analysis [22], we represent a career, also called *path*, as a trajectory of points in the plane, where the x-axis is for time and the y-axis is for exam. In particular, the ideal career is defined by a sequence of points $\tau_{\mathcal{J}} = ((0, e_0), (1, e_1), (2, e_2), \cdots, (n, e_n), (n+1, e_{n+1}))$, where e_i is an exam identifier and i its position in the career. The position $i = 0$ denotes the starting point of the career while $i = n+1$ corresponds to the final examination given last by all students. This particular career can be represented by the bisecting line of the first quadrant. We precise that i represents the order of exams, that is, two students with the same exam at position i, could have taken it in two different dates: what is common to the two students is that they have the exam in the same position in the career. The career of a generic student $\mathcal{J}$ is then represented by a broken line, corresponding to the sequence of points $t_{\mathcal{J}} = ((0, e_{\mathcal{J}_0}), (1, e_{\mathcal{J}_1}), (2, e_{\mathcal{J}_2}), \cdots, (n, e_{\mathcal{J}_n}), (n+1, e_{\mathcal{J}_{n+1}}))$, where $e_{\mathcal{J}_i}$ is the identifier of the exam given by student $\mathcal{J}$ at time i. A sample is illustrated in Figure 1.1, where we consider a career of 5 exams and where the green line represents the ideal career. We then compute the distance between

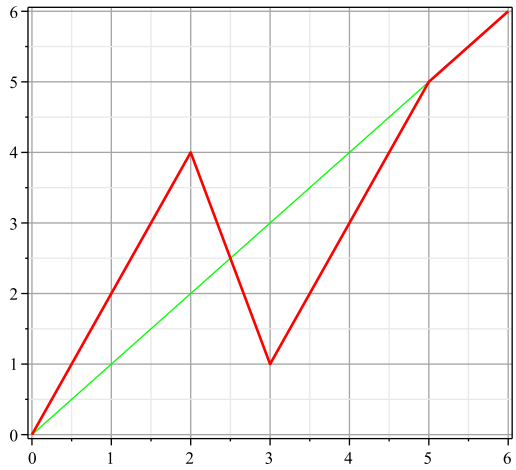


Figure 1.1: The career $(2, 4, 1, 3, 5)$.

a generic career $t_{\mathcal{J}}$ and $\tau_{\mathcal{J}}$, by using two different approaches. The first one uses the *Bubblesort distance*, which corresponds to the number of inversions in the permutation corresponding to $t_{\mathcal{J}}$. The second approach is based on the computation of the *area* between the lines corresponding to $t_{\mathcal{J}}$ and $\tau_{\mathcal{J}}$. We finally put into correlation these two distances with the complete data about students and perform data mining analysis based on clustering techniques.

- A different analysis is introduced by using the *sequential pattern* techniques [24]; we propose a methodology that studies the career of students

and compares one another. This technique has been introduced in [3] and has become an important method in data mining (see, e.g., [92]). Sequential pattern analysis aims to find relationships between occurrences of sequential events, that is, to find if any specific order of the occurrences exists. We can find an interesting approach by this technique in [88]. In this thesis we consider as events the exams taken by a student; the temporal information is the *semester* in which the exam has been taken or the *delay* with which it has been taken.

We study an organization of the university which allows students to take an exam in different sessions after the end of the course, as in Italy. In particular, italian university students can take an exam also with a large delay, they can try to take an exam several times, without restrictions in the number of times, and they can decide to refuse a vote of an examination and give it other times. Therefore, the career of a student can be seen as a sequence $e_1 e_2 \dots e_n$ of n exams, where e_k precedes e_j in the sequence if e_k has been taken at the same time or before e_j. The temporal information allows us to see the career of a student as a sequence $\langle i_1 i_2 \dots i_m \rangle$ where each element i_j is a collection of one or more exams taken in the same semester or having the same delay. If we use the semester as temporal information, m indicates the number of semesters in which a student takes exams; if we use the delay, m indicates the maximum number of delays, in semesters, with which a student takes one or more exams. By analyzing the sequential patterns, we can explain some behaviors which may seem counterintuitive, e.g., course x is scheduled before course y while many students take exam y before x.

Such information may be helpful for changing courses schedule or to find out those courses whose exams are considered difficult by the students and thus could give ideas for reorganizing curricula. Moreover, sequential patterns can be used to refine the analysis of students by introducing in the database the Boolean information about the most significant patterns, according to the fact that a student verifies the pattern or not. This new information can be explored by using clustering techniques in order to understand if students satisfying the patterns have some common characteristics.

- We deepen our analysis based on clustering and Bubblesort distance, by extending the model through the technique that uses *decision trees* [23]. To this purpose, we add to the database some new attributes, for example the *Bubblesort class* which labels the students into K different ways, according to the ranges of values of Bubblesort distance in the K clusters previously found, or the *Grade class* which labels the students into K_1 different ways, according to the large or small final grade of students. These new attributes can be used to classify students, for example by using a

classification algorithm. The aim is to classify students as talented or not, and find the attributes which most influence their career. The greater the database and the information in it are , the more accurate the model based on this technique will result.

We present our analysis in a real case, by applying our models on a student database of University of Florence (Italy).

We can also analyze the career of university students from another point of view, that is, we can study the perspective of each course, by analyzing the distribution of students with respect to the delay with which they take an examination, to discover common characteristics between two or more courses. This is done in terms of Poisson distributions. Usually, *good* students try to pass early every exam, but *not so good* students prefer to postpone many exams, especially if they are considered too difficult or too technical. This observation urged us to study the delay distribution of every exam in the hypothesis that it is a good parameter for classifying students and/or courses.

After an introduction about the context of the thesis, Chapter 2 provides a basic background on modern organization of databases, with some references to datawarehouse techniques and to different types of data.

In Chapter 3 we present some brief categorizations of data mining techniques and a detailed description of those used in the thesis.

Chapter 4 presents our methodology based on several data mining techniques; it is presented on the particular context of analysis, that is, the Italian Universities organizations and the information systems that are their foundations.

In Chapter 5 we illustrate the data mining methodology in a real case, that is, the student database of Computer Science degree at the University of Florence (Italy), relative to academic years from 2001-2002 to 2007-2008. We apply the different proposed models, from clustering to frequent patterns and classification, besides the statistical analysis of the delay distributions. For each considered data set we illustrate the results also by discussing their graphical representations. In the period under analysis, the Computer Science degree had different organizations: during the academic years from 2001-2002 to 2003-2004 each student could choose several exams among five different curricula, while during the other years students could choose between two curricula. For this reason, we apply our models to several dataset. In particular, for the clustering model, we consider three different datasets: 1) the dataset of students enrolled from 2001 to 2003 years and choosing the *Database and Information Systems* curriculum, 2) the dataset of students enrolled in the same period and choosing the *Distribuited Systems* curiculum, finally, 3) the dataset of all students enrolled from 2001 to 2003 years.

We apply the frequent patterns model first to the dataset 1); the careers of these students have 25 common exams. Then we apply the frequent patterns model to the dataset 3); the careers of these students have 16 common exams.

Finally we perform the statistical analysis of the delay distributions on the dataset 3) of all student.

In Chapter 6, we conclude with an overview of the analysis performed on the degree program introduced in Chapter 5 and give some ideas for further and future analysis.

In the Appendixes A and B we list all figures and tables that are not presented along the chapters.

2

DATA ORGANIZATION

Information assets are immensely valuable to any enterprise, and consequently, these assets must be properly stored and readily accessible when they are needed. However, the availability of so many data makes the extraction of the most important information difficult, if not impossible. We can view results from any web search, and we will see that the *data = information* equation is not always correct, that is, too much data is simply too much.

Data warehousing is a phenomenon grown from the huge amount of electronic data stored in recent years and from the urgent need to use data to accomplish goals that go beyond the routine tasks linked to daily processing. In a typical scenario, a large corporation (but this is valid also for universities) has many branches, and senior managers need to quantify and evaluate how each branch contributes to the global business performance. The corporate database stores detailed data on the tasks performed by branches. To meet the needs of managers, tailor made queries can be issued to retrieve the required data. In order for this process to work, database administrators must first formulate the desired query (typically an aggregate SQL query) after closely studying database catalogs. Then the query is processed. This can take a few hours because of the huge amount of data, the query complexity, and the concurrent effects of other regular workload queries on the same data. Finally, a report is generated and passed to senior managers in the form of a spreadsheet.

Many years ago, database designers realized that such an approach is hardly feasible, because it is very demanding in terms of time and resources, and does not always achieve the desired results. Moreover, a mix of analytical queries with transactional routine queries inevitably slows down the system, and this does not meet the needs of users of either type of queries. Recent advanced data warehousing processes separate *online analytical processing* (OLAP) from *online transactional processing* (OLTP) by creating a new information repository that integrates basic data from various sources, properly arranges data formats, and then makes data available for analysis and evaluation aimed at planning and decision-making processes.

The field of application of data warehouse systems is not only restricted to enterprises, but interests almost every field, from epidemiology to demography, from natural science to education. A property that is common to all fields is the need for storage and query tools to retrieve information summaries easily and quickly from the huge amount of data stored in databases or made available by the Internet. This kind of information allows us to study business phenomena,

learn about meaningful correlations, and gain useful knowledge to support decision making processes.

Until the mid-1980s, enterprise databases stored only *operational data*, such as data created by transactional operations involved in daily management processes; however, every enterprise must have quick, comprehensive access to the information required by decision-making processes. This strategic information is extracted mainly from the huge amount of operational data stored in enterprise databases by means of a progressive selection and aggregation process. An exponential increase in operational data has made computers the only tools suitable for providing data for decision-making performed by business managers. This fact has dramatically affected the role of enterprise databases and fostered the introduction of *decision support systems* (DSS).

The concept of decision support systems mainly evolved from two research fields: theoretical studies on decision-making processes for organizations and technical research on interactive Information Technology (*IT*) systems. However, the decision support system concept is based on several disciplines, such as databases, artificial intelligence, man-machine interaction, and simulation, as described in the overview on Educational Data Mining of C. Romero and S. Ventura [78]. Decision support systems became a research field in the mid-'70s and grew more popular in the '80s.

In practice, a *DSS* is an *IT* system that helps managers to make decisions or choose among different alternatives. The system provides value estimates for each alternative, allowing the manager to critically review the results. From the architectural viewpoint, a *DSS* typically includes a model management system connected to a knowledge engine and, of course, an interactive graphical user interface. Data warehouse systems have been managing the data back ends of *DSS*s since the 1990s. They must retrieve useful information from a huge amount of data stored on heterogeneous platforms. In this way, decision-makers can formulate their queries and conduct complex analyses on relevant information without slowing down operational systems.

2.1 DATA WAREHOUSE

By referring to the complete discussion presented in [39], we define *data warehousing* as a collection of methods, techniques and tools used to support *knowledge workers* (senior managers, directors, managers, and analysts) to conduct data analyses that help performing decision making processes and improving information resources. We precise that data warehousing is only one of the different data organizations on which data mining techniques can be applied, but in this thesis we refer to many issues that are typical of this technical context. A data warehouse process is a set of tasks that allow us to turn operational data into decision making support information:

- accessibility to users not very familiar with IT and data structures;
- integration of data on the basis of a standard enterprise model;
- query flexibility to maximize the advantages obtained from the existing information;
- information conciseness allowing for target-oriented and effective analyses;
- multidimensional representation giving users an intuitive and manageable view of information;
- correctness and completeness of integrated data.

Data warehouses (also indicated in this thesis by DW) are placed right in the middle of this process and act as repositories for data. They make sure that the requirements set can be fulfilled. Referring to [49] we can give the following definition:

> A **data warehouse** is a collection of data that supports decision-making processes. It provides the following features:
>
> - It is *subject-oriented.*
> - It is *integrated* and *consistent*.
> - It shows its evolution over time and is *not volatile*.

Data warehouses are subject-oriented because they depend on enterprise-specific concepts, such as customers, products, sales, and orders, rather than students, exams and grades. On the contrary, operational databases depend on many different enterprise-specific applications.

We emphasize on integration and consistency because data warehouses take advantage of multiple data sources, such as data extracted from production and then stored to enterprise databases, or even data from a third party's information systems; we will see a sample of data integration in the case study presented in Chapter 5. A data warehouse should provide a unified view of all the data. Generally speaking, we can state that creating a data warehouse system only requires that existing information is rearranging. This implicitly means that an information system should be previously available.

Operational data usually cover a short period of time, because most transactions involve the latest data. A data warehouse should enable analyses that instead cover a few years. For this reason, data warehouses are regularly updated with operational data and keep on growing, that is, periodically its tables are refreshed by adding new rows into them. If data were visually represented, it might proceed like so: a photograph of operational data would be made at

regular intervals. The sequence of photographs would be stored to a data warehouse and results would be shown in a movie that reveals the status of an enterprise from its foundation until present.

Several differences exist between operational databases and data warehouses, some connected with query types, others depending on the users. Table 2.1, presented also in [39], is a good schema to understand the most important differences.

Feature	**Operational databases**	**Data Warehouses**
Users	Thousands	Hundreds
Workload	Preset transactions	Specific analysis queries
Access	To hundreds/thousands of records, write and read mode	To millions of records, mainly read-only mode
Goal	Depends on applications	Decision-making support
Data	Detailed, both numeric and alphanumeric	Summed up, mainly numeric
Data integration	Application-based	Subject-based
Quality	In terms of integrity	In terms of consistency
Time coverage	Current data only	Current and historical data
Updates	Continuous	Periodical
Model	Normalized	Denormalized, multidimensional
Optimization	For OLTP access to a database part	For OLAP access to most of the database

Table 2.1: Differences between Operational Databases and Data Warehouses.

For what concerns the *data warehouse architecture*, we can have different models, depending on how many physical data levels we have. The *single-layer* architecture, which has the goal to minimize the amount of data stored, is not frequently used and provides a virtual warehouse level; in this architecture, OLAP accesses are performed on the data transactional layer. The requirement for separation plays a fundamental role in defining the typical architecture for a data warehouse system; it is typically called a *two-layer* architecture to highlight a separation between physically available sources and data warehouses; it consists in four subsequent data flow stages: source layer, data staging, data warehouse layer and analysis. With *data staging* layer we mean the whole set of phases to extracting, transforming and loading. These phases need that heterogeneous sources of data are merged into one common schema.

The third layer in the *three-layer* architectures is the *reconciled data* layer or operational data store. This layer materializes operational data obtained after integrating and cleaning source data. As a result, those data are integrated, consistent, correct, current, and detailed; they are then ready for any type of analysis, including the data mining, as we will present in the following chapters. The data warehouse is not directly populated from its sources, but from reconciled data. The main advantage of the reconciled data layer is that it creates a common reference data model for the whole enterprise. At the same time, it sharply separates the problems of source data extraction and integration from those of data warehouse population. We may assume that even two-layer architectures can have a reconciled layer that is not specifically materialized, but only virtual, because it is defined as a consistent integrated view of operational source data. Figure 2.1 shows the most complete architecture of a datawarehouse, with the three layers of data.

As we already observed, the datawarehouse systems are the most popular among the tools to support decision making activities. In addition to these, however, there are wider and more etergeneo solutions, which together are known as *business intelligence* (BI) systems. Techniques on which the BI systems are based are already known for some time but have only recently begun to attract the attention of users to whom the experience with the data warehouse systems has developed the tendency towards the use of sophisticated approaches, such as *data mining*, that we will see in the next chapter.

2.2 TYPES OF DATA

In the data mining research area, besides the problem of the huge amount of data, we have to deal with several types of data and with data quality ([75, 93]); in this sense we think it is important to present a brief overview about the types of data. In general, a *data set* can be viewed as a collection of *data object*, called also *record*, *point*, *vector*, *event*, *observation*, or *entity*, but we know that there are many types of data sets and, as the field of data mining develops and matures, a greater variety of data sets becomes available for analysis. We group the types of data sets into three principal groups : *record data*, *graph-based data* and *ordered data* [90].

In particular, there are different variations of *record data*. Much data mining work assumes that the data set is a collection of *records*, each of which consists of a fixed set of data fields, or attributes, as shown in Figure 2.2. There is no explicit relationship among records or data fields, and every record has the same set of attributes. Record data is usually stored either in *flat* files or in relational databases. Relational databases are certainly more than a collection of records, but data mining often does not use any additional information available in a

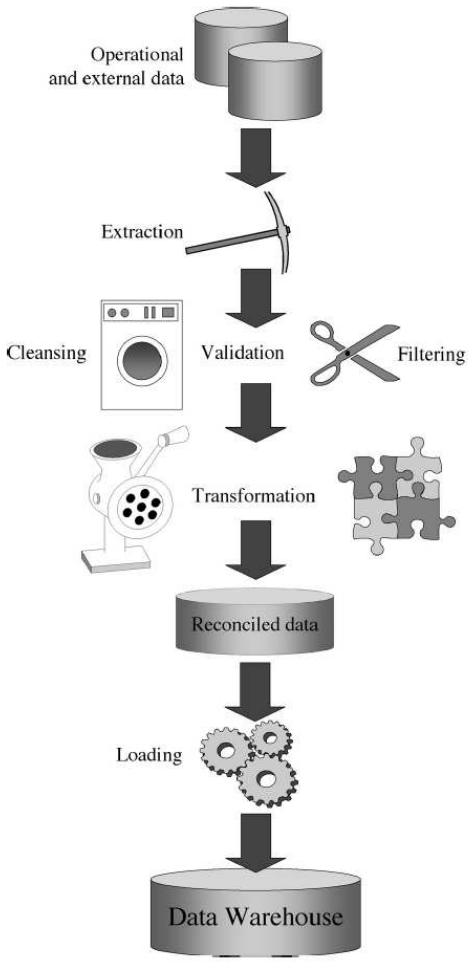


Figure 2.1: The *three-layer* architecture of a DW (source [39])

.

relational database. Rather, the database serves as a convenient place to find records.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Table 2.2: A sample of record data.

Transaction data is a special type of record data, where each record, or transaction, involves a set of items. For example, in a grocery store, the set of products purchased by a customer during one shopping trip constitutes a transaction (Figure 2.3), while the individual products that were purchased are the items. This type of data is called *market basket data* because the items in each record are the products in a person's *market basket*. As we will discuss in Chapter 4, for the analysis of students based on the sequential pattern technique we refer to this representation of data, while for the traditional clustering analysis we refer to record data.

Tid	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Bread, Coke, Diaper, Milk
4	Bread, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Table 2.3: A sample of transaction data.

Another type of record data is the *data matrix*. If the objects in a collection of data all have the same fixed set of numeric attributes, the data objects can be thought of as points, or vectors, in a multidimensional space, where each

dimension represents a distinct attribute describing the object. A set of such data objects can be interpreted as an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute. This matrix is called a data matrix or a pattern matrix; it is the standard data format for most statistical data. Samples of this data can be seen in Table 2.4.

Projection of x Load	**Projection of y Load**	**Distance**	**Load**	**Thickness**
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Table 2.4: A sample of data matrix.

In this chapter we have addressed several issues that allow us to obtain a useful foundation for the data analysis of an information system. In fact, if our goal is to discover any hidden information in the data, we must make sure we will set our analysis on corrected and well organized data; this represents an essential element in order to perform a good analysis with data mining techniques.

3

DATA MINING TECHNIQUES

Data mining is an integral part of the knowledge discovery in databases process (KDD), which is the overall process of converting raw data into useful information.

This process, showed in Figure 3.1, consists in a series of transformation steps, from data preprocessing to postprocessing of data mining results. The input data can be stored in several formats and may reside in a centralized data repository or be distributed across multiple sites, as we already observed in Chapter 2. The purpose of **preprocessing** is to transform the raw input data into an appropriate format for subsequent analysis. The step involved in data preprocessing includes merging data from multiple sources, cleaning data to remove noise and duplicate observations and selecting records and features that are relevant to the data mining task at hand. Because of the many ways data can be collected and stored, data preprocessing is the most laborious and time-consuming step in the overall knowledge discovery process.

Data mining results have to be integrated into a decision support system; for example, in business or in more particular educational applications, the insight offered by data mining results can be integrated with management tools of promotional campaigns, so that effective marketing promotions can be conducted and tested. This integration requires a **postprocessing** step, as we will see in the case study presented in Chapter 5, ensuring that only valid and useful results are embedded into the decision support system. Traditional analysis tools such as OLAP, introduced in the previous chapter, represent a base for a decision support system, but the data mining allows to make a big leap in the direction of the extrapolation of the knowledge of hidden relationships among data.

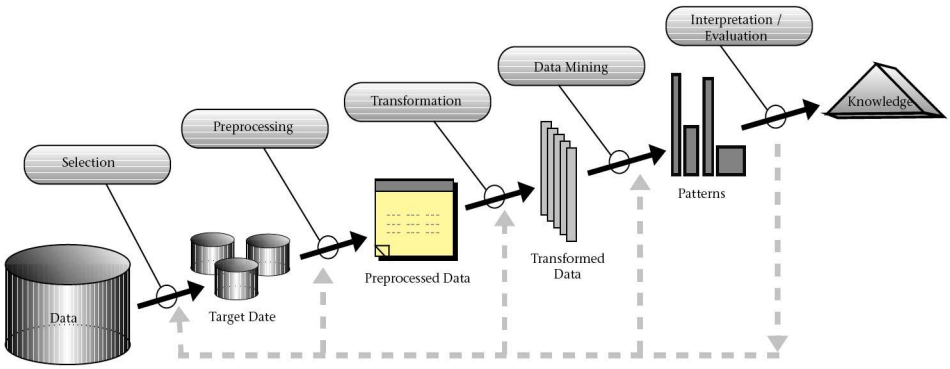


Figure 3.1: An overview of the steps that compose the KDD process (source [35]).

The Origins of Data Mining

Brought together by the goal of meeting the challenge of the difficulty of integrating different and new types of data sets, researchers from different disciplines began to focus on developing more efficient and scalable tools that could handle diverse types of data. This work, which culminated in the field of data mining, built upon the methodologies and algorithms that researchers had previously used. In particular, data mining draws upon ideas, such as sampling, estimation, and hypothesis testing from statistics and search algorithm, modeling techniques, and learning theories from artificial intelligence, pattern recognition, and machine learning. Data mining has also adopted ideas from other areas, such as optimization, information theory and information retrieval. The topic of data mining has inspired many textbooks. Some of these, such as [11], have a strong emphasis on business application, others [48] have an emphasis on statistical learning or [32, 55] on machine learning. There are also some more specialized books, for example on web mining [26], while in [46] the author illustrates how data mining differs from statistics.

In recent years data mining has been applied also to particular areas, such as social network and sentimental analysis. For what concerns *network analysis*, for which we can refer to [68] for a useful overview, we observe that the starting point is *The Small World Problem*, an experiment by which S. Milgram, in 1967, considered the graph of acquaintance chains and then (1969), with J. Travers, tried experimentally that there were same regularities that could variate if we changed starting populations [63]. An interesting basic paper that deepened this area is [40]. *Sentimental analysis*, also called *opinion mining*, is a recent discipline at the crossroads of the following different areas:

- information retrieval, that is, the activity of obtaining information relevant from a collection of information resources. Searches can be based on metadata or on full-text indexing;
- text mining, that refers to the process of deriving high-quality information from text, deriving through the devising of patterns and trends;
- computational linguistics, that is, an interdisciplinary field dealing with the statistical or rule-based modeling of natural language from a computational perspective.

Opinion mining tries to detect the opinions expressed in the natural language texts. It focuses not on the subject mentioned by a document, but on the opinion that the documentexpresses [58]. A particular problem in the area of opinion mining is the *comparative sentences mining*, that is related to a context quite different from opinion sentence identification or classification. Sentiment classification studies the problem of classifying a document or a sentence based on the subjective opinion of the author, instead comparisons can be subjective or

objective. A typical example of opinion is *The picture quality of camera X is great*, besides, an example of comparative sentences can be *Camera X is better than Y*. This problem [53, 54] has many applications, from industry, to know customer opinions about some products in comparison with other products, to politics, to know if people like more a candidate in comparison with another.

Data Mining and Ethics

The use of data for data mining has serious ethical implications, and practitioners of data mining techniques must act responsibly by making themselves aware of the ethical issues that surround their particular applications. When applied to people, data mining is frequently used to discriminate, who gets the loan, who gets the special offer, and so on. Certain kinds of discrimination, racial, sexual, religious, and so on, are not only unethical but also illegal. However, the situation is complex: everything depends on the application.

Recent work in what are being called *reidentification* techniques has provided sobering insight into the difficulty of anonymizing data. The aim is to anonymize the data; by removing all personal information we cannot be sure that data are really anonymous, because we can leave in the information something useful. There is growing interest in developing data mining algorithms that maintain user privacy; as a result, the preserving privacy is an important and recent area of research that accompanies the data mining. Among some general reference in this area we cite [4]; we can also refer to [38, 64] to pursue this issues.

3.1 SUPERVISED AND UNSUPERVISED LEARNING

Data and Knowledge Mining consists in learning from data. In this context, data are allowed to speak for themselves and no prior assumptions are made. This learning from data comes in two flavors: *supervised learning* and *unsupervised learning*.

In supervised learning (often also called *directed data mining*) the variables under investigation can be split into two groups: explanatory variables and dependent variables. The target of the analysis is to specify a relationship between the explanatory variables and the dependent variable. To apply directed data mining techniques the values of the dependent variable must be known for a sufficiently large part of the data set.

Unsupervised learning is closer to the exploratory spirit of data mining as stressed in the definitions given above. In unsupervised learning situations all variables are treated in the same way and there is no distinction between explanatory and dependent variables. However there are still some targets to achieve, such as general data reduction or more specific clustering.

The separating line between supervised and unsupervised learning is the same that distinguishes discriminant analysis from cluster analysis. Supervised

learning requires that the target variable is well defined and that a sufficient number of its values is given. Typically, for unsupervised learning either the target variable is unknown or has only been recorded for a too small number of cases.

The large amount of data usually present in data mining tasks allows to split the data file in three groups: *training* cases, *validation* cases and *test* cases. Training cases are used to build a model and estimate the involved parameters. The validation data helps to see whether the model obtained with the chosen sample may be generalized to other data. In particular, it helps avoiding the phenomenon of overfitting, that is, the situation in which the model is too near to the considered data. Test data can be used to assess the various methods and to choose the one performing the best job on the long run.

3.2 DATA MINING TASKS

Today there are several data mining techniques (corresponding to different algorithms) and various definitions about data mining categories can be given. Data mining tasks can be divided into two major categories:

- **descriptive tasks**, where the objective is to derive patterns (correlations, trends, clusters, ...) that summarize the underlying relationship between data;
- **predictive tasks**, by which the value of a particular attribute *target*, or *dependent variable*, can be predicted by using the values of other attributes.

Typical methods of descriptive modeling are density estimation, smoothing, data segmentation and clustering. There are by now some classic papers and books in the literature on density estimation [85] and smoothing [47]. Clustering is a well studied and well known technique in statistics. Many different approaches and algorithms, distance measures and clustering schemata have been proposed. The most widely used method of choice is `K-means` clustering. Although `K-means` is not particularly tailored for a large number of observations, it is currently the only clustering scheme that has gained positive reputation in both the computer science and the statistics community. The reasoning behind cluster analysis is the assumption that the data set contains natural clusters which, when discovered, can be characterized and labeled. While for some cases it can be difficult to decide to which group the data belong, we assume that the resulting groups are well separated and carry an intrinsic meaning. In segmentation analysis, in contrast, the user typically sets the number of groups in advance and tries to partition all cases in homogeneous subgroups.

Predictive modeling belongs to the category of supervised learning, hence one variable is clearly labeled as target variable X and it will be explained as a function of the other variables. The nature of the target variable determines

the type of model: classification model, if X is a discrete variable, or regression model, if it is continuous. Many models are typically built to predict the behavior of new cases and to extend the knowledge to objects that are new or not yet as widely understood, such as predicting the value of the stock market, the outcome of the next governmental election, or the health status of a person. For example banks use classification schemes to group their costumers into different categories of risk.

3.3 CLUSTERING ANALYSIS

Cluster analysis divides data into groups (*clusters*) that are meaningful, useful, or both. If meaningful groups are the goal, then the cluster should capture the natural structure of the data. In some cases, however, cluster analysis is only a useful starting point for other purposes, such as data summarization. Cluster analysis has long played an important role in a wide variety of fields: psychology and other social sciences, biology, statistics, pattern recognition, information retrieval, machine learning and data mining.

Cluster analysis groups objects only on the basis of information found in the data describing the objects and their relationships. The goal is to group together objects that are similar or related and, at the same time, are different or unrelated to the objects in other clusters. The greater is the similarity (or homogeneity) within a group and the greater are the differences between groups, the more distinct are the clusters.

In many applications the notion of a cluster is not well defined depending on the nature of data and on the desired results. Cluster analysis is related to other techniques that are used to divide data objects into groups. For instance, clustering can be regarded as a form of classification since it creates a labeling of objects, by deriving these labels only from the data. For this reason cluster analysis is referred as unsupervised classification, while, for example, classification based on decision trees is supervised classification.

Various types of clustering can be distinguished: hierarchical versus partitional, exclusive versus overlapping, complete versus partial. The most commonly discussed distinction among different types of clusterings is whether the set of clusters is nested or unnested, or in a more traditional terminology, hierarchical or partitional (see, e.g.,[36, 52, 56, 96]). A partitional clustering is simply a division of the set of data objects into non-overlapping subsets such that each data object is in exactly one subset. If we allow clusters to have subclusters, then we obtain a hierarchical clustering, which is a set of nested clusters organized as a tree. Each node in the tree is the union of its children; in particular the root of the tree is the cluster containing all the objects.

We can also distinguish different types of clusters. For example, we have well-separated clusters and density-based clusters. In the first case a cluster is

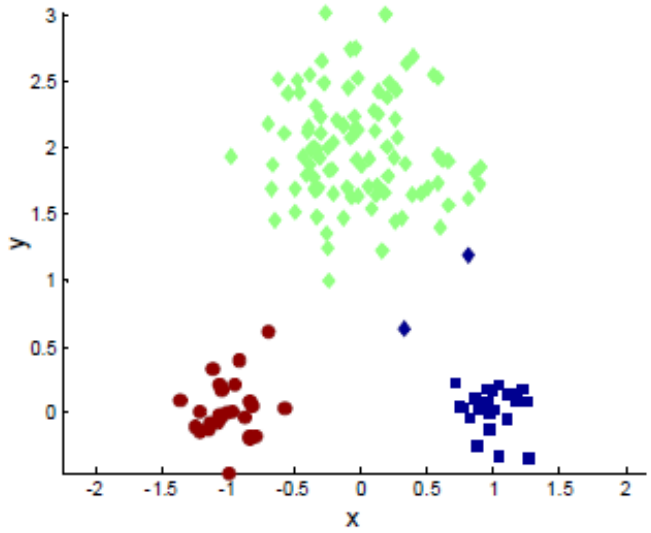


Figure 3.2: A sample of well separated clusters (source [92]).

a set of objects in which each object is closer, or more similar, to every other object in the cluster than to any object not in the cluster. Sometimes a threshold is used to specify that all the objects in a cluster must be sufficiently close to one another. The distance between any two points in different clusters is larger than the distance between any two points within a group. A sample of optimal well separated clusters is illustrated in Figure 3.2. With density-based clusters a cluster is a region dense of objects that is surrounded by a region of low density. This type of cluster is often employed where the clusters are irregular or intertwined, and when noise and outliers are present [1]. In density-based clustering clusters are defined as areas of higher density than the remainder of the data set. Objects in these sparse areas, that are required to separate clusters, are usually considered to be noise or border points; we can see a sample of optimal density based clusters in Figure 3.3. A general survey of clustering is given in [51], while in [10] we find a survey of clustering techniques for data mining.

3.3.1 *K-means clustering*

Input data characteristics are important to decide what type of clustering is the best for the data mining analysis. K-means is a very simple algorithm based on partitional approach; it is the most famous among the clustering algorithms and in many cases it gives good results. Each cluster is associated with a *centroid* (center point) and each point is assigned to the cluster with the closest centroid.

1 Outliers are different from the noise data: an outlier is a data object that deviates significantly from the normal objects as if it were generated by a different mechanism; noise is a random error or variance in a measured variable and should be removed before outlier detection.

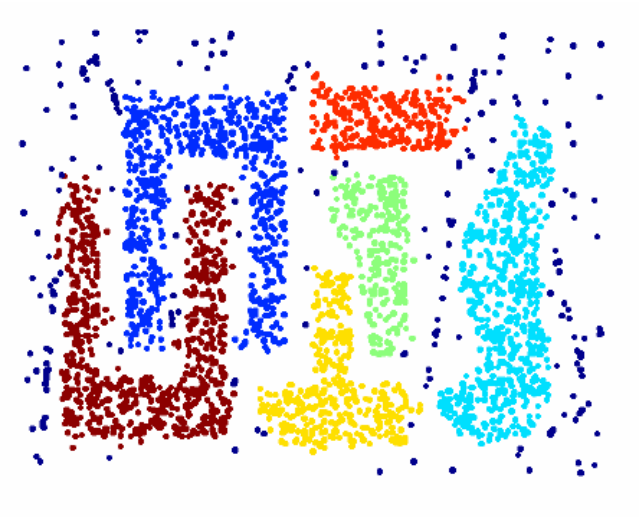


Figure 3.3: A sample of density based clusters (source [92]).

The number of clusters, K, must be specified. The original `K-means` algorithm was proposed by MacQueen [60] and, with many of its variations, is described in detail in the books by Anderberg [5] and by Jain and Dubes [50]. Figure 3.4 shows the steps of the algorithm:

K-means algorithm
1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

Figure 3.4: The `K-means` algorithm.

By applying the implementation of `K-means` algorithm in `WEKA` [94] to data in Table 3.1, by considering the two dimensions (attributes) **FinalGrade** and **Time**, respectively the degree grade and the time, in years, to graduate for a university student, we obtain the clusters showed in the Figure 3.5.

We can see two clusters; the red one corresponds to good students, which have a *high* **FinalGrade** and a *small* **Time**; the students not so good, which correspond to the blue stars, have a *low* **FinalGrade** and a *large* **Time**.

Recomputing the centroid of each clusters in the `K-means` algorithm is fundamental to ensure that every point is assigned to the cluster containing the points at which really it is nearest. For this, the distance of each point to its closest centroid has to be minimized. By considering data whose proximity measure is the Euclidean distance, we can use the *sum of the squared error* (**SSE**)

Id-Student	Sex	FinalGrade	Time
10	F	110	1400
20	M	95	1380
30	M	100	2000
40	F	103	1180
50	M	98	2030
60	F	106	1220
70	F	110	1003

Table 3.1: A table of a student database.

to measure the quality of a clustering. By using the SSE measure, the error of each data point, that is its Euclidean distance to the closest centroid, is calculated, then the total sum of the squared errors is computed. If we have to choose between two different sets of clusters (as we illustrate later), produced by two different executions of `K-means`, we prefer the one with the smallest squared error. It is important also to observe that the obtained clusters depend on the initial choice of centroids.

These and other considerations are the basic concepts of clustering analysis by `K-means`; the goodness of this technique depends for example on the choice of the initial centroids. In the following paragraphs we give some hints, but for an exhaustive discussion we refer to [92].

Choosing initial centroids

When random initialization of centroids is used, different runs of `K-means` produce different total SSEs. Choosing the good initial centroids is the key of the basic `K-means` procedure; a common approach is to choose the initial centroids randomly, but the resulting clusters are often poor. We explicitly observe that, since the `K-means` algorithm finds local optimum for SSE, different initial centroids may lead to a different local optimum. A good technique to address the problem of choosing initial centroids is to perform multiple runs, each with a different set of randomly chosen initial centroids, and then select the set of clusters with the minimum SSE. We observe that the efficacy of this strategy depends on the data set and the number of clusters required.

Derivation of `K-means` to minimize the SSE

The centroids for the `K-means` algorithm can be mathematically derived when the proximity function is the Euclidean distance and the objective is to minimize

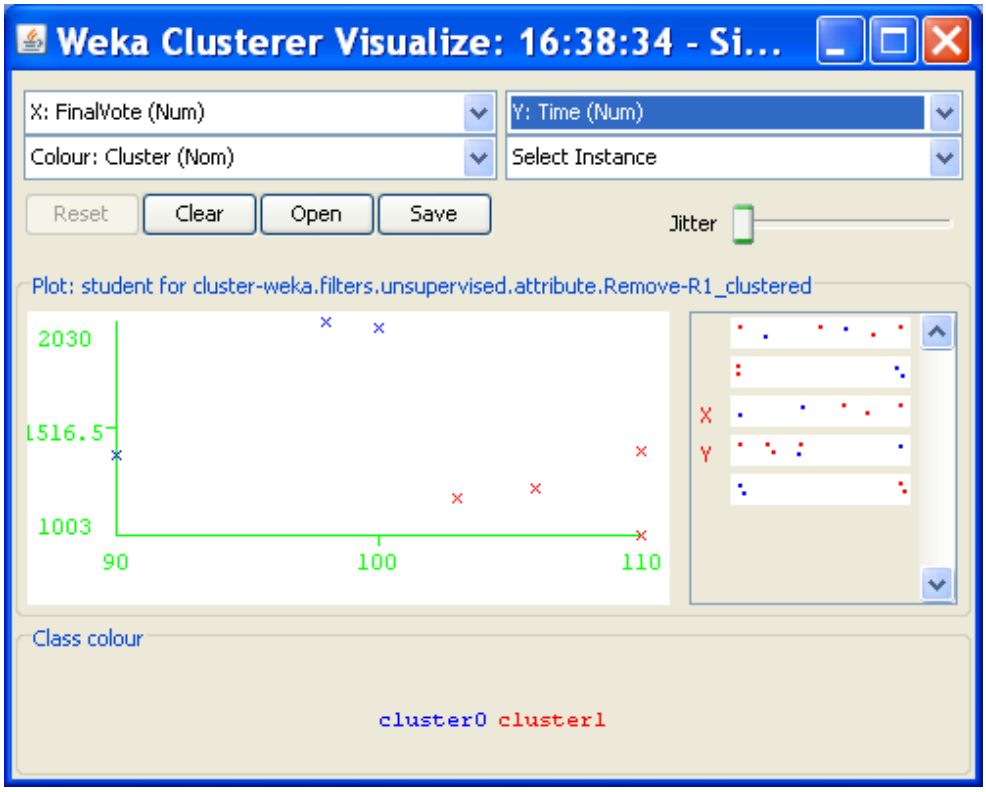


Figure 3.5: Clusters corresponding to the students data of Table 3.1.

the SSE. This is done by minimizing the following equation, which refers to one dimensional data,

$$\text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2,$$

where C_i is the i^{th} cluster, x is a point in C_i and c_i is the mean of the i^{th} cluster. We can solve for the k^{th} centroid c_k, by differentiating the SSE , setting it equal to 0 and solving; we obtain that the best centroid for minimizing the SSE of a cluster is the mean of the points in the cluster. The K-means algorithm can be applied to a variety of different objective functions; we can use the Manhattan distances of points from the center of their clusters and choose the data partition which minimizes the *sum of this distance absolute error* (**SAE**). By considering one dimensional data, that is, $\text{dist} = |c_i - x|$, we have

$$\text{SAE} = \sum_{i=1}^{K} \sum_{x \in C_i} \text{dist}(c_i, x).$$

Analogously to what was done for SSE, by solving for c_k, we find that $c_k = \textit{median}\{x \in C_k\}$, the median of the points in the cluster. The median of a group of points is linear to compute and less susceptible to distortion by outliers.

In the open source software WEKA we can run the *K*-means algorithm with both measures, SSE and SAE.

3.3.2 *DBSCAN clustering*

Density-based clustering locates regions of high density that are separated from one another by regions of low density. DBSCAN is a simple and effective density-based clustering algorithm that illustrates a number of concepts important for any density based clustering approach. There are several distinct methods for defining density; DBSCAN is based on the center-based approach, in which density is estimated for a particular point in the data set by counting the number of points within a specified radius, *Eps*, of that point. This includes the point itself. This method is simple to implement, but the density of any point will depend on the specified radius. For example, if the radius is large enough, then all points will have a density of m, the number of points in the data set. Likewise, if the radius is too small, then all points will have a density of 1. There are several approaches for deciding on the appropriate radius; we refer to [34, 82] to explore this topic and to [6] for OPTICS, an interesting outgrowth of DBSCAN.

The center-based approach to density allow us to classify a point as being in the interior region of a dense region (a *core* point), on the edge of a dense region (a *border* point), or in a sparsely occupied region (a *noise* point). One of the most important differences between *K*-means and DBSCAN is about noise; both DBSCAN and *K*-means are partitional clustering algorithms that assign each object to a single cluster, but *K*-means typically clusters all the objects, while DBSCAN discards objects that it classifies as a noise. In particular, the main difference is in the forms of the clusters that two methods can obtain; *K*-means always obtains convex clusters whereas DBSCAN allows for non-spherical clusters. In other words, DBSCAN clusters according to the axiom *if two points are close, and in a dense region, they should be in the same cluster*, while *K*-means is rather based upon the axiom *every point should be closer to the center of its own cluster than to the center of another cluster*. Figure 3.6 shows the steps of the algorithm DBSCAN.

DBSCAN algorithm
1: Label all points as core, border, or noise points.
2: Eliminate noise points.
3: Put an edge between all core points that are within *Eps* of each other.
4: Make each group of connected core points into a separate cluster.
5: Assign each border point to one of the clusters of its associated core points.

Figure 3.6: The DBSCAN algorithm.

3.4 ANALYSIS BASED ON CLASSIFICATION

Classification, the task of which is to assign objects to one among several predefined categories, is a pervasive problem encompassing many different applications. The input data for a classification task is a collection of records. Each record, also known as an *instance* or *example*, is characterized by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the attribute set and y is a special attribute, designed as the *class label* (also known as *category* or *target* attribute). The class label must be a discrete attribute and this is the distinguishing feature of classification from regression, a predictive model task in which y is a continuous attribute. Classification is the task of learning a target function f that maps each attribute set $\mathbf{x}$ to one of the predefined class labels y. The target function is also known as a classification model, which can serve as an explanatory tool to distinguish between objects of different classes (descriptive modeling), but it can also serve to predict the class label of unknown records. Classification techniques are most suited for predicting or describing data sets with binary or nominal categories. They are less effective for ordinal categories (for example to classify a person as a member of high, medium or low income group), because they do not consider the implicit order among the categories. However there are some approaches using cost-sensitive classification; classifying a high instance as medium receives a lower penalty than classifying it as low. The goal is to minimize the expected cost and in this way classifiers that predict an ordinal class attribute can be learned.

A classification technique is a systematic approach for building classification models from an input data set. There are many classifiers, such as rule based, neural networks and decision tree classifiers; each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data. The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of records it has never seen before.

A general approach for solving classification problems starts by identifying a *training set* which consists of records whose class labels are known; the training set is used to build a classification model, which is subsequently applied to the *test set*, which consists of records with unknown class labels. Evaluation of the performance of a classification model is based on the count of test records correctly and incorrectly predicted by the model. The tool and the parameters by which the evaluation can be given are the confusion matrix, the accuracy, the precision and the recall (see, e.g., [92]).

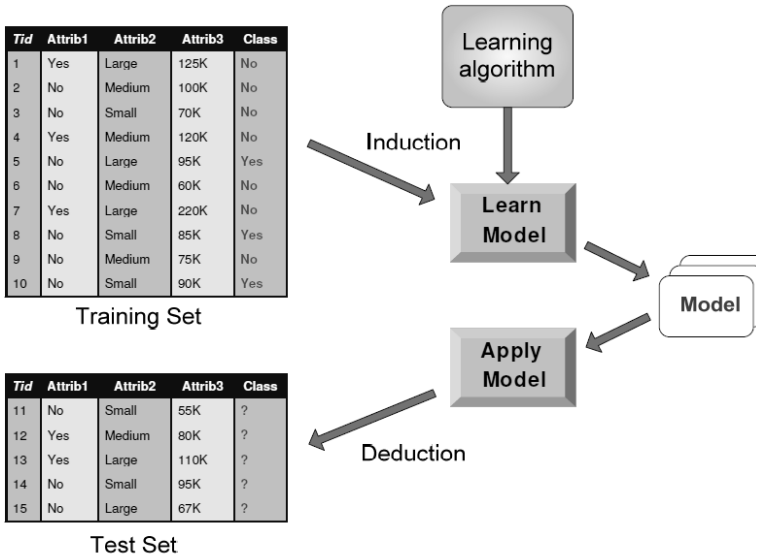


Figure 3.7: General approach for building a classification model (source [92]).

3.4.1 *Decision trees*

Building an optimal decision tree is the key problem in decision tree classifiers. In general, many decision trees can be constructed from a given set of attributes. While some of the trees are more accurate than others, finding the optimal tree is computationally infeasible because of the exponential growth of the search space. However, various efficient algorithms have been developed to construct a reasonably accurate, although suboptimal, decision tree in a reasonable amount of time. These algorithms usually employ a greedy strategy that builds a decision tree by making a series of locally optimal decisions according to the attribute to be used for partitioning the data. For example, Hunt's algorithm, which is the basis of many decision tree induction algorithms, including `ID3`, `C4.5` and `CART`, is of this kind. An overview of decision tree induction algorithms can be found in the survey articles [17, 65, 67, 81].

3.4.2 *Hunt's Algorithm*

Hunt's algorithm builds a decision tree in a recursive fashion by partitioning the training records into successively purer [2] subsets. The general recursive procedure is defined in Figure 3.8.

The procedure is recursively applied to each subset until all the records in the subset belong to the same class. The Hunt's algorithm assumes that each combination of attribute sets has a unique class label during the procedure. If

2 A subset A is purer than a subset B if A contains more records belonging to a particular class than B contains.

***Hunt's* algorithm**
1: Let D_t be the set of training records that reach a node t.
2: If D_t contains records that belong to the same class y_t, then t is a leaf node labeled as y_t.
3: If D_t is an empty set, then t is a leaf node labeled by the default class, y_d.
4: If D_t contains records that belong to more than one class, use an attribute test (that is an attribute of the whole data sets) to split the data into smaller subsets.

Figure 3.8: The *Hunt's* algorithm.

all the records associated with D_t have identical attribute values except for the class label, then it is not possible to split these records any further. In this case, the node is declared a leaf node with the same class label as the majority class of training records associated with this node.

Various considerations can be made about this algorithm, both for what concerns the methods to express test conditions of different types of attributes, and for what concerns the measures for the selection of the best split.

There are many measures to determine the best way to split the records. These measures are defined in terms of the class distribution of the records before and after splitting and they are often based on the *degree of impurity* of the child nodes. The smaller the degree of impurity, the more skewed is the class distribution. For example, by considering a two class problem, the class distribution at any node can be written as (p_0, p_1), where $p_1 = 1 - p_0$. A node with class distribution $(0, 1)$ has zero impurity, whereas a node with uniform class distribution $(0.5, 0.5)$ has the highest impurity. Some of the most used measures of node impurity are:

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t),$$

$$Gini(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2,$$

$$Classificationerror(t) = 1 - max_i[p(i|t)],$$

where $p(i|t)$ denotes the fraction of records belonging to class i at a given node t, c is the number of classes and $0 log_2 0 = 0$ in entropy calculations.

In the case study illustrated in Chapter 5 we will present some results obtained with the implementation of `C4.5` algorithm in the open source software `WEKA`. This algorithm employs the entropy measure as its splitting function. An

in-dept discussion of the C4.5 decision tree algorithm is given in [76], where the author, besides explaining the methodology for decision tree growing and tree pruning, also describes how the algorithm can be modified to handle data sets with missing values. Another well known algorithm is CART, developed by Breiman et al. [76]; it uses the Gini index as its splitting function. These decision tree algorithms assume that the splitting condition is specified one attribute at a time. An option for using linear combinations of attributes is proposed in a CART implementation [14].

We refer to [92] for details and further information of this topic and on classification techniques in general.

3.5 ASSOCIATION ANALYSIS

Association analysis is the discovery of interesting relationships (rules) hidden in large data sets. It studies the frequency of items occurring together in transactional databases and is based on a threshold, called *support* (s), that identifies the fraction of transactions that contain an itemset. Another threshold, *confidence*, which is the conditional probability that an item appears in a transaction when another item appears, is used to find association rules.

The *support count* (σ) of an itemsets is defined as the frequency of its occurrences.

For example, in the transactional database of Table 3.2 the list of exams for each student is showed; by considering the itemsets $i_1 = \{7\}$ and $i_2 = \{3, 7\}$, we have $\sigma(i_1) = 3$, $\sigma(i_2) = 2$, $s(i_1) = 0.75$ and $s(i_2) = 0.5$. In fact, by considering the itemset i_1, the exam 7 is taken by 3 students, and this represents the 75% of students. By considering the itemset i_2, the pair of exams $3, 7$ is taken by 2 students, and this represents the 50% of students.

Id-Student	**Items (exams)**
10	$1, 2, 3, 4, 5, 7$
20	$1, 2, 3, 4, 7$
30	$1, 4, 7$
40	$1, 2, 3, 5$

Table 3.2: A transactional student database.

3.5.1 *Association rules*

Rule induction methods are widely applied tools for mining large data bases. They are often used as a starting point in undirected data mining, that is, when we do not know what specific patterns to look for. One form of rule induc-

tion methods are *association rules*, well known in market basket analysis. They were proposed in [1] with the aim to provide an automated process, which could find new connections among items, and especially to answer questions like: *which products (items) are likely to be bought together?* Association rules have been involved in many areas and applications, such as in telecommunication or insurance business.

In general, an association rule is an implication of the form $X \to Y$, where X and Y are mutually exclusive itemsets. The quality of an association rule is measured by *support* and *confidence* (see [15] to investigate the use of objective measures to filter patterns).

Let I be the set of items and D be a transaction database. A transaction t contains a subset of items chosen from I. The *transaction width* is defined as the number of items present in a transaction. A transaction t_j is said to contain an itemset X if X is a subset of t_j. The support count of an itemset X, that is the number of transactions that contain it, can be stated as follows:

$$\sigma(X) = |t_i | X \subseteq t_i, t_i \in D|,$$

where the symbols | | denote the number of elements in a set.

If X e Y are two disjoint itemsets of elements in D, the rule $X \to Y$ has *support* s in the database D, if the fraction s of transactions in D contains $X \cup Y$. A rule holds with *confidence* $c = c(X \to Y)$, if the fraction c of transactions in D that contain X also contain Y. The formal definitions of these metrics are

$$\text{Support}, \quad s(X \to Y) = \frac{\sigma(X \cup Y)}{N};$$

$$\text{Confidence}, \quad c(X \to Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}.$$

By referring to the example of Table 3.2, with $X = \{3, 7\}$ and $Y = \{2\}$, for the rule $R = X \to Y$ we have $s(R) = 0.5$ and $c(R) = 1$.

Most data mining software offers a procedure to generate all association rules with confidence and support that exceed some user specified minimum thresholds *minsup* for support, and *minconf* for confidence. A common strategy adopted by many association rule mining algorithms is to decompose the problem in two major subtasks:

1. **Frequent itemset generation,** whose objective is to find all the itemsets that satisfy the *minsup* threshold. These itemsets are called frequent itemsets.

2. **Rule generation,** whose objective is to extract all the high confidence rules from the frequent itemsets found in the previous step. These rules are called *strong rules*.

The computational requirements for frequent itemset generation are generally more expensive than those of rule generation. A lattice structure can be used to enumerate the list of all possible itemsets. A brute force approach for finding frequent itemsets is to determine the support count for every candidate itemset in the lattice structure; this approach can be very expensive, because we need to compare each candidate against every transaction. If the candidate is contained in a transaction, its support count will be incremented.

By considering a data set containing d items, we have that the total number of itemsets is 2^d and the total number of possible rules extracted from the data set is given by the following expression:

$$R = \sum_{k=1}^{d-1} \left[\binom{d}{k} \sum_{j=1}^{d-k} \binom{d-k}{j} \right] = 3^d - 2^{d+1} + 1.$$

For example, for $d = 6$ we have $R = 602$ rules. The number of comparisons in the brute force approach is a $O(NMw)$, where N is the number of transactions, w the maximum transaction width and $M = 2^k - 1$.

The most used technique to reduce the computational complexity of frequent itemsets generation is based on the *Apriori* principle, introduced with the corresponding algorithm by the authors in [2], and which allows to eliminate some of the candidate itemsets without counting their support values.

***Apriori* principle.** *If an itemset is frequent, then all of its subsets must be frequent.*

Conversely, if an itemset is infrequent, then all of its supersets must be infrequent too. Hona et al. [44] developed the min-*Apriori* algorithm for finding association rules in continuous data without discretization. Considerations about the time and storage complexity of the Apriori algorithm are in [33], while other algorithms for mining frequent itemsets include the DHP (dynamic hash and pruning) algorithm proposed in [70] and the Partition algorithm developed in [83]. The method used by the *Apriori* algorithm to generate frequent itemsets is showed in Table 3.9.

For the pseudo code of the frequent itemset generation part of the *Apriori* algorithm and for a detailed analysis of association analysis we refer to [92]; we find an implementation of the *Apriori* algorithm in `WEKA` [94]. Many algorithms have been developed based on the *Apriori* principle; one of the most serious problems is to reduce the number of rules generated during the execution of the algorithm. An overview about this aspect is treated in the last part of the following section.

`Apriori` algorithm

1: Let k=1.
2: Generate frequent itemsets of length 1.
3: **repeat**
4: Generate length $(k+1)$ candidate itemsets from length k frequent itemsets.
5: Prune candidate itemsets containing subsets of length k that are infrequent.
6: Count the support of each candidate by scanning the Database.
7: Eliminate candidates that are infrequent, leaving only those that are frequent.
8: **until** No new frequent itemsets are identified.

Figure 3.9: The *Apriori* algorithm.

3.6 FREQUENT PATTERN OR SEQUENTIAL PATTERN ANALYSIS

When an ordinal relation, usually based on temporal precedence, exists among events occurring in the data of a transactional database, we are dealing with *sequential patterns*. If we refer to a table of the database containing the transactions, each row records the occurrences of events associated with a particular object at a given time. For example, we can have a record indicating that at timestamp $t = 1$ the customer A bought the products P1 and P2. Such information can be used for identifying recurring features of a dynamic system or predicting future occurrences of certain events.

Since its introduction, *sequential pattern mining* has become an important data mining tool, and it has been used in a broad range of applications, including the analyses of customer purchase behavior, disease treatments, Web access patterns, DNA sequences and many others. The problem is to find all sequential patterns with higher, or equal support to a predefined minimum support threshold in a data sequence database, similarly to what introduced for association analysis. The support of a sequential pattern is the number, or percentage, of data sequences in the database containing that pattern. Different techniques and algorithms have been proposed to improve the efficiency of this task.

In this section we present the basic concepts of sequential patterns analysis. A *sequence* is an ordered list of *elements* and can be denoted as

$$s = \langle i_1 i_2 i_3 \dots i_m \rangle ,$$

where each element i_j is a collection of one or more *events*, i.e.,

$$i_j = \{e_1, e_2, \dots, e_{k_j}\} , \quad j = 1, \cdots , m.$$

The events of an element correspond to the same temporal information, that is, occur at the same time. The *length of a sequence* corresponds to the number

of elements in the sequence, while a k-*sequence* is a sequence that contains k events.

sequence *s*	**sequence *t***	**Is *t* a subsequence of *s*?**
$\langle\{2,4\}\{3,5,6\}\{8\}\rangle$	$\langle\{2\}\{3,6\}\{8\}\rangle$	Yes
$\langle\{2,4\}\{3,5,6\}\{8\}\rangle$	$\langle\{2\}\{8\}\rangle$	Yes
$\langle\{1,2\}\{3,4\}\rangle$	$\langle\{1\}\{2\}\rangle$	No
$\langle\{2,4\}\{2,4\}\{2,5\}\rangle$	$\langle\{2\}\{4\}\rangle$	Yes

Table 3.3: Sequences and subsequences.

A sequence t is a *subsequence* of another sequence s if each ordered element in t is a subset of an ordered element is s. Formally, the sequence $t = \langle t_1 t_2 \dots t_m \rangle$ is a subsequence of $s = \langle s_1 s_2 \dots s_n \rangle$ if integers $1 \leqslant j_1 \leqslant j_2 \leqslant \dots \leqslant j_m \leqslant n$ exist such that $t_1 \subseteq s_{j_1}$, $t_2 \subseteq s_{j_2}, \dots, t_m \subseteq s_{j_m}$. If t is a subsequence of s, then we say that t is *contained* in s. Table 3.3 gives examples illustrating the idea of subsequences for various sequences.

Let D be a data set containing one or more *data sequences,* that are ordered lists of events associated with single data object. Similarly to the association analysis, the *support* of a sequence s is the fraction of all data sequences that contain it. If the support for s is greater than or equal to a user-specified threshold `minsup`, then s is a *sequential pattern* or *frequent sequence*. Given a data set of sequences D and a user-specified minimum support threshold `minsup`, the task of *sequential pattern discovery* is to find all sequences with support $\geqslant$ `minsup`.

Table 3.4 illustrates the situation presented in Table 3.2 by considering the temporal information. In particular, the table contains the fields **Student-Id**, **Semester** and **Exams** which represent the student identifier, the temporal information and the list of identifiers for the exams, respectively. For example, student with identifier 10 has taken exams with identifier 3, 2 and 7 in the semester 1 and exam 1 in the semester 2.

According to Table 3.4, the identifier of an exam can be seen as an event and a list of exams can be seen as an element. The sequence $\langle\{3,2,7\}\{1\}\{4,5\}\rangle$ is associated to student 10 and represents a portion of his (her) career. This sequence has length 3 and corresponds to a 6-sequence. Instead, if D is given by the careers of students 10, 20, 30, and 40,

$$s_{10} = \langle\{3,2,7\}\{1\}\{4,5\}\rangle$$

$$s_{20} = \langle\{3\}\{7,2\}\{4,1\}\rangle$$

$$s_{30} = \langle\{4,7\}\{1\}\rangle$$

$$s_{40} = \langle\{2,1\}\{3,5\}\rangle$$

Id-Student	Semester	Exams
10	1	3, 2, 7
10	2	1
10	3	4, 5
⋮	⋮	⋮
20	1	3
20	2	7, 2
20	4	4, 1
⋮	⋮	⋮
30	2	4, 7
30	3	1
⋮	⋮	⋮
40	2	2, 1
40	3	3, 5
⋮	⋮	⋮

Table 3.4: Database of sequences.

the new sequence $s = \langle\{1\}\{5\}\rangle$ has support 0.5 in D since s is a subsequence for s_{10} and s_{40}.

Several algorithms were proposed in the literature for the generation of sequential patterns, all based on concept related to the generation of association rules, examined in Section 3.5.1. In [89] the authors illustrate the method for handling concept hierarchy using extended transactions, while an alternative algorithm was proposed in [45], where frequent data items are generated one level at a time. One of the problem related to searching frequent patterns is the high number of patterns that can be generated from data and that we have to analyze. If the minimal support threshold is set too low, or the data is highly correlated, the number of frequent patterns itself can be prohibitively large. To overcome this problem, several proposals have been made to construct a concise representation of the frequent patterns, instead of mining all frequent patterns. About these considerations we refer to [19, 20], where the authors illustrate how in many cases, first mining the concise representation and then creating the frequent itemsets from this representation outperforms existing frequent set mining algorithms.

The association analysis formulation described is based on the premise that the presence of an item in a transaction is more important than its absence. As a consequence, patterns that are rarely found in a database are often considered

to be uninteresting and are eliminated using the support measure. Such patterns are known as *infrequent patterns*. Although a vast majority of infrequent patterns are uninteresting, some of them might be useful to the analysis, particularly those that correspond to negative correlations in the data. The problem of mining infrequent patterns has been investigated by many authors. For example, in [84] the problem of mining negative association rules is examined using a concept hierarchy, while in [91] the idea of mining indirect associations for sequential and non-sequential data is proposed. There have been proposed quite a few interesting and effective algorithms, such as CLOSET [73], MAFIA [18], CHARM [97] and `CloSpan` developed for efficient mining of frequent closed itemsets. ; in particular, CHARM regards itemsets mining and it represents an important study in this field.

In the following sections we illustrate the main technical characteristics of the two algorithms used in this thesis, `SPAM` and `CloSpan`, to find sequential patterns.

3.6.1 *SPAM*

`SPAM` (Sequential Pattern Mining) is an efficient algorithm integrating a variety of algorithmic contributions into a practical tool, described in [7]. The algorithm is based on the construction of a lexicographic tree of sequences which is traversed in a depth-first manner and in which the root is conventionally labeled 0. An additional feature of `SPAM` is its property of online outputting sequential patterns of different length (compare this to a breadth-first search strategy that first outputs all patterns of length one, then all patterns of length two, and so on).

The implementation of `SPAM` we consider uses a vertical bitmap data layout allowing for simple and efficient counting. For the sake of clarity, we illustrate the main concepts for understanding how the algorithm `SPAM` works and refer to the above cited bibliography for a complete description.

A *sequence-extended sequence* is a sequence generated by adding a new transaction consisting of a single item to the end of its parent's sequence in the tree. An *itemset-extended sequence* is a sequence generated by adding an item to the last itemset in the parent's sequence, such that the item is greater than any item in the last itemset. For example, if we have a sequence $s_a = (\{a, b, c\}, \{a, b\})$, then $(\{a, b, c\}, \{a, b\}, \{a\})$ is a sequence-extended sequence of s_a and $(\{a, b, c\}, \{a, b, d\})$ is an itemset-extended sequence of s_a.

Each sequence in the sequence tree can be considered as either a sequence-extended sequence or an itemset-extended sequence. If sequences are generated by traversing the tree, each node in the tree can generate sequence-extended children sequences and itemset-extended children sequences.

The algorithm SPAM generates sequence-extended sequences by the *sequence-extension step* (*S-step*) and itemset-extended sequences by the *itemset-extension step* (*I-step*). Thus each node n in the tree has two associated sets: S_n, the set of candidate items that are considered for a possible *S-step* extension of node n, and I_n, which identifies the set of candidate items that are considered for a possible *I-step* extension.

The other important phase of SPAM is pruning, based on the algorithm *Apriori* and aimed at minimizing the size of S_n and I_n at every node n. The technique guarantees that all nodes corresponding to frequent sequences are visited. The S-step and I-step pruning phases go as follows.

The first technique prunes S-step children. Consider a sequence s at node n and suppose its sequence-extended sequences are $s_a = (s, \{i_j\})$ and $s_b = (s, \{i_k\})$. Suppose s_a is frequent but s_b is not frequent. By the Apriori principle, $(s, \{i_j\}, \{i_k\})$ and $(s, \{i_j, i_k\})$ can not be frequent, since both contain the subsequence s_b. Hence, we can remove i_k from both S_m and I_m, where m is any node corresponding to a frequent sequence-extended child of s.

The second technique prunes I-step children. Consider the itemset-extended sequences of $s = (s', \{i_1, ...; i_n\})$. Suppose the sequences are $s_a = (s', \{i_1, ..., i_n, i_j\})$ and $s_b = (s', \{i_1, ..., i_n, i_k\})$, and suppose that $i_j < i_k$. If s_a is frequent and s_b is not frequent, then by the Apriori principle, $(s', \{i_1, ..., i_n, i_j, i_k\})$ cannot be frequent. Hence, we can remove i_k from I_m, where m is any node corresponding to a frequent itemset-extended child of s. Finally we can remove from S_m the same items as we did using S-step pruning. Suppose that during S-step pruning $(s, \{i_l\})$ was discovered to be infrequent. Then $((s_a, \{i_l\})$ will be infrequent for any frequent itemset-extended child s_a of s.

3.6.2 *CloSpan*

CloSpan is an algorithm for mining closed sequential patterns in a sequence database. Given a collection of sequences and a minimum support threshold, it is able to find all of the subsequences whose frequency is above the threshold. When mining long frequent sequences, or when using very low support threshold, the performance of efficient methods for mining frequent patterns often deteriorates dramatically. An interesting solution, called mining frequent closed itemsets [71], was proposed to overcome this difficulty. A frequent itemset I is *closed* if there exists no superset of I with the same support in the database. Before CloSpan there have been no efficient methods developed for mining closed sequential patterns. Most techniques developed in closed itemset mining cannot work for frequent subsequence mining because subsequence testing requires ordered matching which is more difficult the simple subset testing. There are two approaches for mining closed or max frequent patterns; the first greedily finds the final closed pattern set, the second finds a closed pat-

tern candidate set and conducts post pruning on it. In [95] authors explore the second approach, by considering that with the today's technology to maintain a great number of sequences in main memory is easy and the sequences have overlapped parts, that is, there exist compressed data structures to store them. The algorithm `CloSpan` develops several efficient search space pruning methods. A novel concept about the equivalence of projected database is introduced, which can unify these optimizations in a single step. A simple condition of such equivalence is formalized and a hash-based algorithm is designed to efficiently execute the search space optimization with negligible cost. The performance of `CloSpan` shows that it not only generates a complete closed subsequences set which is substantially smaller than that generated, for example, by PrefixSpan [74], an algorithm which explores prefix-projection in sequential pattern mining, but also runs much faster.

`CloSpan` divides the mining process into two stages; in the first stage a candidate set, usually larger than the final closed sequence set, is generated. The second stage helps eliminate non-closed sequences. The search space pruning phase of `CloSpan` does not modify the underlying frequent pattern mining algorithm but it only defines the early termination condition of search branches; this method could be extended to other existing sequential pattern mining algorithms, like `SPAM`. We could find that it is feasible to calculate the corresponding size of projected database in `SPAM` with a little additional cost.

In Chapter 5, by comparing some results obtained with `SPAM` and `CloSpan`, we will see that `CloSpan` is effectively better than `SPAM` for what concerns the number of generated patterns.

4

DATA MINING ON STUDENT CAREERS

In this work we propose a methodology, composed by several models, to be applied to a database containing information about students and their exams in a university organization. In particular, for each student, the database contains general information such as the sex, the place of birth, the grade obtained at the high school level, the year of enrollment at the university, the date and the grade of final examination besides information about each exam, that is, the identifier of the exam, the date and the grade. We refer to an organization of the university which allows students to take an exam also many times, in different sessions, after the end of the course, as in Italy; we do not have information about exams retakes, that is, we do not know if students fail some trials, although this seems to be highly relevant for assessing the success rate of the educational. Some constrains between exams can be fixed in order to force students to take some exams in a specific order; however, usually students have many degrees of freedom to choose their own order of exams. An important information which is a basilar aspect of our methodology is the *semester*; an academic year is divided into two semesters, during which the courses are taken according to the established curriculum. A student can take an exam in the same semester of the course, that is, just after the end of the course, or later, with a delay of one or more semesters. As we illustrate in Chapter 5, we need to look at several heterogeneous sources to find the necessary information to insert in the student database the temporal information, that is, the semester of each course. This phase, such as we presented in Section 2.1 when we considered the design of a datawarehouse, is very important to obtain a good level of integrated, corrected and datailed data, over which we can perform a Data Mining analysis.

4.1 CLUSTERING MODEL

The information on the semester just introduced allows us to define an *ideal path* and to compare it with the path of a generic student. More precisely, we consider a database containing the data of N students, each student characterized by a sequence of n exams identifiers and a particular path $\mathcal{I} = (e_1, e_2, \cdots, e_n)$, the *ideal path*, corresponding to the ideal student who takes every examination just after the end of the corresponding course, without delay. Since in the same semester there are many courses, the ideal path is not unique. In this thesis we consider courses relative to the same semester sorted according to the prefer-

ence of students, that is, in the ideal path the order of the exams corresponding to courses taken in the same semester is chosen according to the order preferred by students. In this case, the number of semesters is not important to determine the ideal path, but it is sufficient to know the number of exams. For example, if the degree program has 3 semesters and 7 exams, two in the first semester, three in the second and two in the third, we can represent the ideal career by the sequence $(1,2,3,4,5,6,7)$; in this sample, for what concerns the exams of the first semester, the code 1 was assigned to the exam that students preferred to take before the other one, that is, with code 2. Similar considerations can be made for the exams of the other two semesters. A different solution consists in giving the same temporal identifier to courses in the same semester; in this case if we have s semesters, we can assign to the exams the values varying from 1 to s . For example, such as in the previous example, with $s = 3$, the sequence $(1,1,2,2,2,3,3)$ would represent an ideal path of 7 exams, two in the first and third semester and three in the second.

Without loss of generality, we can assume that $e_i = i$, $i = 1, \cdots, n$, that is, $\mathcal{I} = (1,2,\cdots,n)$, and we can choose to represent the ideal path with the *identity permutation* of integers from 1 to n.

The path of a student $\mathcal{J}$ with $\mathcal{J} = 1, \cdots, N$, can be seen as a sequence $\mathcal{S}_\mathcal{J} = (e_{\pi_\mathcal{J}(1)}, e_{\pi_\mathcal{J}(2)}, \cdots, e_{\pi_\mathcal{J}(n)})$ of n exams, where $e_{\pi_\mathcal{J}(i)}$, $i = 1, \cdots, n$, is the identifier of the exam taken by the student $\mathcal{J}$ at time i and $\pi_\mathcal{J}$ indicates the corresponding permutation of $1, \cdots, n$. Therefore, $\mathcal{S}_\mathcal{J}$ can be seen as a permutation of the integers 1 through n.

As illustrated in [22], the idea is to understand how the order of the exams affects the final result of students. To this purpose, we compare a path $\mathcal{S}_\mathcal{J}$ with $\mathcal{I}$ by using the *Bubblesort distance*, which is defined as the number of exchanges performed by the Bubblesort algorithm to sort an array containing the numbers from 1 to n. The number of exchanges can be computed easily since it corresponds exactly to the number of inversions in the permutation. Given a permutation $\pi = (\pi_1, \pi_2, \cdots, \pi_n)$ of the integers 1 through n, an *inversion* is a pair $i < j$ with $\pi_i > \pi_j$. If q_j is the number of $i < j$ with $\pi_i > \pi_j$ then $q = (q_1, q_2, \cdots, q_n)$ is called the *inversion table* of π [61]. We use the notation $\sigma(\pi)$ to denote the number of inversions in the permutation, that is, the sum of the entries in the inversion table: $\sigma(\pi) = \sum_{j=1}^{n} q_j$. For the position above, this also represents the Bubblesort distance of π from the identity permutation. For example, the permutation $\pi = (5,2,3,1,4)$, as illustrated in Table 4.1, corresponds to $q = (0,1,1,3,1)$ and $\sigma(\pi) = 6$.

The Bubblesort distance is bounded above by $n(n-1)/2$; moreover, if we assume a uniform distribution over the $n!$ permutations, then the average number of inversion is $n(n-1)/4$ (see, e.g., [86]).

A second approach for understanding how the order of the exams affects the results of students concerns the graphical representation of the careers; we represent them in the integer lattice, where the x-axis denotes the time and

index	1	2	3	4	5
π	5	2	3	1	4
q	0	1	1	3	1

Table 4.1: The inversion table for permutation $(5, 2, 3, 1, 4)$.

$S_{\mathcal{J}}$	$\sigma(S_{\mathcal{J}})$	$\mathcal{A}_{S_{\mathcal{J}},\mathcal{I}}$
(2,1,3,4,5)	1	1.5
(2,1,3,5,4)	2	3
(1,2,5,4,3)	3	4
(2,4,1,3,5)	3	5
(3,5,1,2,4)	5	8.8
(5,2,3,1,4)	6	8
(5,4,3,2,1)	10	12

Table 4.2: Some careers and their distances from the ideal career for $n = 5$.

the y-axis the sequence of the exams according to the order of the ideal career. We precise that it is possible that two students have the same ideal career but they end their studies in different times; it is also possible that two students end their studies in the same time, but in a different order. Our model only considers the order in which exams are taken by students, without considering the exact time. The ideal career is defined by the sequence of points

$$\tau_{\mathcal{I}} = ((0, e_0), (1, e_1), (2, e_2), \cdots, (n, e_n), (n+1, e_{n+1})),$$

where $e_0 = 0$ denotes the starting point of the career and $e_{n+1} = n+1$ denotes the final examination given last by all students. Therefore, $\tau_{\mathcal{I}}$ can be represented as the bisecting line of the first quadrant. The career of a generic student $\mathcal{J}$ is then represented by a broken line corresponding to the sequence of points

$$t_{\mathcal{J}} = ((0, e_{\mathcal{J}_0}), (1, e_{\mathcal{J}_1}), (2, e_{\mathcal{J}_2}), \cdots, (n, e_{\mathcal{J}_n}), (n+1, e_{\mathcal{J}_{n+1}})).$$

i	e_i
1	*Calculus*
2	*Programming*
3	*Algorithms and Data Structures*
4	*Databases*
5	*Theoretical Computer Science*

Table 4.3: A sample scenario for the careers in Table 4.2.

By convention, we have $e_{\mathcal{J}_0} = 0$ and $e_{\mathcal{J}_{n+1}} = n+1$ for every student $\mathcal{J}$, that is, the resulting trajectory begins at $(0,0)$ and terminates at the point $(n+1, n+1)$, that is, the sequence is artificially extended with the final exam e_{n+1}, marking the end of the sequence. We then compare the career of a student $\mathcal{J}$ with $\mathcal{I}$ by computing the area between $t_{\mathcal{J}}$ and $\tau_{\mathcal{I}}$, that is, the sum of the areas of the triangles between the two lines, as follows:

$$\mathcal{A}_{\mathcal{J},\mathcal{I}} = \sum_{i=0}^{n} \int_{i}^{i+1} |(x-i)(e_{\mathcal{J}_{i+1}} - e_{\mathcal{J}_i}) + e_{\mathcal{J}_i} - x| dx$$

In Table 4.2 we illustrate the careers of 7 students for $n = 5$ and their distances from the ideal career $(1,2,3,4,5)$; Table 4.3 gives a sample scenario. Two of these careers are illustrated in Figures 4.1 and 4.2. It is evident from the graphical representation that the career $(2,1,3,4,5)$ is closer to the ideal career than the career $(2,1,3,5,4)$. The first career has $\sigma = 1$ and $\mathcal{A} = 1.5$, while the second one has $\sigma = 2$ and $\mathcal{A} = 3$. We can conclude that the two representations are similar but not identical. We put in evidence that it is possible to have students ending in the same time but in different order, but in our case study, such as we will present in Chapter 5, this does not happen. For a generic student $\mathcal{J}$ we

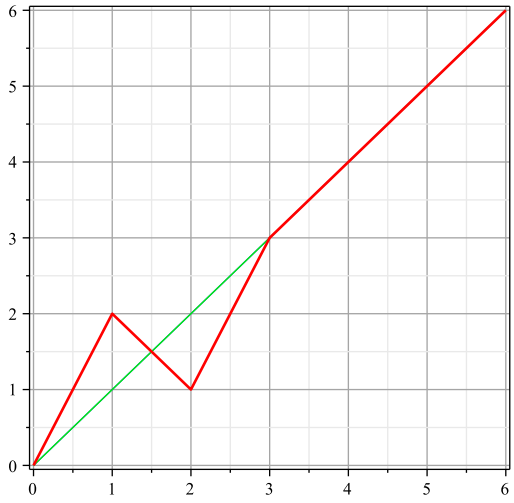


Figure 4.1: The career $(2,1,3,4,5)$.

compute $\sigma(\mathcal{S}_{\mathcal{J}})$ and $\mathcal{A}_{\mathcal{J},\mathcal{I}}$, $\mathcal{J} = 1, \cdots, N$; these new values are inserted into the database in order to compare the career of the generic student with the ideal path $\mathcal{I}$.

After this preprocessing phase, we can assume that for each student our database contains at least the following information: the graduation time, that is, the total length of study, **Time**, the final grade, **FinalGrade**, the **Bubblesort** distance and the **Area** distance, together with other personal information. We

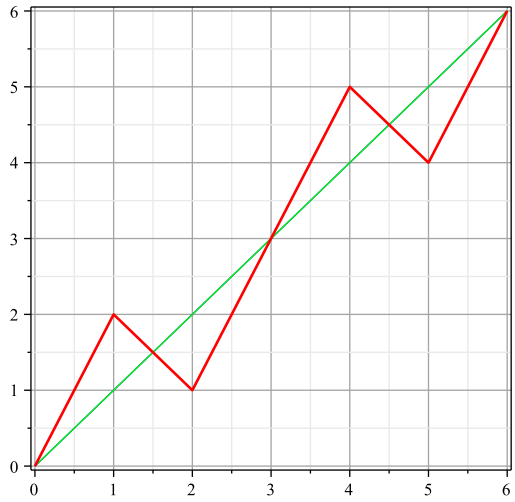


Figure 4.2: The career $(2, 1, 3, 5, 4)$.

can proceed by applying a cluster algorithm, for example `K-means` (see Chapter 3). We wish to point out that the Bubblesort and Area distances are new attributes inserted in our database, such as any other, for example the year of enrollment, the mark obtained at the high school, and so on; we use an implementation of `K-means` that uses the Euclidean distance to assign each *point* to the appropriate *centroid*; at the step 3 of the algorithm described in Table 3.4, for each point the closest centroid is found by computing the Euclidean distance.

If the cluster algorithm splits the students into K well defined groups characterized by similar Bubblesort distance and/or Area distance, we can infer important conclusions about students and the degree program. We observe explicitly that students who have taken the exams in the same order, that is, students with the same path, can have different final grade and graduation time. The idea is to understand if there exists a relation between the two distances and the success of students, that is, if students taking the exams with an order which is similar to the ideal path, obtain better results than the other students. In other words, if the students corresponding to paths having small distance from the ideal path achieve good performance, then, for example, we may conclude that the academic degree is well structured. However, if there exist many good students with large distances, and some of them have good performance, then we can infer that some courses may be scheduled in an incorrect order and the organization should probably be modified. These are some considerations that we can infer from the proposed clustering analysis; we will examine this issue when we will present the results of the case study in Chapter 5.

4.2 SEQUENTIAL PATTERN MODEL

As far as we know, there are no many works based on sequential pattern mining in the context of education. Recently, in [72] the authors discuss student and education responsible perspectives on curriculum mining and present the achievements of the ongoing project aiming to develop curriculum mining software including process mining, data mining and visualization techniques, by using pattern sets.

Our methodology [24] involves the use of the `SPAM` algorithm proposed in [7], an implementation of which can be found in [87], or the more efficient `CloSpan` algorithm [27, 95], which finds the closed patterns, that is, those containing no super-sequence with the same support. Therefore, the career of a student can be seen as a sequence $e_1 e_2 \dots e_n$ of n exams, where e_k precedes e_j in the sequence if e_k has been taken at the same time or before e_j. The temporal information allows us to see the career of a student as a sequence $\langle i_1 i_2 \dots i_m \rangle$ where each element i_j is a collection of one or more exams taken in the same semester or having the same delay. We illustrate a general methodology which can be applied to any kind of university degree, provided a preprocessing phase is performed on the original data to take into account temporal information.

In Section 3.6 we described the sequential pattern theory. We now explain our methodology based on sequential pattern analysis, by using the algorithm `SPAM` or the algorithm `CloSpan`. This second algorithm is more efficient than the first one, because it produces a smaller number of frequent patterns than those produced by `SPAM`, such as we illustrated in Chapter 3, Section 3.6.2. We also present how the student data have been processed to perform a pattern analysis. Our model provides two different steps; the first for the generation of the sequential patterns, the second for the selection of the most interesting among them.

4.2.1 *The methodology*

In order to study sequential patterns in any student database, we consider the *career* of a student, that is, the way the student implements her or his exams over the degree-learning time: a student can take an exam immediately after a course (the *ideal* choice) or later. We study the career of students and compare one another. As we pointed out in the Introduction, the sequential pattern technique has been introduced in [3] and has become an important method in data mining (see, e.g., [92]).

In particular, we consider as events the exams taken by a student; the temporal information is the *semester* in which the exam has been taken or the *delay* with which it has been taken.

As already observed, we consider an organization of the university which allows students to take an exam in different sessions after the end of the course, as in Italy; we refer to the career of a student such as a sequence $e_1 e_2 \dots e_n$ of n exams, as we introduced just before. By analyzing the sequential patterns, we can explain some behaviors which may seem counterintuitive, e.g., course x is scheduled before course y while many students take exam y before x. Such information may be helpful for changing course schedules or to find out those courses whose exams are considered difficult by the students and thus could give insight for reorganizing the curricula.

In Section 4.1 we have defined the *ideal career* as the sequence of n exams taken by an ideal student; in this representation the order of exams relative to courses given in the same semester is the one chosen by most students. By convention, we identified the ideal career by the identity permutation, that is, we identified the first exam in the sequence by code 1, the second by code 2, and so on. The career of a generic student was therefore a permutation of the integers 1 to n.

By representing the career of students by permutations we only consider the order in which the exams have been taken by students. Sequential pattern analysis, by dealing the temporal information, allows us to study the careers in greater depth than using the clustering.

Finding sequential patterns in large databases is an important data mining problem. There are several algorithms implementing techniques for finding frequent sequences based on the *Apriori* principle [92], introduced in Chapter 3, according to which if an itemset is frequent, then all of its subsets must also be frequent, or, if an item set is infrequent then all its supersets must also be infrequent. The algorithms `SPAM` and `CloSpan`, which are used in our model, belong to this category.

Id-Student	**Semester**	**Exam**
10	1	3
10	1	2
10	1	7
10	2	1
⋮	⋮	⋮
20	1	3
20	2	7
20	2	2
⋮	⋮	⋮

Table 4.4: Example of data input for `SPAM`.

Frequent sequences
1 - 4
1 -1 5 - 2
2 - 3
2 -1 1 - 2
2 -1 4 - 2
2 -1 5 - 2
2 7 - 2
2 7 -1 1 - 2
2 7 -1 4 - 2
3 - 3
3 -1 1 - 2
3 -1 4 - 2
4 - 3
5 - 2
7 - 3
7 -1 1 - 3
7 -1 4 - 2

Table 4.5: Output of SPAM on data of Table 4.4.

Table 4.5 shows the output obtained by using SPAM with a support equal to 0.5 from the data of Table 3.4, in the format accepted by SPAM and illustrated in Table 4.4. Each line of the output file is a frequent sequence and can be interpreted as follows. The last number is the frequency of the sequence; the data in Table 4.5 correspond to the output obtained by running SPAM by considering an input data set of 4 students and by choosing minsup equal to 0.5; with these parameters we obtain all the sequences that have frequency $\geqslant 2$. The number -1 between two numbers, representing two exam codes, indicates that the two exams are taken in different times (semesters, if we use this temporal information), that is, the first exam is taken one or more semesters before the second; the symbol $-$ indicates the end of the sequence. For example, the first line of the output indicates that 4 students have taken exam 1; the eighth line indicates that 2 students have taken exams 2 and 7 in the same semester and then have taken exam 1 in a later semester. In this example, only the exam with code 4 has been taken by all students; in fact only the first line has a support $= 4$. Besides we can observe that the longest patterns are 3-sequence, introduced in Section 3.6, corresponding to the eighth and ninth lines; they are two 3-sequences of length 2 (they involve 2 elements).

The example illustrated in Tables 4.4 and 4.5 refers to the use of sequential pattern analysis where the temporal information corresponds to the *semester* in which the student takes exams. Another temporal information which can be used is the *delay* with which a student takes exams. This delay is expressed in semesters and is the difference between the semester in which the student takes an exam and the semester in which the course has been given by the teacher. The frequent patterns obtained by using one or the other temporal information have a different meaning. With the semester, we obtain patterns emphasizing the order with which students prefer to take exams; with the delay, we obtain patterns grouping the exams according to the delay with which they are taken by students. These patterns can be used to understand which exams are more difficult for the students or, on the contrary, do not present great difficulties. Therefore these results can suggest to modify the schedule of the degree program or confirm that the degree program is well structured. We observe explicitly that the use of specific temporal information changes the representation as a sequence of the ideal career introduced in Section 4.2.1. Table 4.6 shows an example of preprocessed data with the two temporal information, by referring to the careers of students in Table 3.4, where **Semester1** is the semester in which the course was given by a teacher and **Semester2** is the semester in which student took the corresponding exam. For example, with the semester, the student 10 has career $c_{10}^{[s]} = \langle\{3,2,7\}\{1\}\{4,5\}\rangle$, where the apex s indicates the semester; with the delay, the same student has career $c_{10}^{[d]} = \langle\{3,2,7,1\}\{4,5\}\rangle$, where the apex d indicates the delay. It is important to note that in our analysis we consider only a gap of semester (or delay), regardless of the exact values of the gap, which can be either a semester or more semesters.

More precisely, if n is the number of exams, S the number of semesters in which the degree program is organized and p_j the number of exams corresponding to the semester j, then the ideal career corresponds to the sequence

$$c_I^{[s]} = \langle i_1 \dots i_S \rangle, \quad i_j = \left\{ \sum_{k=1}^{j-1} p_k + 1, \dots, \sum_{k=1}^{j-1} p_k + p_j \right\},$$

with $\sum_{j=1}^{S} p_j = n$, where the temporal information is the semester. Where we consider the delay as temporal information, the ideal career corresponds to the sequence

$$c_I^{[d]} = \langle\{1, \dots, n\}\rangle .$$

By considering a career of $n = 12$ exams, with 4 semesters and $p_1 = p_3 = 3$, $p_2 = 4$, $p_4 = 2$ we obtain $c_I^{[s]} = \langle\{1,2,3\}\{4,5,6,7\}\{8,9,10\}\{11,12\}\rangle$ and $c_I^{[d]} = \langle\{1, \dots, 12\}\rangle$.

In Chapter 5, we examine a real student database by using both approaches.

Id-Student	Exam	Semester1	Semester2	Delay
10	3	1	1	0
10	2	1	1	0
10	7	1	1	0
10	1	2	2	0
10	4	1	3	2
10	5	1	3	2
20	3	1	1	0
20	7	1	2	1
20	2	1	2	1
20	4	1	4	3
20	1	2	4	2
⋮	⋮	⋮	⋮	⋮

Table 4.6: An example of postprocessed data for a student database.

4.2.2 *Clustering on sequential patterns*

By using one of the two algorithms presented in Chapter 3, `SPAM` or `CloSpan`, a fundamental step of our model is searching the frequent patterns that satisfy the specified threshold *minsup* and consists in considering one of the two temporal information associated to each exam. Among the sequential patterns found, we have to discover the most meaningful, that is, those having a higher support or showing a particular behavior. We stress that this phase cannot be completely automated and requires a deep knowledge of the context under examination. In particular, it is mandatory to know the organization of the degree program and the schedule of courses.

An obvious criterion is to consider the patterns of greater length or involving the maximum number of exams, and between them, to select those most regular, for example corresponding to subsequences of the ideal career; alternatively we can consider the patterns showing unexpected irregularities. Since we chose to denote the ideal career as the identity permutation, it is simple to discover patterns which are ordered subsequences of the ideal career. For example, according to the Section 4.2.1, if $c_I^{[s]} = \langle\{1,2,3\}\{4,5,6,7\}\{8,9,10\}\{11,12\}\rangle$, then the sequence $\langle\{3\}\{5,7\}\rangle$ is obviously a subsequence of c_I while $\langle\{8\}\{5\}\{7\}\rangle$ is not. The pattern selection step can be simplified by processing the result of `SPAM` (or `CloSpan`) to obtain a more readable format in which the frequent sequences are sorted according to the number of exams and to the length.

The database may contain several kinds of information about students. For example, for each student we can have the grade obtained at the high school

level, the type of high school, the year of enrollment at the university, the mark of final examination and the length of the studies.

The next step consists in inserting in the database the information relative to the sequential patterns previously identified; in practice, every student verifying the pattern P has assigned value 1, 0 otherwise. The aim is to understand if students satisfying the patterns have some common characteristics. In Table 4.7 we illustrate a sample scenario corresponding to the patterns $\langle\{1\}\{5\}\rangle$, $\langle\{7\}\{5\}\rangle$, $\langle\{7\}\{1\}\rangle$ and the information about the exams of students showed in Table 3.4, where the temporal information is the semester. Once the database has been updated in this way, we can perform a clustering analysis to find out if there is any correlation between student attributes and the sequential patterns. We could find that students verifying some patterns corresponding to particular subsequence of the ideal career obtain better results, that is, a high grade in a small time, than students not verifying them; if this happens we could conclude that the degree program is well structured because students following it have good performances. In Chapter 5, we apply this methodology to a real case study.

Id-Student	$\langle\{1\}\{5\}\rangle$	$\langle\{7\}\{5\}\rangle$	$\langle\{7\}\{1\}\rangle$
10	1	1	1
20	0	0	1
30	0	0	1
40	1	0	0

Table 4.7: Some patterns for Table 3.4.

4.3 CLASSIFICATION MODEL

We can extend the analysis presented in Section 4.1 through the technique based on decision trees. To this purpose, we need to add to the database a new attribute *class* which labels the students into different ways; we can choose the `Bubblesort_class` to label the students into K groups, according to the ranges of values of Bubblesort distance in the K clusters previously found. For example, with $K = 2$, we can assign the label *smallDist* to good students, that is, those having good final results; we assign *largeDist* to not so good students, having worst final results. This new attribute can be used to classify students, for example by using the `C4.5` algorithm, introduced in Section 3.4.1 (see e.g, [92]). The aim is to classify students as talented or not and find the attributes which most influence their career.

We can also try to classify with respect to other attributes: for example, we can predict whether a student obtains a high (low) final grade or has a long

(short) career, by introducing a `Grade_class` or a `Time_class` attribute in the database. The most interesting use of classification techniques is to build predictive models, in which the value of a particular attribute can be predicted by using the values of other attributes. In our analysis we can obtain a predictive model if we choose, for example, the `Grade_class`; by analyzing the values of the other information in the student database, properly preprocessed, we can obtain a model that shows which are the attributes that determine belonging to the *high grade class* rather than *low grade class*.

We can also be interested to descriptive classification models; for example if we choose the `Bubblesort_class`, which labels students into *near* the ideal career and *far* from it, according to the clustering results, we obtain a descriptive model, because the information about the distance which a student takes from the ideal career is related to the final grade and the duration time.

We explicitly observe that the greater are the database and more the information in it, the more accurate will result the model based on this technique.

5

THE CASE STUDY

In this chapter we present the analysis performed by applying the data mining models presented in Chapter 4 on a real case. We consider a university student database, that is, data for a comprehensive management of the entire life-cycle of students by beginning with the complex phase of the entry of students into the training programs of university, then by dealing teaching, courses and examinations; in this sense the database represents the support for the career of students from an administrative and educational point of view. In particular, the management of the careers of students includes information about recruitment and enrollment, from the teaching/learning and the administration point of view. Each university expresses its strategic goals modeling the courses and the education paths; in every university information system there are modules that manage the planning and scheduling of courses; consequently information about the planning of the courses is stored in the database, obtained after an important preprocessing phase. This database can be considered such as a datawarehouse containing all the information about the students and their careers and on which the data mining analysis can be performed. We used the software `WEKA` [94] for the implementation of the *K*`-means` algorithm to perform the clustering analysis. This software, that was developed at the University of Waikato in New Zealand, provides implementations of learning algorithms that we can apply to our datasets. Among algorithms provided by `WEKA`, for our analysis we use also the `J48` algorithm, that is an implementation of `C4.5`, one of the most used algorithms to perform the classification analysis. For the frequent pattern analysis we used `SPAM` and `CloSpan`, that were illustrated in Section 3.6.

5.1 CONTEXT OF ANALYSIS

The context of our analysis is based on students enrolled for the first time at the degree courses in Computer Science and Statistics at the University of Florence (Italy) from 2001-2002 up to 2007-2008 academic years, according to Ministerial Decree n. 509/1999. We consider students who did not enroll before to another degree course; students from other courses than the under consideration ones, waiving the studies or declined, are excluded from the analysis.

Data concern the *laurea triennale*, that is, the academic degrees under analysis are structured in three years; the information is available in two data files, the first containing general information about students, the second containing the

information about all the exams taken by every student. Tables 5.1 and 5.2 illustrate the organization of the data source considered in our analysis.

Field	**Description**
PROG	anonymous identifier of student
RESULT	= AI if student abandoned
	= AT if student is currently enrolled
	= LT if the student is a graduate
IMM-YY	enrollment academic year
IMM-COURSE	enrollment degree program
	= 317 Statistics
	= 371 Computer Science
HS-CODE	highschool
	= 1 Professional school
	= 2 Technical school
	= 3 Educational psychology school
	= 4 Scientific high school
	= 5 Classic high school
	= 6 Linguistic high school
	= 7 Art high school
	= 8 Other school
	= 9 Foreign school
HS-VOTE	final highschool grade
LAU-YY	degree program academic year
LAU-VOTE	degree grade
LAU-DATE	degree date

Table 5.1: The general information about student.

In the source database there are 544 students of which 110 students in Statistics and 434 in Computer Science, while the table of exams has 8469 records. We concentrated our analysis on the graduated students in Computer Science (IMM-COURSE=371 and RESULT=LT in Table 5.1), so that we considered 141 students and their exams. Every year of the academic degree in Computer Science is organized in two semesters; there are several courses in each semester and at the end of a semester students can take their examinations. Students can take exams in different sessions during the same year of the course, after its end, or in later years. In the period under analysis, the Computer Science de-

Field	Description
PROG	anonymous identifier of student
EXA-CODE	exam code
EXA-PROG	exam prog
EXA-DESC	exam description
EXA-YY	academic year of the exam
EXA-DATE	exam date
EXA-VOTE	exam grade in thirtieths; 31 is cum laude
EXA-RAT	rating examination if provided

Table 5.2: Information about exams.

gree had different organizations: during the academic years from 2001-2002 to 2003-2004 each student could choose several exams among five different curricula, while during the other years students could choose between two curricula. In particular, in the first three years, the curricula were:

1.a *Database and Information Systems;*

1.b *Distributed Systems;*

1.c *Numerical Applications;*

1.d *Physics Applications;*

1.e *Information Sciences.*

During the academic years 2004-2005 and 2007-2008 the curricula were:

2.a *Computer Science;*

2.b *Computer Science Applications.*

In our analysis, for the first three years, we consider data of students of the Curricula 1.a and 1.b, because most of the students enrolled during academic years from 2001-2002 to 2003-2004 chose these curricula. In particular, 38 students chose the Curriculum 1.a, 47 students chose the Curriculum 1.b, the remaining 8 students were divided between the other curricula. During the academic years from 2004-2005 to 2007-2008, Curricula 1.a and 1.b were merged into a single curriculum, called Curriculum 2.a; the other old curricula were merged into the new Curriculum 2.b.

Much of this information is not available in the data files from which our analysis began; an important preprocessing phase was required to obtain a useful database for our data mining analysis. In the following sections, we refer to Curriculum 1.a with Curriculum 1 and to Curriculum 1.b with Curriculum 2.

5.2 DATA PREPROCESSING

Several cleaning and transformation operations were necessary to obtain a correct database to perform our analysis. As we presented in Chapter 2.1, many steps have to be performed to obtain corrected, consistent and complete data for a good analysis. First of all, we detected missing and wrong data; for example in the source data files there were some exams having the date successive to the date of the final exam (that is, by referring to fields in Tables 5.1 and 5.2, LAU-DATE $<$ EXA-DATE); in other records the grade of the exam was missing. Every time we corrected the mistake by adopting the best technique according to general rules for data mining; for example if in a record the date of an exam was missing, we inserted in the corresponding record field the date, according to the schedule of courses and the behavior of the students. If the grade of an exam was missing, we assigned the grade according to the mean of the exam grades calculated for all students having taken that exam, and for all exams of the student. In the source database some exams were recorded in different ways compared to the corresponding laboratory tests; in some cases we have a single record and a single grade for the theoretical and the practical exam, in others we find two different records, one for the theoretical exam and the other for the laboratory test. We decided to modify the data to have a unique type of recording, in agreement to the examination involved.

Apart the operations to make the data consistent and complete, a preprocessing phase was necessary to add some indispensable information for our analysis, such as the semester (according to its different definitions) and the curriculum chosen by each student.

After organizing the source data in relational tables and solving the problems of bad or missing data, we performed several steps by modifying data and adding new information.

Different curricula

The preprocessing phase aims to obtain a database describing the career of each student as a sequence of exams. We needed the following information:

- for each curriculum, the established exams and their schedule in years and in semesters;
- for each student, the chosen curriculum;
- for each student, the list of courses with the semester in which they were taken, and the list of the corresponding exams, sorted by the semester in which they were taken.

We found some important information by looking at several heterogeneous sources, such as paper documentation relative to the academic years under

consideration and old on-line documents about the organization of the degree program. For example, by compiling the list of the required exams of a specific curriculum, we were able to assign a curriculum to every student; by considering the schedule of each curricula and the enrollment year of each student, we assigned to each exam of each student the semester in which the student would have taken the course and taken the corresponding exam.

For each curriculum we formulated the *ideal career*, as introduced in Chapter 1 and defined precisely in Sections 4.1 and 4.2.1; let us remember that the ideal career is the sequence of exams that corresponds to the ideal student, who takes every exam at the end of the corresponding course, without delay. There are several particular cases; for example, if a student delays every exam exactly the same time, his/her career, in terms of sequence of ordered exams, will be equal to the ideal career. Another special case can be if a student takes exams in the reverse order to that of the ideal career; with data of our case study these cases do not happen but are possible and, such as we will see, do not modify our clustering analysis.

The semester

The methodologies presented in Chapter 4 are based on the careers of students expressed in terms of the semester in which each exam is taken by the students. This information is fundamental for our purposes; its level of granularity is the best for our analysis because it allowed to uniform all the years considered, and because the semester represents the actual and useful period in which an academic year is divided. For each exam and for each student, we inserted in the source database several temporal information about the semester: the semester of the course, that is, the semester in which the course was given by the teacher; the semester of the exam, that is, the semester in which the student has taken the exam, and the delay, that is, the difference between the previous two semesters.

Data for analysis

Tables 5.3 and 5.4 show an example of the most important information in our final database. In particular, in Table 5.3 the column **HSchool** is the result of a transformation, because in the source database we had numerical codes to indicate the different types of high school, such as shown in Table 5.1; we first defined groups of high schools, then we associated a label to each group. We will illustrate in the next sections that, for example, in classification analysis we need discrete attributes. The column **Curriculum** corresponds to a new attribute, inserted in the database after that a curriculum was assigned to every student.

In Table 5.4 the attribute **Exam** indicates the new encoding of exams, that can be a numerical or alphabetical code; in the preprocessing phase we assigned

Id-Student	Enrollment	Curriculum	HSchool	HGrade	Date	FinalGrade
10	2001	1	Lyceum	92	15-12-2005	100
20	2001	2	Lyceum	100	17-12-2004	110
50	2001	1	Lyceum	80	10-02-2006	102
60	2001	1	Technic	90	15-12-2005	100
70	2001	2	Technic	92	15-12-2005	100
110	2002	2	Technic	92	07-07-2007	100
120	2002	1	Lyceum	100	12-04-2008	105
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 5.3: A sample of table of students.

Id-Student	Exam	Date	Grade	Semester1	Semester2	Delay
10	PR	15-01-2002	28	1	1	0
10	BD	22-02-2002	28	1	1	0
10	LP	28-02-2002	26	1	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
20	PR	20-01-2002	30	1	1	0
20	LP	27-02-2002	26	1	2	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 5.4: A sample of table of exams.

numerical codes to the exams, so that we can easily sort the exams and obtain the ideal career as the identity permutation (see Section 4.1). The attributes **Semester1**, **Semester2** and **Delay** represent the new temporal information introduced in the data mining models described in Chapter 4.

Tables 5.5 and 5.6 show the schedule of the Computer Science degree program for Curricula 1 and 2 during the academic years from 2001-2002 to 2003-2004. After inserting all information in these tables we can perform the preprocessing step that, for each exams of each student, adds into the database the attributes about the different temporal information. Before applying tha data

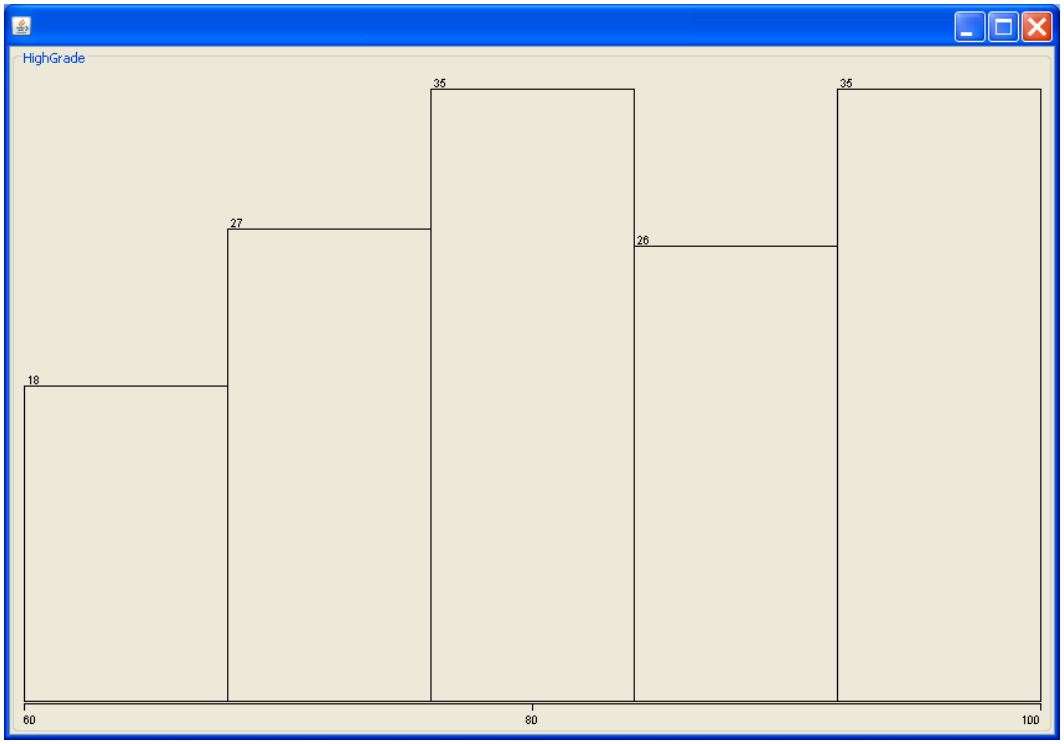


Figure 5.1: Distribution of grades obtained at highschool from graduate students.

mining model to our students data, we can analyze the distributions of some attributes, such as Figures 5.1, 5.2 and 5.3 illustrate. We obtained these distributions by using a function of WEKA, that for each distribution chooses how many intervals to use to display the data distribution. In particular, such as we can see in Figure 5.1, for the attribute HighGrade, that is, the grade obtained at the highschool, the range of grades from 60 to 100 is divided in 6 intervals long 8; the most of students of our case study obtained a good grade, equal or greater than 80. For what concerns the attribute Grade, that is, the final degree grade, such as we can see in Figure 5.2, there are 6 intervals, corresponding to the entire range from the minimum grade 83 to the maximum grade 110 (111 indicates cum laude). Also for this attribute we can observe that the most of stu-

Exam	Description	Semester
1	Programming	1
2	Algebra	1
3	Differential Calculus	1
4	Algorithms and Data Structures	2
5	Computer Architecture	2
6	Mathematical Logic	2
7	Integral Calculus	2
8	Databases and Information Systems	3
9	Physics	3
10	Programming Methodologies	3
11	Concurrent Programming	3
12	Numerical Calculus	3
13	Languages and Compilers	4
14	Computer Networks	4
15	Laboratory of Information Systems	4
16	Laboratory of Operating Systems	4
17	Operating Systems	4
18	Probability and Statistics	4
19	Software Engineering	5
20	Communication Techniques	5
21	Human-Computer Interaction	5
22	Distributed Databases	5
23	Theoretical Computer Science	5
24	IT Work Organization	6
25	Data Structures for Databases	6

Table 5.5: Table of exams for Curriculum 1.

Exam	Description	Semester
1	Programming	1
2	Algebra	1
3	Differential Calculus	1
4	Algorithms and Data Structures	2
5	Computer Architecture	2
6	Mathematical Logic	2
7	Integral Calculus	2
8	Databases and Information Systems	3
9	Physics	3
10	Programming Methodologies	3
11	Concurrent Programming	3
12	Numerical Calculus	3
13	Languages and Compilers	4
14	Computer Networks	4
15	Laboratory of Computer Networks	4
16	Laboratory of Operating Systems	4
17	Operating Systems	4
18	Probability and Statistics	4
19	Software Engineering	5
20	Communication Techniques	5
21	Programming of Networks	5
22	Modeling and Simulation	5
23	Theoretical Computer Science	5
24	Networks Security	6
25	Concurrent and Distributed Systems	6

Table 5.6: Table of exams for Curriculum 2.

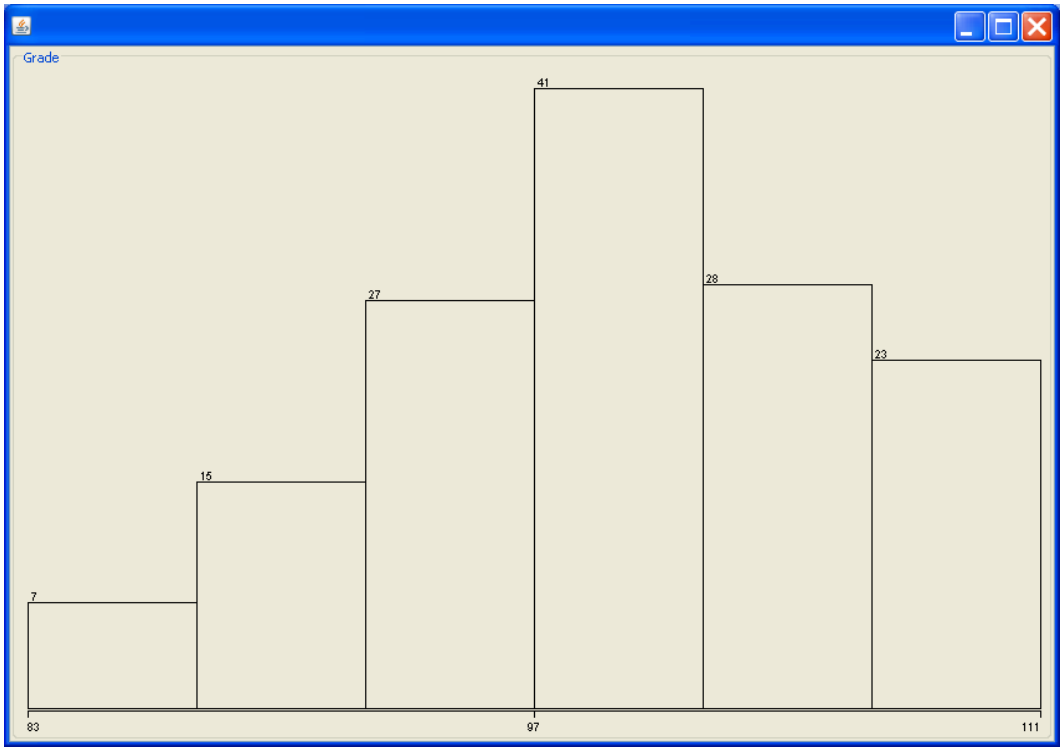


Figure 5.2: Distribution of final grades obtained from students.

dents obtained good grades, greater than 97. Finally, Figure 5.3, concerning the duration time of studies in days, illustrates that there is a large part of students (exactly 41 students) who graduated without delays; however there are some students having a too large degree time and this is an aspect that deserves to be deepened.

5.3 APPLYING THE CLUSTERING MODEL

We analyzed the careers of the students beginning their studies during the academic years from 2001-2002 to 2003-2004 and graduated up to now. We considered students belonging to the two most chosen curricula: *Databases and Information Systems* and *Distributed Systems*. In particular, we analyzed the careers of $N = 85$ students characterized by a sequence of $n = 25$ exams. As illustrated in Section 4.1, the ideal career corresponds to a student which has given every examination just after the end of the corresponding course, without delay. Therefore, for each curriculum we computed the ideal career, after we had identified the semester in which courses were originally hold by the teacher.

In the ideal career, courses relative to the same semester can be sorted by taking into account the preference of students or by giving the same identifier

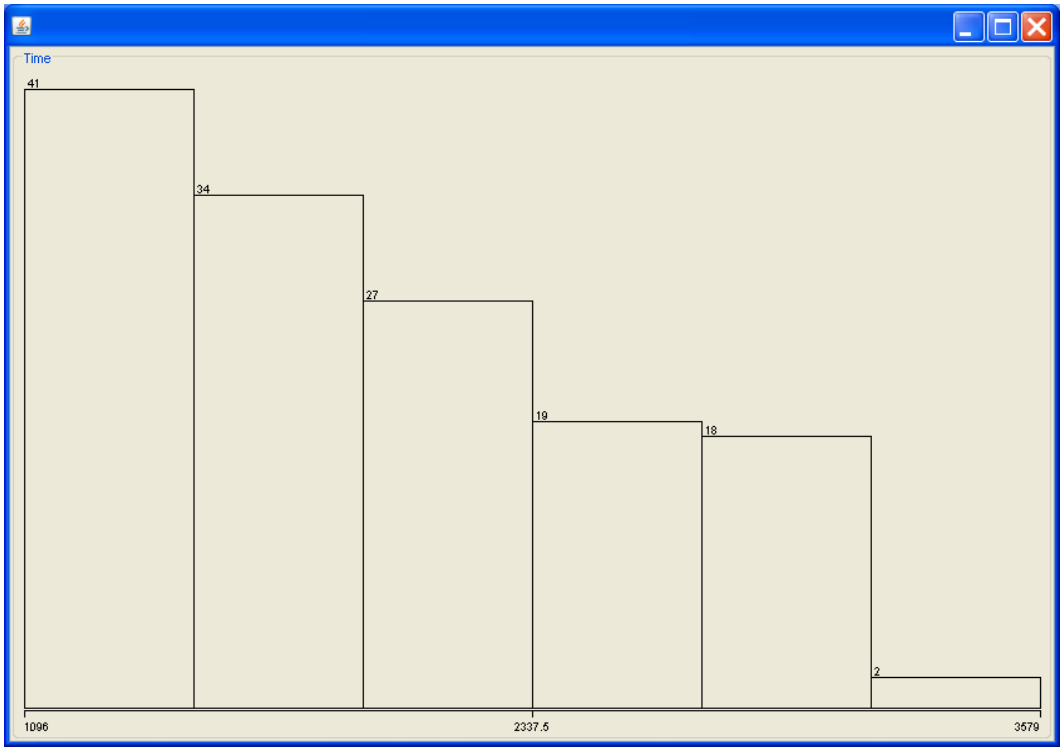


Figure 5.3: Distribution of graduation times of students.

to courses in the same semester, as illustrated in Chapter 4. As we will see later, our tests show that the results obtained are the same by using one or the other ideal career. It could be interesting to apply our model by using a random ideal career; as we refer in the conclusions, we intend to investigate in this direction, to prove that this observation can indicate a weakness of the method (insensitivity to the ideal path) as a strength (stability against small changes).

Tables 5.7 and 5.8 illustrate the two ideal careers, according to Tables 5.5 and 5.6.

$\langle\{1,2,3\}\{4,5,6,7\}\{8,9,10,11,12\}\{13,14,15,16,17,18\}\{19,20,21,22,23\}\{24,25\}\rangle$

Table 5.7: The ideal career for curricula of Tables 5.5 and 5.6.

We note that we obtain the same ideal career for both curricula, because we use the same numerical code. We observe that, for example, for Curriculum 1 the exam code 25, the last in Table 5.7, corresponds to the exam *Data Structures for Databases*, as showed in Table 5.5, while for Curriculum 2, as showed in

$\langle\{1,1,1\}\{2,2,2,2\}\{3,3,3,3,3\}\{4,4,4,4,4,4\}\{5,5,5,5,5\}\{6,6\}\rangle$

Table 5.8: The ideal career when exams of a same semester have the same code.

Table 5.6, the same code corresponds to the exam *Concurrent and Distributed Systems*. We can observe that the two curricula differ only in five exams, which are scheduled in the last year, apart from the exams with code 15, corresponding respectively to *Laboratory of Information Systems* for Curriculum 1 and to *Laboratory of Computer Networks* for Curriculum 2, that are exams of the second semester of the second year. The organization of the first two years of Curriculum 1 and Curriculum 2 is the same, except for one exam.

Then, for each student $\mathcal{J}$ of both curricula we computed the distances $\sigma(\mathcal{J})$ and $\mathcal{A}_{\mathcal{J},\mathcal{I}}$ from the ideal career and inserted these values into two fields **Bubblesort** and **Area** of the database; in order to compute these distances we used the software `Maple` [37].

As already observed, we considered also the alternative representation of the ideal career (see Table 5.8), in which exams taken in the same semester have the same code (as we introduced in 4.1); for each student of both curricula we computed the distance **BubblesortSem**, that is, the number of inversions in the permutation corresponding to the student with respect to the ideal career in Table 5.8. As we will see later, numbers indicating this distance are smaller than those corresponding to the Bubblesort distance, computed when the ideal career is that in Table 5.7. In fact with this alternative representation of the ideal career, it can happen that students taking exams in different order have the same career representation. By referring to the example indicated in Section 4.1, let us consider the sequence $(1,2,3,4,5,6,7)$, the ideal career of 7 exams scheduled in three semester; there are two exams in the first semester, three in the second and two in third. The alternative ideal career corresponds to the sequence $(1,1,2,2,2,3,3)$. Table 5.9 shows the distances from the two ideal careers of two student careers. We can observe that the **Bubblesort** distances are different, while the two careers have the same **BubblesortSem** distance ($= 0$) from the alternative ideal career.

Student career	**Bubblesort**	**BubblesortSem**
$(1,2,5,4,3,7,6)$	4	0
$(2,1,4,5,3,6,7)$	3	0

Table 5.9: A sample of two student careers and their distances from the two representations of the ideal career.

As a curiosity, for what concerns the Bubblesort distance computed for our student careers, we obtained an average value of 95.79 and 80.57 respectively for the two curricula, against the theoretical value of 150; this value is computed over all 25! permutations of numbers 1 to 25, as discussed in Chapter 4, while our real values correspond to a small number of permutations. We note that the careers we analyzed were all different, with different distances, that is, there are not students with the same career. We tried to sort our data according to both fields, **Bubblesort** and **Area**, and as already illustrated in Table 4.2, we found some pairs of careers having values of **Bubblesort** and **Area** in reverse order, as illustrated in Tables 5.10, 5.11 and 5.12; this holds for example for students with **Id-Student** 39 and 45.

Student 39 has an **Area** distance equal to 133.13, smaller than the **Area** distance of student 45, that is equal to 140.19. Instead the **Bubblesort** distance of student 39 is equal to 101, larger than the **Bubblesort** distance of student 45, that is equal to 45. However, although these two distances are not completely equivalent, the difference seems not to be important for the clustering analysis. This will be best seen in the results showed in the next subsection.

To understand how the order of the exams affects the career of the students, we performed several tests by using the `K-means` implementation of `WEKA` (see, e.g., [94]). We first analyzed the careers of students separately for the two curricula. In particular, in both cases, we obtained significant results with $K = 2$ and by selecting as clustering attributes the graduation time, **Time**, and the final grade, **FinalGrade**. The results obtained separately for the two curricula show that the results are not influenced from particular choices of curriculum. With these parameters we can see that students are well divided into two groups: the group of students who graduated relatively quickly and with high grades and the group of students who obtained worst results. Luckily, we observed that students in the first group are characterized by *small* values of **Bubblesort** and **Area**, while students in the second group have *larger* values. Our analysis shows also that the career of a student seems not to be affected by the results achieved at the high school level. We performed similar tests by adding the distance values as attributes of clustering, and we obtained two more distinct clusters, which divide students more precisely in terms of **Time**, **FinalGrade** and **Bubblesort** (or **Area**) distance. We finally analyzed together the students belonging to the two curricula obtaining 2 clusters with the same characteristics as before, thus confirming our previous analysis. This result confirms that, regardless of the curriculum, the more students follow the order given by the ideal career, the more they obtain good performance in terms of graduation time and final grade.

Before the clustering analysis, we conducted an important study about the correlation between all attributes that characterize our students; the correlation matrix illustred in Figure 5.4 shows that there are better correlations between attributes **Time** and **Bubblesort** (**Bubb.**)(0.55) and between attributes **Fi-**

Id-Student	HighSchool	Area	Bubblesort	Bubbl.Sem	Time	FinalGrade
13	scientific	184.42	140	124	3126	99
23	scientific	174.33	128	115	2767	100
25	scientific	170.04	119	108	1540	102
29	technical	78.36	57	45	3052	108
39	other	133.13	101	90	2580	95
45	scientific	140.19	98	86	2111	100
47	scientific	72.02	54	38	1593	105
49	scientific	131.34	100	86	1957	100
55	technical	134.47	106	92	1670	99
56	scientific	89.24	67	52	2034	105
60	scientific	91.04	73	55	2111	96
65	scientific	190.52	142	124	2321	96
66	technical	165.19	119	104	2111	86
67	technical	98.13	77	63	1460	92
71	scientific	192.06	136	120	3003	89
74	other	111.21	80	65	1376	106
76	technical	166.49	106	94	2767	98
79	technical	182.09	128	115	2402	99
80	technical	168.32	121	106	2767	92
82	scientific	200.29	135	117	2402	97
83	technical	124.21	96	79	2402	105
84	scientific	159.31	117	102	3579	97
93	scientific	78.06	56	45	1824	97
96	scientific	115.05	91	77	2034	98
97	technical	78.26	60	46	2111	99
103	technical	110.04	89	80	3496	93
105	technical	73.02	59	44	3126	95
110	technical	108.17	76	64	1904	100
112	scientific	122.29	102	93	3052	89
115	other	175.26	131	117	2188	93
120	technical	118.14	87	70	1540	100
140	technical	125.07	88	74	1460	99
144	technical	103.36	72	62	1540	101
145	technical	84.15	68	60	1540	105

Table 5.10: Table of students enrolled during 2001-2002 and 2003-2004 of curricula 1 and 2 - first part.

Id-Student	HighSchool	Area	Bubblesort	Bubbl.Sem	Time	FinalGrade
146	scientific	79.01	62	52	2038	104
147	technical	116.25	81	72	2324	99
148	technical	150.16	114	102	2403	93
149	technical	112.26	86	77	2038	93
150	scientific	75.52	54	44	1540	99
151	technical	70.03	50	41	1540	103
153	scientific	67.05	50	40	1376	97
154	scientific	70.19	58	48	1670	106
156	technical	107.16	84	74	2324	93
160	technical	88.38	62	52	2038	106
161	scientific	139.18	110	99	2114	103
162	scientific	89.23	77	67	2038	105
163	technical	150.35	114	104	3132	91
164	technical	105.03	80	65	3132	100
172	other	113	88	76	1957	98
173	technical	159.11	121	115	2688	95
175	technical	58.48	43	37	1593	98
176	technical	116.37	76	69	3000	89
178	technical	94.04	63	57	1593	99
179	technical	169.34	122	117	2583	91
180	scientific	115.57	86	75	3067	100
183	technical	155.16	117	110	2688	94
189	other	165.38	125	108	2583	94
191	scientific	180.01	123	116	2403	103
192	other	113.34	74	68	1460	102
197	scientific	80.29	59	47	1540	106
202	scientific	37.02	29	18	1306	111
203	technical	101.23	76	63	1670	91
217	other	136.54	96	82	2762	99
219	scientific	112.13	85	70	1593	99
225	technical	162.35	121	105	2121	85
226	technical	97.24	80	65	2768	108
233	scientific	110.44	77	66	1960	97
234	technical	67.48	51	33	1542	111

Table 5.11: Table of students enrolled during 2001-2002 and 2003-2004 of curricula 1 and 2 - second part.

Id-Student	HighSchool	Area	Bubblesort	Bubbl.Sem	Time	FinalGrade
236	scientific	94.03	62	48	1460	98
246	scientific	190.28	146	139	2767	90
250	scientific	133.42	97	86	2219	103
252	scientific	56.04	40	26	1960	100
254	scientific	94.03	64	48	1383	105
256	scientific	161.03	122	108	2039	96
257	scientific	162.42	120	108	2039	98
264	other	103.03	77	71	1960	102
270	scientific	79.11	59	44	1306	107
292	scientific	155.22	107	96	2398	91
296	technical	155.19	103	93	1750	98
311	technical	220.55	175	169	2398	98
312	technical	51.05	35	23	1096	111
313	technical	86.11	64	52	2639	106
314	scientific	219.14	155	147	2937	89
317	other	120.13	86	78	1852	101
321	scientific	49.04	40	28	1593	111

Table 5.12: Table of students enrolled during 2001-2002 and 2003-2004 of curricula 1 and 2 - third part.

nalGrade (**FinalG.**) and **Bubblesort** (-0.58) than between attributes **FinalGrade** and **Time** (-0.44). We obtain similar results with the **Area** and the **Bubblesort-Sem** (**Bubb.Sem**) distances; also these attributes are well correlated with the final grade and with the time. We can also note that the **HighGrade** (**HighG.**), that is, the vote that students obtained at the high school, is not well correlated with the final results of university studies.

	Area	**Bubb.**	**Bubb.Sem**	**AreaSem**	**Time**	**FinalG.**	**HighG.**
Area	1	0	0	0	0	0	0
Bubb.	1	1	0	0	0	0	0
Bubb.Sem	1	1	1	0	0	0	0
AreaSem	0.90	0.96	0.98	1	0	0	0
Time	0.48	0.55	0.52	0.48	1	0	0
FinalG.	-0.53	-0.58	-0.58	-0.59	-0.44	1	0
HighG.	-0.27	-0.30	-0.31	-0.31	-0.16	-0.35	1

Figure 5.4: Matrix of correlations amomg attributes of student dataset.

Results on students of Curriculum 1

We performed several tests by using $K = 2$ and the clustering attributes **Bubblesort** distance, **FinalGrade** and **Time**. Figures 5.7, 5.8, 5.9 correspond to tests with clustering attributes **Bubblesort** distance, **FinalGrade** and **Time**; the attribute **Time** is expressed in days while the **FinalGrade** is an integer between 66 and 110 (111 denotes 110 cum laude). Figure 5.7 shows students divided into two well separated clusters; the *good* students correspond to red stars, having high **FinalGrade** and small **Time**. The *not so good* students correspond to blue stars, having low **FinalGrade** and large **Time** distances. We observe that we have well separated clusters also by performing clustering without the attribute **Bubblesort** distance, but the introduction of the attribute distance (**Bubblesort** or **Area**) determines clusters better separated than clusters obtained without the distance as clustering attribute. Figures 5.8 and 5.9 emphasize the correlation between **FinalGrade** and **Bubblesort** distance, and between **Time** and **Bubblesort** distance, respectively; we can conclude that *students having high final grade and graduated in a relative small time have a small distance from the ideal career*. We also performed test by using the **Area** distance. The results are similar and are illustrated in Appendix A, Figures 7.1, 7.2 and 7.3.

We deepened the clustering analysis by investigating the *correlation index* to measure the goodness of obtained clusters; as illustrated in [92], the correlation index computed by using the *proximity* and *similarity*, or *incidence*, matrices (see, e.g. [41, 69]), is a way to measure the goodness of obtained clusters. In

particular, we computed the proximity matrix $P = (P_{ij})$ having one row and one column for each element of the dataset; each element P_{ij} represents the Euclidean distance between elements i and j in the dataset. In the incidence matrix $I = (I_{ij})$ each element I_{ij} is 1 or 0 if the elements i and j belong to the same cluster or not. Finally, we computed the *Pearson's correlation*, as defined in [92], between the linear representation by rows of matrices P and I. For the clustering results illustrated in Figures 5.7, 5.8 and 5.9, we obtained a Pearson's correlation of -0.38; we also computed the *cosine similarity* between the linear representation by rows of matrices P and I , by obtaining a value of 0.42.

We present also the results of the test performed by using $k = 2$ and the clustering attributes **FinalGrade**, **Time** and **BubblesortSem** distance, that is, the distance from the ideal career illustrated in Table 5.8; in Appendix A, the Figures 7.6, 7.7 and 7.8 show clusters that are similar to those obtained with the **Bubblesort** and **Area** distances; this confirms that the choice of the ideal career showed in Table 5.7 is not restrictive for our case study. Figures 5.5 and 5.6 show the details of the results, with the coordinates of the two centroids, obtained by considering the **Bubblesort** and the **BubblesortSem** distance respectively. We can observe that the results are very similar for what concerns the centroids; they obviously have different values for the distances, but very similar values of **Time** and **FinalGrade**.

Results on students of Curriculum 2

Also for this subset of students we performed similar tests as those described in the previous subsection, that is, by considering $K = 2$ and the clustering attributes **Bubblesort** distance, **FinalGrade** and **Time**. We obtained similar results, which we can see in Figures 5.10, 5.11 and 5.12. We again evaluated the clusters, by obtaining a Pearson's correlation of -0.54 and a cosine similarity of 0.39. Also for these data we obtain good results by considering the **Area** distance instead of **Bubblesort** distance. We performed also for these students tests by considering the **BubblesortSem** distance, obtaining the results of Figures 7.9, 7.10 and 7.11, showed in Appendix A. We again observe that the choice on the ideal career is indifferent, that is, the results of clustering analysis did not change.

Results on students of Curricula 1 and 2

Also for the entire dataset, referring to alla students enrolled from 2001 to 2003 years, we performed the analysis with $K = 2$ and with the clustering attributes **Bubblesort** distance, **FinalGrade** and **Time**. Figures 5.13, 5.14 and 5.15 illustrate the results on students of Curricula 1 and 2, by using the **Bubblesort** distance. We point out that we obtained similar results also by using the **Area** and the **BubblesortSem** distances; in particular, by comparing the results of Figures 5.14 and 5.15 with Figures 7.4 and 7.5, in Appendix A, we can observe

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.
              EuclideanDistance -R first-last" -I 500 -S 35678
Relation:     Student2001-2003Cur1-weka.filters.unsupervised.
              attribute.Remove-R1-2,5-6,8-10
Instances:    38
Attributes:   3
              Bubblesort
              Time
              FinalGrade
Test mode:    evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 3.0802570187792853
Missing values globally replaced with mean/mode

Cluster centroids:
                        Cluster#
Attribute    Full Data         0         1
                  (38)      (18)      (20)
=========================================
Bubblesort     98.0789   76.0556     117.9
Time         2251.8684 1801.7778   2656.95
FinalGrade     98.7368  103.0556     94.85


Clustered Instances

0      18 ( 47%)
1      20 ( 53%)
```

Figure 5.5: Log of the results obtained by WEKA *K*-means on the students of Curriculum 1, by using **Bubblesort** distance.

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.
              EuclideanDistance -R first-last" -I 500 -S 10
Relation:     Student2001-2003Cur1-weka.filters.unsupervised.
              attribute.Remove-R1-3,5-6,8,10
Instances:    38
Attributes:   3
              Time
              FinalGrade
              BubblesortSem
Test mode:    evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 3.046748970467072
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute       Full Data         0         1
                     (38)      (20)      (18)
=============================================
Time            2251.8684   2656.95 1801.7778
FinalGrade        98.7368     94.85  103.0556
BubblesortSem     86.5263       106   64.8889


Clustered Instances

0      20 ( 53%)
1      18 ( 47%)
```

Figure 5.6: Log of the results obtained by WEKA *K*-means on the students of Curriculum 1, by using **BubblesortSem** distance.

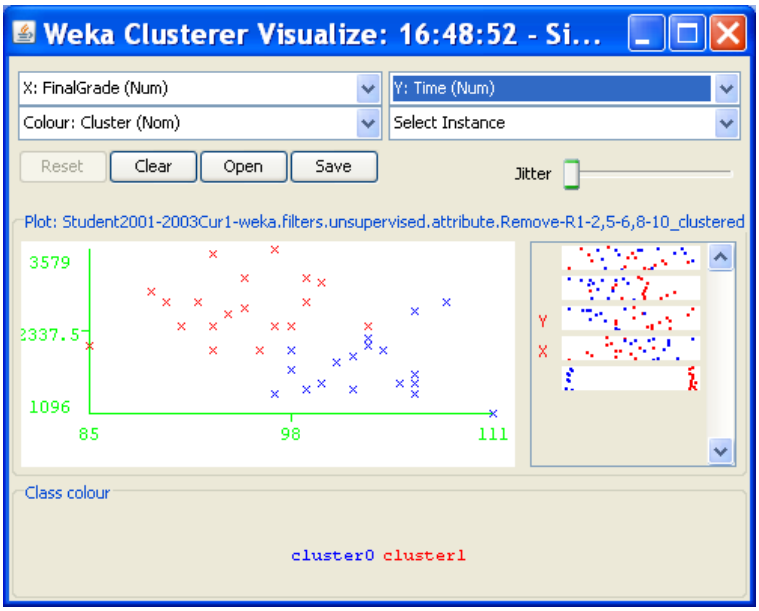


Figure 5.7: Students of Curriculum 1 with respect to `FinalGrade` and `Time`.

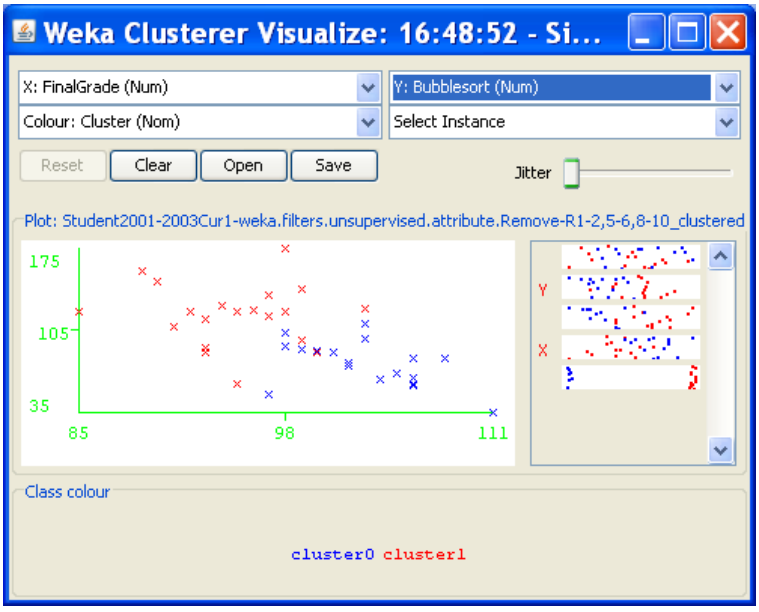


Figure 5.8: Students of Curriculum 1 with respect to `FinalGrade` and `Bubblesort` distance.

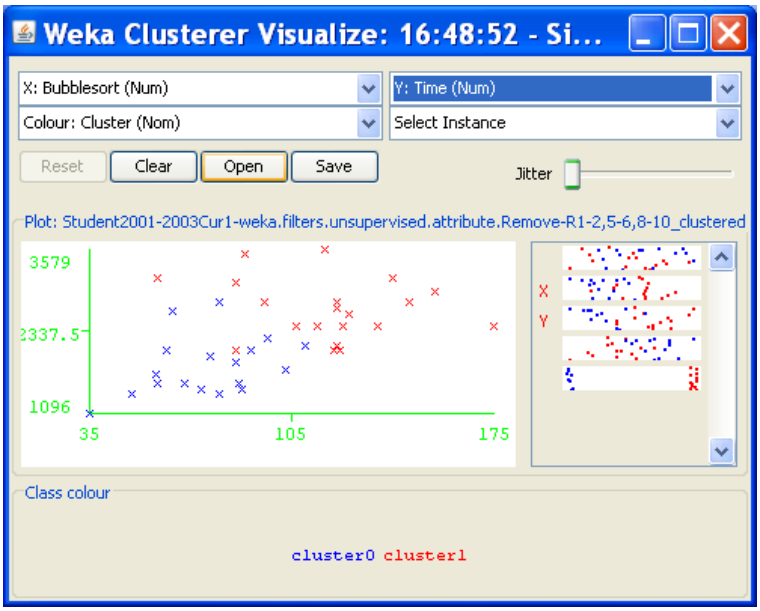


Figure 5.9: Students of Curriculum 1 with respect to `Bubblesort` distance and `Time`.

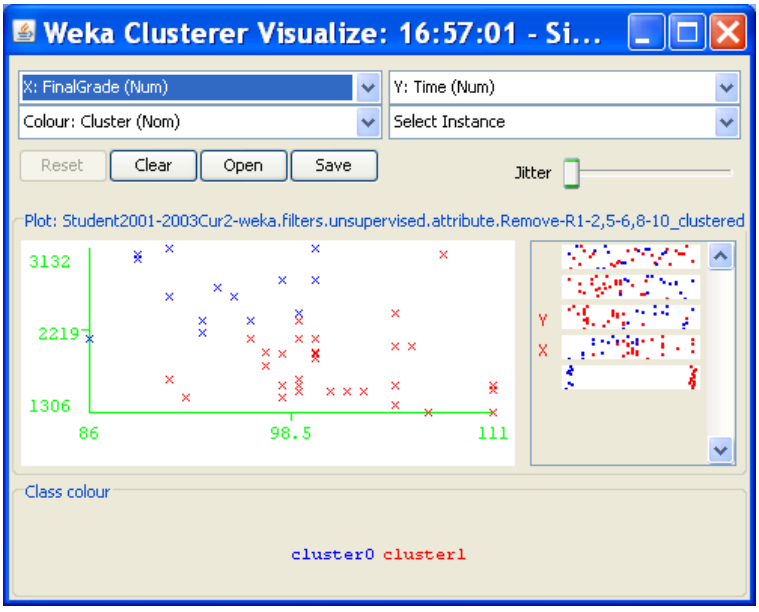


Figure 5.10: Students of Curriculum 2 with respect to `FinalGrade` and `Time`.

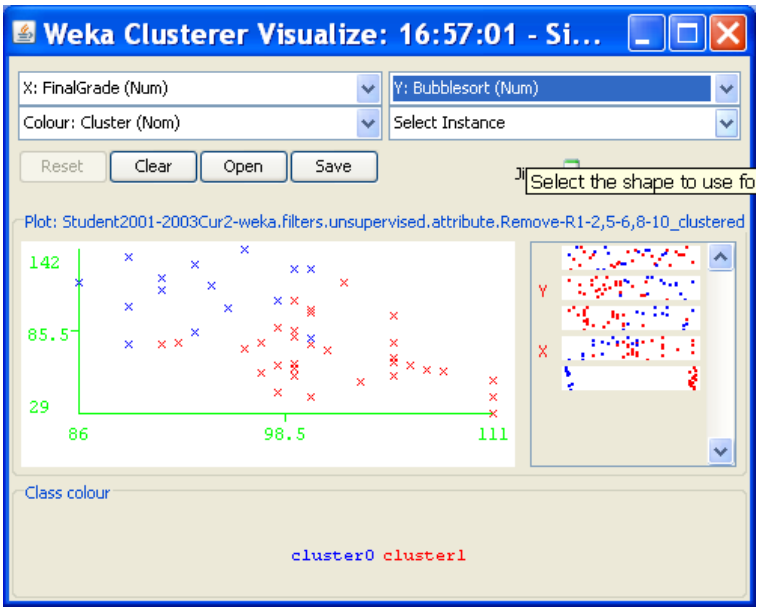


Figure 5.11: Students of Curriculum 2 with respect to `FinalGrade` and `Bubblesort` distance.

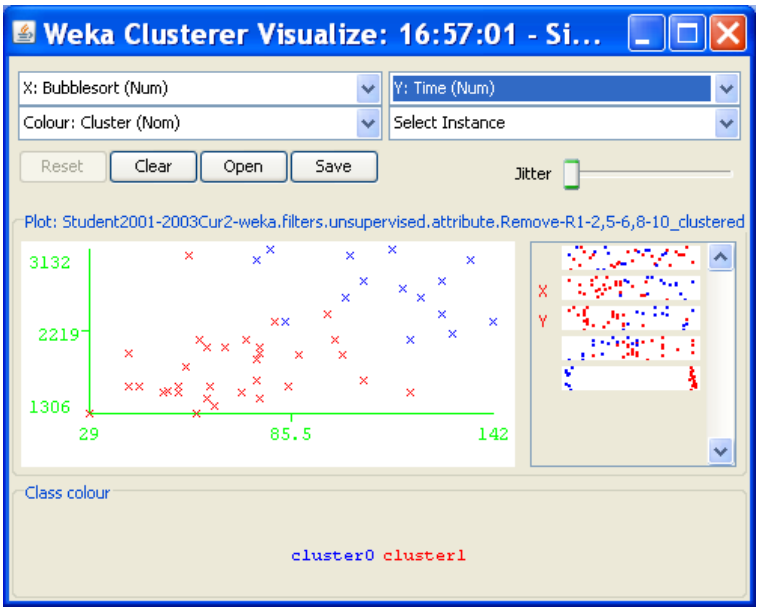


Figure 5.12: Students of Curriculum 2 with respect to `Bubblesort` distance and `Time`.

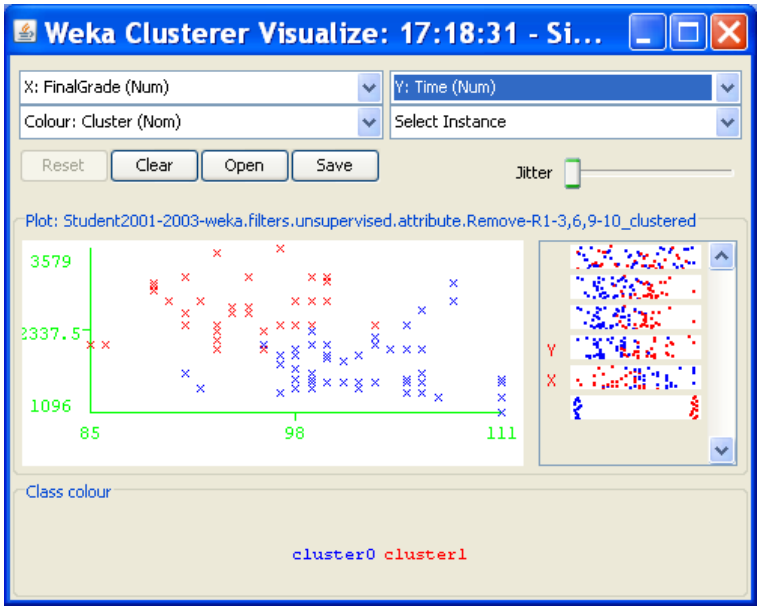


Figure 5.13: Students with respect to `FinalGrade` and `Time`.

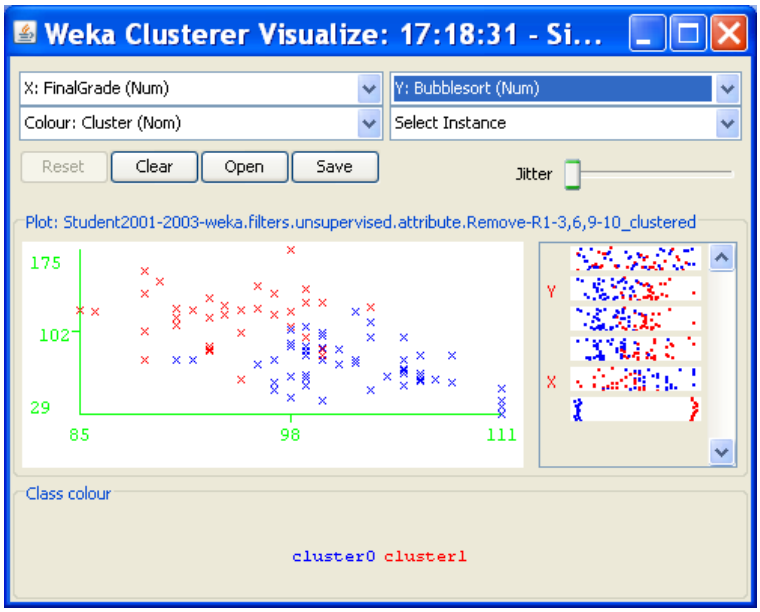


Figure 5.14: Students with respect to `FinalGrade` and `Bubblesort` distance.

that with the **Area** distances we obtained very similar clusters than with the **Bubblesort** distance. Also in this case we can conclude that the information about the distance of a student career from the ideal career allows us to split students into two well defined groups.

The clusters obtained by considering all students have a Pearson's correlation of -0.47 and a cosine similarity of 0.39.

5.3.1 *Extending clustering by applying classification model*

We extend the results obtained with clustering model by running the `WEKA J48` algorithm (an implementation of `C4.5` algorithm, introduced in Section 3.4.1). We performed several tests applying the algorithm to different sets of students and by choosing different class attributes. For example, by running `J48` on students of Curriculum 1 and Curriculum 2 enrolled in the academic years from 2001-2002 to 2003-2004, we obtained a good classification tree, showed in Figure 5.16, by using the **FinalGrade** *class*, labeled with **Grade**, for simplicity. We can observe that the results obtained with clustering showed in Figures 5.7, 5.8 and 5.9 are confirmed; students having a small distance from the ideal career obtain good results, that is, a high **FinalGrade** and a small **Time**. We observe that the tree can be considered a forecasting model, because a student may have, for example, a high grade if he/she will maintain a small distance from the ideal career. In this model the type of high school (node **HighSchool**) is considered; Figure 5.16 shows also that students from the scientific school will have a good grade if they maintain very close to the ideal career. This result was obtained by using both the tree cross-validation that the training set. We could perform a more accurate analysis by considering information about dropouts and failed students; however we do not have this type of data and we can only consider the behaviour just illustrated. Results obtained by using the **Bubblesort** *class* and the **Time** *class* are showed in Figures 5.17 and 5.18 respectively. In particular, Figure 5.18 is a sample of overfitting, that is, with our limited data, the model is too close to the data; Figure 5.17 ia clear even if a bit trivial: students which will obtain a large or medium final grade could have a small distance from the ideal career, while students with a small grade could be far from the ideal career.

We precise that for a good predictive analysis, we also need to consider the dropouts, but, as we just observed, we do not have this information; we will consider this aspect to perform additional analysis, such as we will present in Chapter 6.

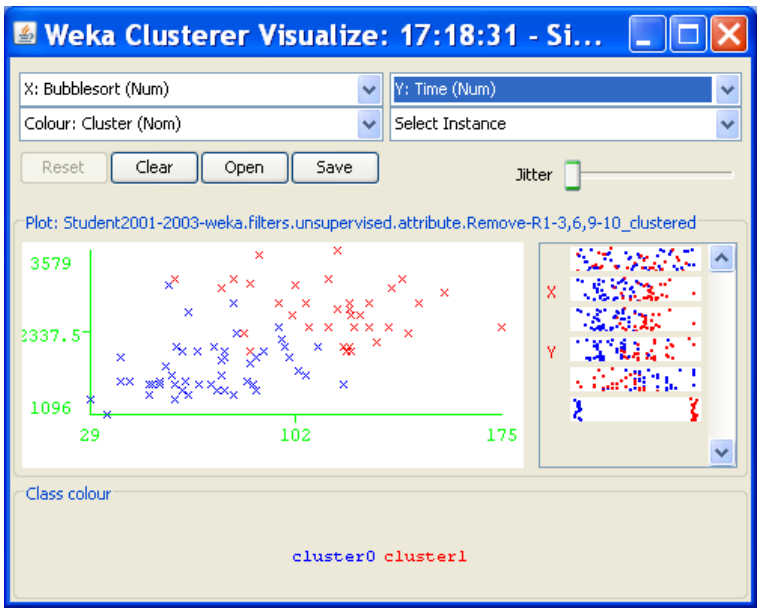


Figure 5.15: Students with respect to `Bubblesort` distance and `Time`.

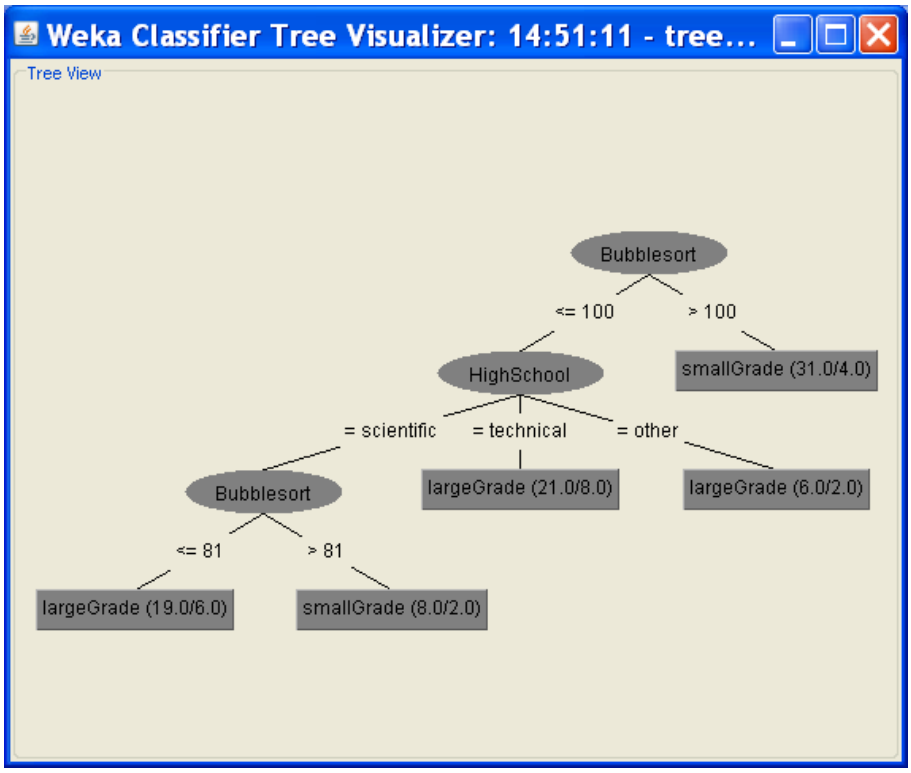


Figure 5.16: Decision tree with class = **FinalGrade**.

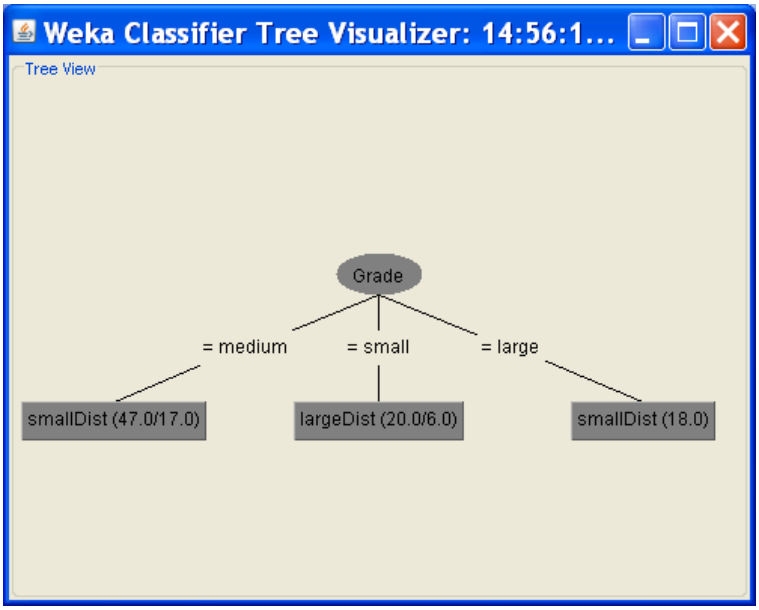


Figure 5.17: Decision tree with class = **Bubblesort**.

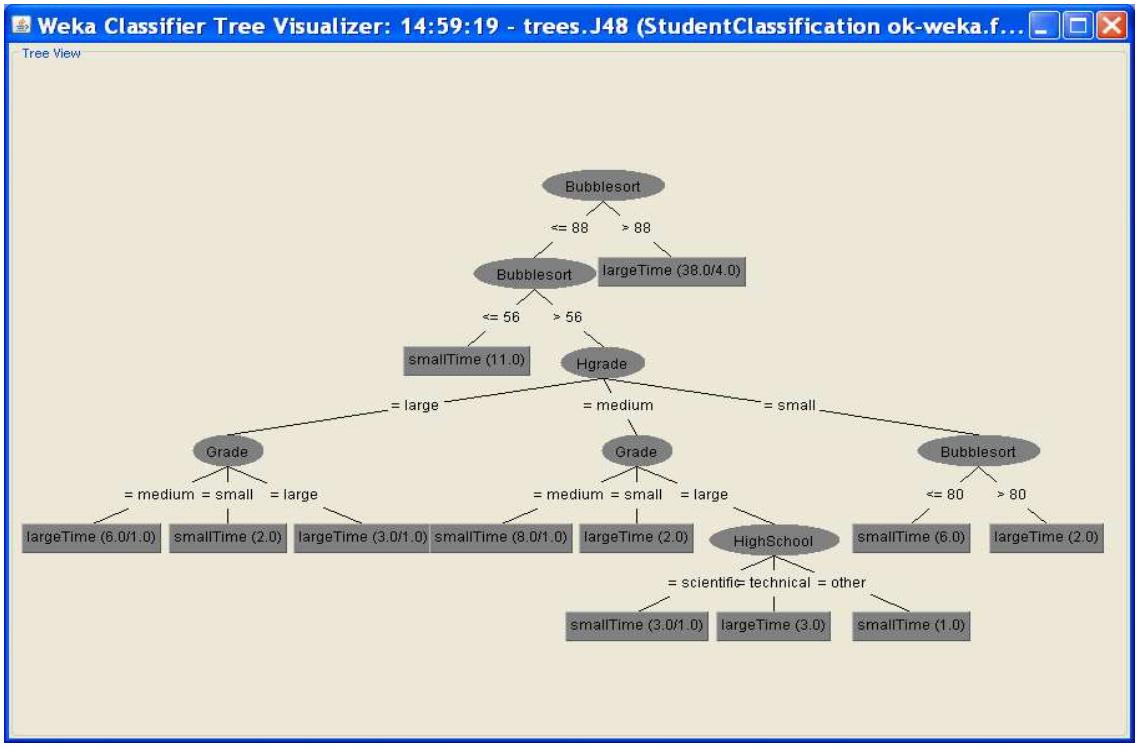


Figure 5.18: Decision tree with class = **Time**.

5.4 APPLYING SEQUENTIAL PATTERN MODEL

We considered the graduated students which began their career during the years 2001-2007. For each student, the data sets contain, among other fields, the identifier of the student, **Id-Student**, the year of enrollment, **Enrollment**, the date, **Date** and the mark, **FinalGrade**, of final examination, and, for each exam, the identifier of the student, **Id-Student**, the identifier of the exam, **Exam**, the date, **Date**, and the corresponding mark, **Grade**. For each student there are many items, one for each examination taken by the student, such as Tables 5.3 and 5.4 illustrate. As we already said, for each student, we added the attribute **Time** corresponding to the length of studies. In Tables 5.13 and 5.14 a portion of real data are illustrated.

By adding to Table 5.14 the temporal information, as we did in Section 5.2, we obtain Table 5.15, from which we can easily obtain the file to use as input for the algorithm SPAM, by selecting, for each student and for each exam, the fields **Id-Student**, **Exam** and **Semester2** (or **Delay**). The file obtained in this way is just in the format accepted by SPAM and illustrated in Table 4.4 of the previous chapter. As already observed, during the years 2001-2007 the organization of the academic degree suffered several changes and provided different curricula. For this reason, it is not possible to analyze together all the exams of all the students in the database. So we decided to analyze the following two groups of students characterized by the same subset of exams:

1) a subset of the data containing 950 records corresponding to the 25 exams taken by 38 students of the Curriculum 1 during the years 2001-2004 (see Table 5.5);

2) a subset of the data containing 2256 records corresponding to the 16 exams taken by all the 141 students. Table 5.16 shows these exams. We observe that the order of exams is different from that shown in Tables 5.5 and 5.6; in this case we are dealing with exams taken by all students enrolled during the whole period under analysis. As already observed, in this period there were some changes about the academic degree schedule. The order of exams presented in Table 5.16 reflects these considerations.

These data sets are not very large but allowed us to illustrate the methodology on a real case study.

Table 5.17 shows some frequent patterns obtained by running SPAM on data set 1); this output has been obtained by a useful postprocessing phase, because, as we presented in Section 4.2.1, the output of SPAM is not very simple to read. In Table 5.17 the number at the right of each row indicates the support of the pattern, the braces enclose the items (or events or exams) and brackets enclose the frequent patterns. In our analysis it has been useful to sort the output according to the number of items involved in the patterns, as we have already observed in Section 4.2.2.

Id-Student	Enrollment	Date	FinalGrade	Time
75	2001	2005-07-08	106	1540
148	2002	2008-04-29	104	2937
156	2002	2007-04-27	106	2111
314	2003	2006-09-06	110 cum laude	1593
⋮	⋮	⋮	⋮	⋮

Table 5.13: Data for students

Id-Student	Exam	Date	Grade
75	18	2002-01-14	27
75	21	2002-02-20	28
75	15	2002-09-04	26
75	19	2004-01-29	25
⋮	⋮	⋮	⋮
148	2	2003-01-31	22
148	21	2003-02-25	25
148	1	2003-06-10	24
148	17	2004-06-04	27
⋮	⋮	⋮	⋮

Table 5.14: Data for exams

Student	Exam	Date	Semester1	Semester2	Delay
75	18	2002-01-14	1	1	0
75	21	2002-02-20	1	1	0
75	15	2002-09-04	2	2	0
75	19	2004-01-29	3	5	2
⋮	⋮	⋮	⋮	⋮	⋮
148	2	2003-01-31	1	1	0
148	21	2003-02-25	1	1	0
148	1	2003-06-10	2	2	0
148	17	2004-06-04	2	4	2
⋮	⋮	⋮	⋮	⋮	⋮

Table 5.15: Data for finding patterns

Exam	Description
1	Programming
2	Computer Architecture
3	Differential Calculus
4	Integral Calculus
5	Algorithms and Data Structures
6	Languages and Compilers
7	Physics
8	Computer Networks
9	Concurrent Programming
10	Operating Systems
11	Probability and Statistics
12	Programming Methodologies
13	Numerical Calculus
14	Laboratory of Operating Systems
15	Communication Techniques
16	Theoretical Computer Science

Table 5.16: Common exams of 141 students.

By starting from the input format used by SPAM, we performed an important preprocessing phase to obtain the appropriate input format for the algorithm CloSpan, such as described in [27]. This algorithm also produces an output that needs a postprocessing step.

For each data set we run both algorithms to find frequent patterns; in the next sections we present tests and results and compare the outputs obtained with SPAM and CloSpan; as we could expect, CloSpan is always more efficient than SPAM about the number of patterns produced.

5.4.1 *Tests and results on students enrolled in 2001-2004 of Curriculum 1*

We ran SPAM on our data set of students by using several values of support. In Table 5.17 we can see a small subset of the 1271 patterns produced by SPAM on the students data set 1), by using the semester as temporal information and minsup equal to 0.5. As we already observed, we processed this result to obtain a more readable format in which the frequent sequences are sorted according to the number of events; the maximum value of events in a pattern was 5 and, in particular, we obtained twelve 5-sequences of length 4, one 5-sequence of length 5, almost two hundred 4-sequences and many other sequences involving a smaller number of exams. In this case the ideal career corresponds to the identity permutation of length 25 and the career of a generic student is a permutation of the integers 1 to 25. In particular, when the temporal information is the semester, we have

$$s_I^{[s]} = \langle\{1,2,3\}\{4,5,6,7\}\{8,9,10,11,12\}\{13,14,15,16,17,18\}\{19,20,21,22,23\}\{24,25\}\rangle\,.$$

By comparing this sequence with Table 5.17 we can find some interesting results. For example, we can observe that 70% of students take exams according to the pattern in the third line. This pattern corresponds to the ideal career except for exam with code 12 which should be taken before the pair 16, 17. Exam code 12 corresponds to *Numerical Calculus*, while the pair 16, 17 to *Operating systems* and the corresponding *Laboratory*. A more detailed analysis on the data base shows also that the average rating for exam 12 is lower than the corresponding rates for 16, 17 and that the average delay of exam 12 is much larger than the delay of the pair 16, 17. Therefore we can conclude that exam 12 is considered very difficult by students.

A similar analysis has been done for other frequent sequences. For example, line 6 in Table 5.17 show that the exams 19, *Software Engineering*, and 12, *Numerical Calculus*, are made in reverse order than expected from the ideal career by a large part of the students. Again, the average grade of exam 12 is smaller than that of exam 19.

By running SPAM with minsup equal to 0.5 we also obtained frequent patterns with minsup equal to 1; some of them are obvious, for example each exam was

Frequent sequences
$\langle\{4\}\{8\}\{16,17\}\{12\}\rangle$ 0.5
$\langle\{2\}\{8\}\{16,17\}\{25\}\rangle$ 0.5
$\langle\{1\}\{10\}\{16,17\}\{12\}\rangle$ 0.7
$\langle\{2\}\{8\}\{3\}\{7\}\{12\}\rangle$ 0.5
⋮
$\langle\{2\}\{8\}\{21\}\{12\}\rangle$ 0.5
$\langle\{2\}\{8\}\{19\}\{12\}\rangle$ 0.6
$\langle\{2\}\{8\}\{17\}\{25\}\rangle$ 0.6
$\langle\{1\}\{13\}\{19\}\{18\}\rangle$ 0.6
$\langle\{4\}\{8\}\{17\}\{12\}\rangle$ 0.5
⋮
$\langle\{8\}\{23\}\rangle$ 1
$\langle\{13\}\{23\}\rangle$ 1
$\langle\{1\}\{23\}\rangle$ 1
⋮
$\langle\{3\}\rangle$ 1
$\langle\{2\}\rangle$ 1
$\langle\{1\}\rangle$ 1

Table 5.17: Some frequent sequences for the data set of students 1) obtained with `SPAM`: `minsup`$= 0.5$

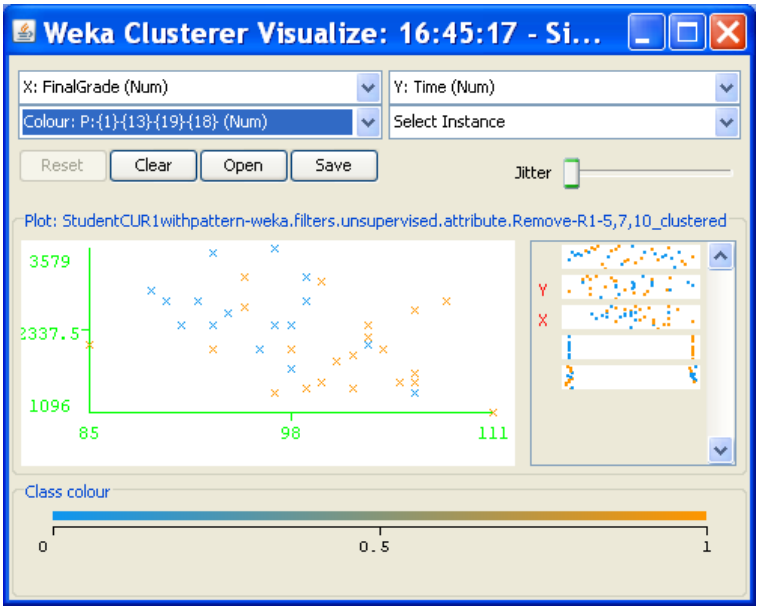


Figure 5.19: Students of data set 1) with respect to `FinalGrade` and `Time`: pattern P_1 in evidence.

taken by 100% of the students, while others are non trivial, as for example $\langle\{8\}\{23\}\rangle$, $\langle\{13\}\{23\}\rangle$ and $\langle\{1\}\{23\}\rangle$. These patterns show that all students gave exam 23, *Theoretical Computer Science*, after exams 8, 13 and 1, corresponding to *Databases and Information Systems*, *Languages and Compilers* and *Programming*. At a first sight, these patterns might seem not interesting, because the exam 23 corresponds to a course of semester 5 (first semester of the third year), while exams 8, 13 and 1 correspond to courses given in previous semesters (3, 4 and 1, respectively). However, `SPAM` gives this kind of result only for exam 23, while there are other exams corresponding to courses given in semesters 5 and 6, for example exams 19, *Software Engineering*, and exam 25, *Data structures for databases*, which are not so frequently taken after other particular exams. Moreover, we explicitly observe that in the years under consideration, prerequisites between exams were not fixed. Again, the analysis highlights a difficulty of students to take the exam 23.

We performed identical tests (with the same values of support) with `CloSpan`, obtaining the same results. The algorithm `CloSpan` produces, for this data set, an output of 1060 patterns; also in this case the maximum value of events in a pattern was 5, there were twelve 5-sequences of length 4 but only 140 4-sequences, while with SPAM they were 200. After the postprocessing phase, we obtain the following sample output:

$$\langle(1)\,(10)\,(16\ 17)\,(12)\rangle \quad 24 \quad 4 \quad 5,$$

where 24 indicates the support, 4 is the length of the pattern, that is, the number of its itemsets, and 5 is the number of the items involved in the patterns; we compared this pattern with the third pattern of Table 5.17, that is, $\langle\{1\}\{10\}\{16,17\}\{12\}\rangle$ 0.7. We observe that they correspond to the same pattern with the same support. In fact, in the output of `CloSpan` the support is expressed in frequencies (24 out of 38 students), in the output of `SPAM` the support is expressed in percentage. In `CloSpan` output the three numbers at the end of each row are: the frequency of students verifying the pattern, the number of elements of the patterns (itemsets of pattern) and the number of events (exams) involved in the pattern.

Table 5.19 shows the comparison between the output of `SPAM` and `CloSpan`.

We also run `SPAM` on the students data set 1) by using the delay as temporal information and `minsup` equal to 0.3, thus obtaining 3853 patterns with maximum value of events equal to 6. In this case, we have $s_I^{[d]} = \langle\{1,\ldots,25\}\rangle$ and we found the interesting 5-sequence $\langle\{8,13,14,15,24\}\rangle$ of length 1 verified by 30% of students. Exam 14 corresponds to *Computer Networks*, exam 15 to *Laboratory of Information Systems* and, finally, exam 24 to *IT Work Organization*. We explicitly observe that all the exams in the pattern are Computer Science exams. An analysis of the database shows that the students verifying the pattern take the corresponding exams with delay equal to zero. We can conclude that students take without difficulties the exams in the pattern.

Clustering on frequent patterns

As illustrated in Section 4.2.2, we updated the database with the Boolean information about the frequent patterns which appeared to be most interesting and performed many tests by using the `K-means` implementation of `WEKA` (see, e.g., [94]). Among the various patterns, for data set 1), we considered the pattern $P_1 = \langle\{1\}\{13\}\{19\}\{18\}\rangle$ with support 60% (see Table 5.17) and the pattern $P_2 = \langle\{8,13,14,15,24\}\rangle$, examined above. The cluster analysis on the attributes P_1 and P_2, together with the final grade and graduation time, with $K = 2$, shows that students who satisfy both patterns achieve better results than those who do not satisfy them. Figure 5.7 illustrates the two distinct clusters: the red one corresponds to the group of students who graduated relatively quickly and with high grades; the other cluster corresponds to students who obtained worse results. Figure 5.19 highlights students who verify pattern P_1. A possible interpretation of this result is that pattern P_1 involves an almost ordered subsequence of the ideal career, with only an inversion between 18 and 19, and P_2 identifies students without delay in the corresponding exams.

5.4.2 *Tests and results on all students enrolled in 2001-2007*

For what concerns the larger data set 2), having $s_I^{[d]} = \langle\{1,\ldots,16\}\rangle$, the sequential pattern and clustering analysis gave similar results. In particular, we point out the pattern $P_3 = \langle\{2,6,7,8,11\}\rangle$ verified by 45 students and obtained by using the delay as temporal information. In this case, by using `minsup` equal to 0.3, `SPAM` produced 708 patterns with maximum value of events equal to 6.

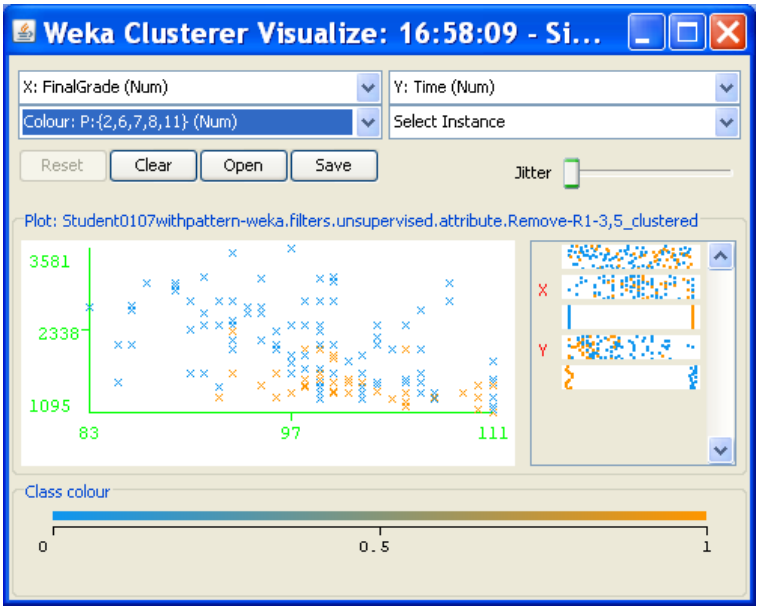


Figure 5.20: Students of data set 2) with respect to `FinalGrade` and `Time`: pattern P_3 in evidence.

The cluster analysis on the attribute P_3, together with the final grade and graduation time, with $K = 2$, shows that students satisfying the pattern (red and blue in Figure 5.13) achieve good results. Figure 5.20 highlights students verifying pattern P_3. Tests on this data set gave good results also with $K = 3$, such as Figures 5.21 and 5.22 show. There is a medium group of students having medium results, that is, medium **FinalGrade** and medium **Time**. In particular, Figure 5.22 shows that students having worst results (corresponding to green stars in Figure 5.21) do not verify the pattern P_3. Table 5.18 shows the log file obtained by `WEKA`, with the coordinates of the three centroids; the medium students correspond to cluster 0, and to blue stars in Figure 5.21.

We performed identical tests (with the same dataset and the same values of support) with `CloSpan`, obtaining the same results obtained with `SPAM`. We refer to Table 5.19 that points out that `CloSpan` is more efficient than `SPAM` for what concerns the number of resulting patterns, because, as we illustrated in Section 3.6.2, `CloSpan` produces only closed patterns (i.e. patterns corresponding to sets

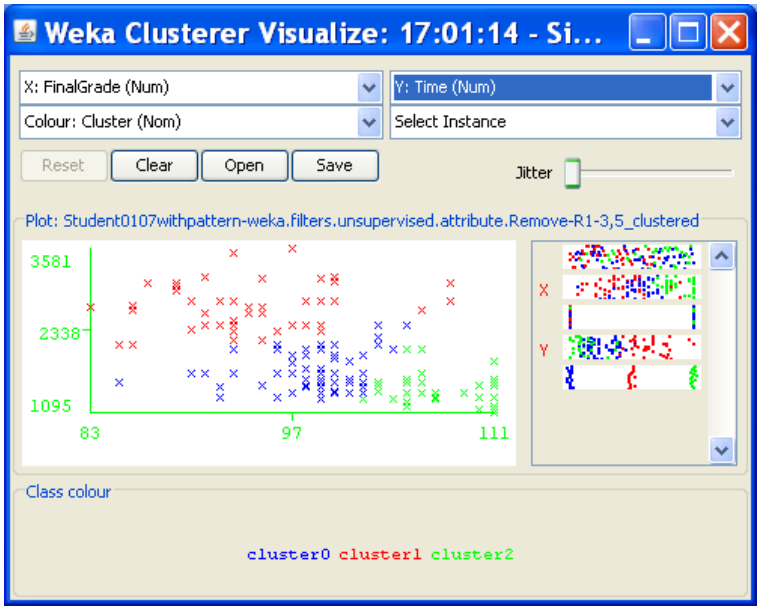


Figure 5.21: Students of data set 2) with respect to `FinalGrade` and `Time` with $K = 3$.

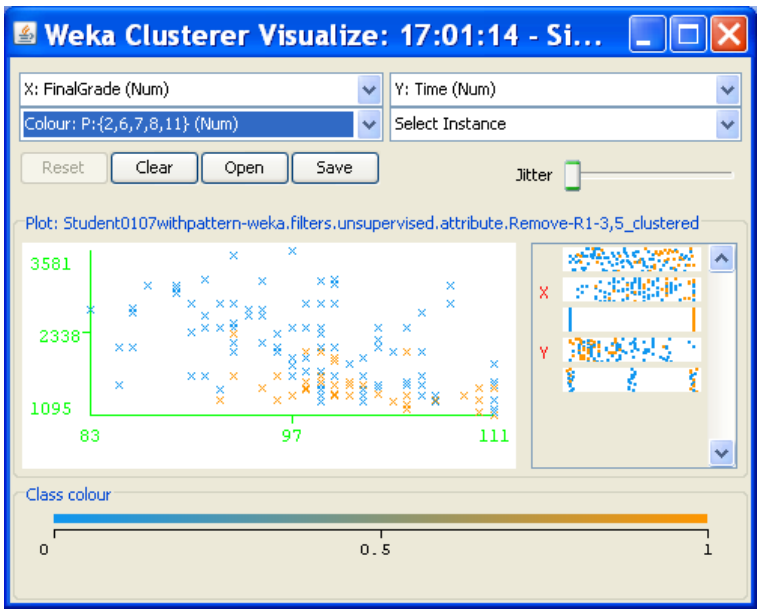


Figure 5.22: Students of data set 2) with respect to `FinalGrade` and `Time` with $K = 3$ and pattern P_3 in evidence.

```
=== Run information ===

Scheme:       weka.clusterers.SimpleKMeans -N 3 -A "weka.core.
              EuclideanDistance
              -R first-last" -I 500 -S 123
Relation:     Student0107withpattern-weka.filters.unsupervised.
              attribute.Remove
-R1-3,5
Instances:    141
Attributes:   3
              FinalGrade
              Time
Ignored:
              P:{2,6,7,8,11}
Test mode:    evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 6
Within cluster sum of squared errors: 5.210414793664353
Missing values globally replaced with mean/mode

Cluster centroids:
                        Cluster#
Attribute    Full Data         0         1         2
                 (141)      (59)      (44)      (38)
=========================================================
FinalGrade      99.539   98.4407        94  107.6579
Time         1952.0284 1728.9661 2697.5909 1435.0789


Clustered Instances

0       59 ( 42%)
1       44 ( 31%)
2       38 ( 27%)
```

Table 5.18: Log of the results obtained by WEKA *K*-means on the students data set 2) with K = 3.

	Data set 1	**Data set 2**
SPAM with semester and minsup=0.5	1261	234
CloSpan with semester and minsup=0.5	1060	233
SPAM with delay and minsup=0.3	3853	708
CloSpan with delay and minsup=0.3	2181	646

Table 5.19: Comparing outputs of `SPAM` and `CloSpan`.

that do not have supersets with the same support). In particular, we can observe that the difference is more significant when the execution of `SPAM` produces a great number of patterns. We can note this fact by reading the values in the first column of Table 5.19, that is, by considering the results concerning data set 1).

5.5 ANALYSIS OF DELAYS DISTRIBUTIONS

A statistical analysis can be accomplished by data verifying (auditing) with the aim to know if data are generated by a particular model. Among the analysis of this kind we are interested to the analytical representation of statistical distributions. It consists in finding a mathematical function that represents an observed statistical phenomenon. Often in statistics we have to deal with the following problem: we have some quantitative observations $x_1, x_2, .., x_n$ and we wish to verify if these observations (our samples) originate from a particular population characterized by a density function that we know through an analytical form. In the analytical representation we can single out the following phases:

1. choice of the function (model) that best fits the characteristics of the data distribution;
2. estimation of the parameters of the selection function;
3. calculating the degree of matching of the observed frequencies compared with those obtained with the theoretical model.

A frequency distribution is one of the most common graphical tools used to describe a single population. It is a tabulation of the frequencies of each value (or range of values). There is a wide variety of ways to illustrate frequency distributions, including histograms, relative frequency histograms, density histograms, and cumulative frequency distributions. Histograms show the

frequency of elements that occur within a certain range of values, while cumulative distributions show the frequency of elements that occur less than a certain value. A frequency distribution is the first form of summary statistical data, because it summarizes the information contained in the set of individual values.

5.5.1 *Analysis of the real case*

Before to present our analysis, we precise that it is not a traditional statistical analysis; in Section 5.2 we performed a distribution analysis for some attributes representing the careers of students, also by using some simple functions of the software `WEKA`; we know the distributions of the time to graduate, of the grade obtained at the final examination and for each exam, and so on. This study began by observing the delay distribution for each exam; in particular we observed the delays concerning 16 exams taken by 141 graduate students enrolled between 2001-2002 and 2007-2008 academic years, that is, students corresponding to data set 2) introduced in Section 5.4.

The analysis shown so far concerns the perspective of the student, who evaluates how difficult and important an exam is, in order to decide to take it immediately at the end of the course, or delay it as much as possible. In this section we propose the analysis of the student database from the perspective of each course, by analyzing the distribution of students with respect to the delay with which they take an examination, to discover common characteristics between two or more courses.

Usually, *good* students try to pass early every exam, but *not so good* students prefer to postpone most exams, especially if they are considered too difficult or too technical. We are interested in studying the delay distribution of every exam in the hypothesis that it is a good parameter for classifying students and/or courses.

In general, delays conform to some Poisson distribution, with average (and variance) λ and probability mass function

$$P_\lambda(k) = e^{-\lambda} \cdot \lambda^k / k!$$

for $k \geqslant 0$. The Poisson distribution is discrete and, in our case, k represents the delay of the exam from the end of the course, measured in semesters. So, if N is the number of students, $P_\lambda(0) \cdot N$ is the number of those who passed the exam within the first semester; $P_\lambda(1) \cdot N$ are the students who passed during the second semester, and so on. Finally, the distribution is unimodal and attains its maximum value at $k \approx \lambda$.

If we look at the actual distributions of students with respect to the delay with which they took their examinations (Figures 5.23 and 5.24 show two examples, the first about a Computer Science exam, *Databases and Information Systems*, the

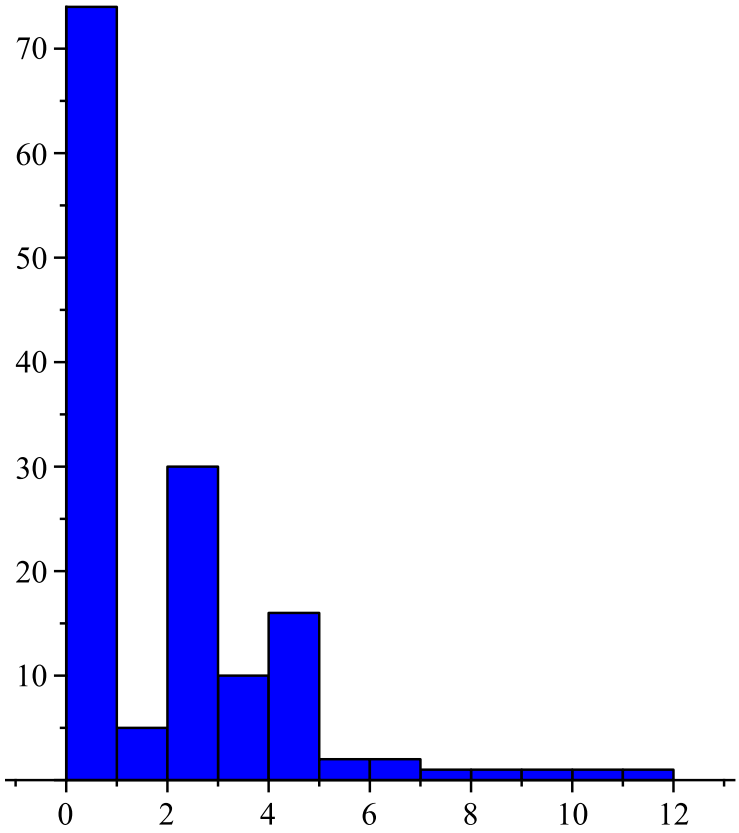


Figure 5.23: Distribution of delay of exam Databases of Information Systems.

second about a Mathematics exam, *Integral Calculus*), we observe that most of them are bimodal, with a sharp peak at $k = 0, 1$ and a second and smoother peak at $k = 5$ or $k = 6$. It is important to observe the very different behaviors of these two distributions; such as we will point out later, we can infer that there are many students who delay more the *Integral Calculus* exam than the *Databases and Information Systems* exam.

The obvious interpretation is that there are two different distributions, the first one relative to *good* students and the second relative to *not so good* students, who delay their exams of about two years. The two distributions are superimposed and generate the two peaks.

In other words, by examining the distributions for each exam, we can infer that students are divided into two classes:

1. students who tend to take an exam as soon as a course is terminated;
2. students who delay difficult exams to the end of their career.

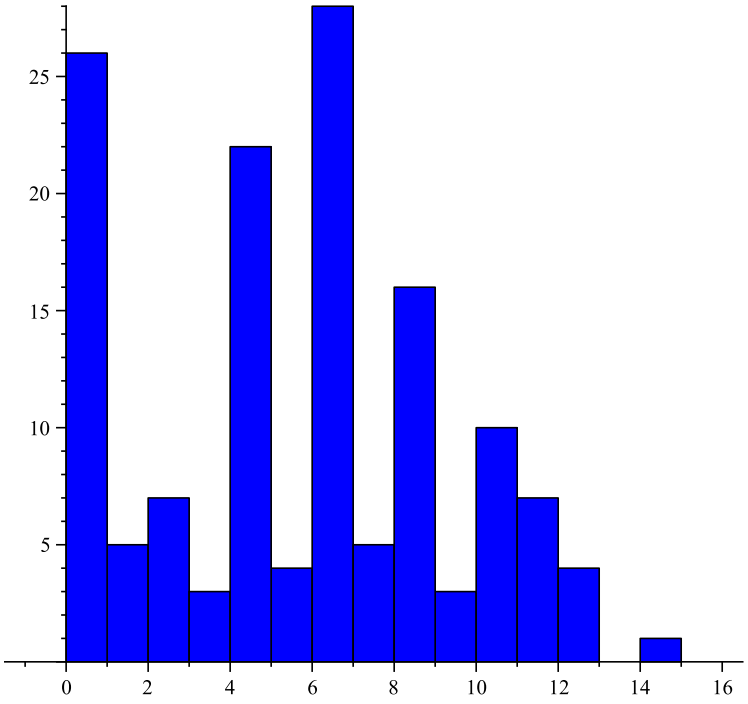


Figure 5.24: Distribution of delay of exam Integral Calculus.

This does not mean that we analyze whether a student taking an exam with some delay then also takes other exams with the same delay; we now are interested in studying the beheviour of exams in terms of delays with which they are taken by students. In order to analyze this behavior in a more formal way, we need to find the two Poisson distributions. We consider n courses $c_1, c_2, \cdots, c_n$ taken by N students and a database containing, for each course c_i, the number of students $D_{c_i}(k)$ which take the exam with delay k, for $k = 0, \cdots, d_i$, where d_i is the maximum delay relative to course c_i. We then use the following algorithm to determine the average values λ_g and λ_{ng} characterizing the two Poisson distributions and the corresponding numbers $N(\lambda_g)$ and $N(\lambda_{ng})$ of students. We can make the hypothesis that the λ_g-distribution decreases very fast so that it reduces to $k = 0, 1, 2$ as meaningful values. This approach may seem similar to the algorithm *EM* but, such as we will discuss in the section of conclusions, our analysis proceeds in a different way although reaching similar results to those obtained with *EM*. We observed the behaviors of distributions of delays expressed in semester, but in the following approach we considered the delay in years, because this type of aggregation seems appropriate to analyze this aspect of student data. In the case of exams corresponding to Figures 5.23 and 5.24, in semester we have the two sequences of delays

$$(74,\ 5,\ 30,\ 10,\ 16,\ 2,\ 2,\ 1,\ 1,\ 1,\ 1,\ 1)$$

and

$$(26,\ 5,\ 7,\ 3,\ 22,\ 4,\ 28,\ 5,\ 16,\ 3,\ 10,\ 7,\ 4,\ 0,\ 1),$$

that, in years, become the sequences

$$(79,\ 40,\ 18,\ 3,\ 2,\ 2)$$

and

$$(31,\ 10,\ 26,\ 33,\ 19,\ 17,\ 4,\ 1).$$

This means that for the exam *Databases of Information Systems* there are 74 students who took it without delay, 5 students who took it with delay of 1 semester, 30 students with a delay of 2 semesters, and so on. The delays in years are obtained by adding two by two the delays in semesters.

With this assumption, our first step consists in separating the first two values of sequences from the rest and try to approximate the λ_{ng}-distribution. We iterate this approximation process until a fixed point is obtained. This process can modify the values for $k = 0$ and $k = 1$, so that we have to use these new values to approximate the λ_g-distribution. Again, we proceed until a fixed point is found. The algorithm stops here returning, for each course, the two desired approximations.

For example, let us consider the course *Operating System* having the delays distribution, in semesters, $(67,\ 8,\ 21,\ 10,\ 17,\ 1,\ 6,\ 3,\ 4,\ 1,\ 1,\ 2)$, and, in years, $(75,\ 31,\ 18,\ 9,\ 5,\ 3)$; the decreasing rate is too large for being relative to a single Poisson distribution with the same mean $\lambda = 0.915$; in fact, this mean corresponds to the distribution $[56,\ 52,\ 24,\ 7,\ 2,\ 0]$, very far from our empirical values. Therefore, we isolate the first two values and try to approximate the others $[0,\ 0,\ 18,\ 9,\ 5,\ 3]$ using the relative average $\lambda_{ng} = 2.8$. This affects the first two positions, giving $[2,\ 6,\ 8,\ 8,\ 5,\ 3]$, the actual average of which is $\lambda_{ng} = 2.42$, a better approximation to the empirical values. We iterate the process until we find a fixed point, that is, a distribution generating itself. After nine iterations we find the distribution $[8,\ 16,\ 18,\ 9,\ 5,\ 2]$, thus explaining the behavior of more than 40% of the students, who prefer to delay this exam.

As observed before, these values modify the distribution of the first two positions relative to the *good* students. This becomes $[67,\ 15,\ 0,\ 0,\ 0,\ 0]$ and an average value of 0.18. We assume for λ_g this new value and, similarly to the previous approximation, iterate the process until we find a fixed point. We stop by finding $\lambda_g = 0.20$, corresponding to the distribution $[67,\ 14,\ 1,\ 0,\ 0,\ 0]$. In this way, we have explained the behavior of *good* students, corresponding to the 58% of the total. Summing up the two distributions we find $[75,\ 30,\ 19,\ 9,\ 5,\ 3]$, very close to the empirical distribution $[75,\ 31,\ 18,\ 9,\ 5,\ 3]$; in fact, we computed $\chi^2 = 0.24$, to be considered together with five degrees of freedom. Table 5.20 illustrates the details of this procedure, performed by using the software `Maple`.

Empiric distribution $[75, 31, 18, 9, 5, 3]$ **Starting distribution** $[0, 0, 18, 9, 5, 3]$

$$[0, 0, 18, 9, 5, 3], \lambda_{ng} = 2.80$$

$$[2, 6, 18, 9, 5, 3], \lambda_{ng} = 2.42$$

$$[4, 9, 18, 9, 5, 3], \lambda_{ng} = 2.30$$

$$[5, 12, 18, 9, 5, 3], \lambda_{ng} = 2.12$$

$$[6, 13, 18, 9, 5, 3], \lambda_{ng} = 2.100$$

$$[7, 14, 18, 9, 5, 3], \lambda_{ng} = 2.00$$

$$[8, 15, 18, 9, 5, 3], \lambda_{ng} = 1.95$$

$$[8, 16, 18, 9, 5, 3], \lambda_{ng} = 1.93$$

$$[8, 16, 18, 9, 5, 3], N(\lambda_{ng}) = 59$$

$$[67, 15, 0, 0, 0, 0, 0], \lambda_{g} = 0.18$$

$$[67, 15, 1, 0, 0, 0, 0], \lambda_{g} = 0.20$$

$$[67, 14, 1, 0, 0, 0], N(\lambda_{g}) = 82$$

Theoric distribution $[75, 30, 19, 9, 5, 3]$

Table 5.20: Steps for the approximation of the delays distribuition of the Operating System exam.

This example concerns a Computer Science course of the second year. It is an important exam so we could have expected most students would prefer to give at once, as it happens for other Computer Science exams.

We applied the algorithm (by using the software for symbolic calculus `Maple` [37]) to $n = 15$ courses taken by $N = 141$ students in Computer Science at the University of Florence. The analysis confirmed that for each course c_i we have

$$D_{c_i}(k) \sim P_{\lambda_{g_i}}(k) \cdot N(\lambda_{g_i}) + P_{\lambda_{ng_i}}(k) \cdot N(\lambda_{ng_i}),$$

with a good approximation.

In particular, we found that some Computer Science exams are characterized by $\sum_{k \geqslant 0} P_{\lambda_g}(k) \sim 70\%$. Instead, Mathematics exams are delayed and often appear as the last exams taken before the final examination.

Table 5.21 shows these results sorted by average grade (**avg-grade**); more than 60% of students prefer to take Computer Science immediately; instead, Mathematics exams are delayed and often appear as the last exams given before the final examination. Strangely, the exam of Physics is given immediately by the 81% of all the students. Figures 5.25 and 5.26, by representing delays aggrega-

tion for Computer Science exams and for Mathematics exams, show clearly this behavior.

Exam	λ_g	$\%\lambda_g$	λ_{ng}	$\%\lambda_{ng}$	χ^2	**avg-grade**
Languages and Compilers	0.12	79	1.1	21	0.7	25.94
Numerical Calculus	1	10	2.89	90	5.15	25.92
Computer Network	0.12	90	2.47	10	0	25.9
Physics	0.15	81	2.11	19	0.64	25.9
Computer Architecture	0.13	79	3.09	21	2.59	25.85
Concurrent Programming	0	20	1.8	80	3.43	25.8
Operating System	0.22	58	1.89	42	0.24	25.48
Programming	0.16	78	2.26	22	0.47	25.39
Databases and Information Systems	0.29	60	1.4	49	0.98	25.3
Algorithms and Data Structures	0.3	77	2.38	23	1.55	25.29
Programming Methodologies	0.24	59	1.73	41	1.26	24.85
Theoretical Computer Science	0.29	41	2.29	59	1.53	24.14
Integral Calculus	0	16	2.84	84	9.55	23.74
Differential Calculus	0	38	2.57	62	9.6	22.76
Probability and Statistics	1.27	15	1.92	85	8.39	20.88

Table 5.21: Results about approximations of the delays distribution of some exams.

For details, the delay distributions of some exams shown in Table 5.21 are presented in Appendix B, with Figures 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7 and 8.8, beyond those of the examples given in this section.

In the next chapter we present some conclusion of our analysis; for what concerns the analysis of delays distribution, we can anticipate that the analyzed students are really divided in two groups. The first group of students who take exams (in particular, Computer Science exams) as soon as the corresponding course is terminated; the second group of students who delay difficult exams (in particular, Mathematics exams) to the end of their career.

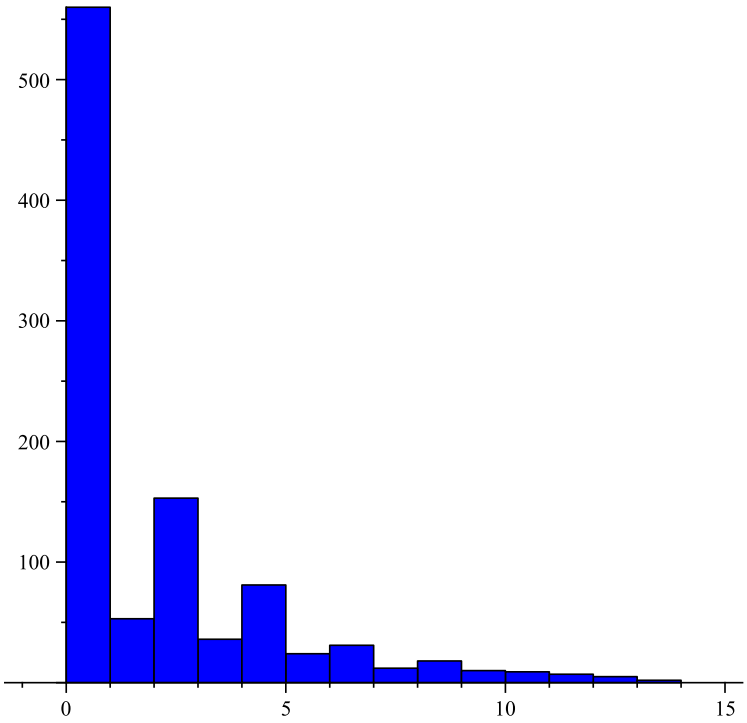


Figure 5.25: Distribution of delays of Computer Science exams.

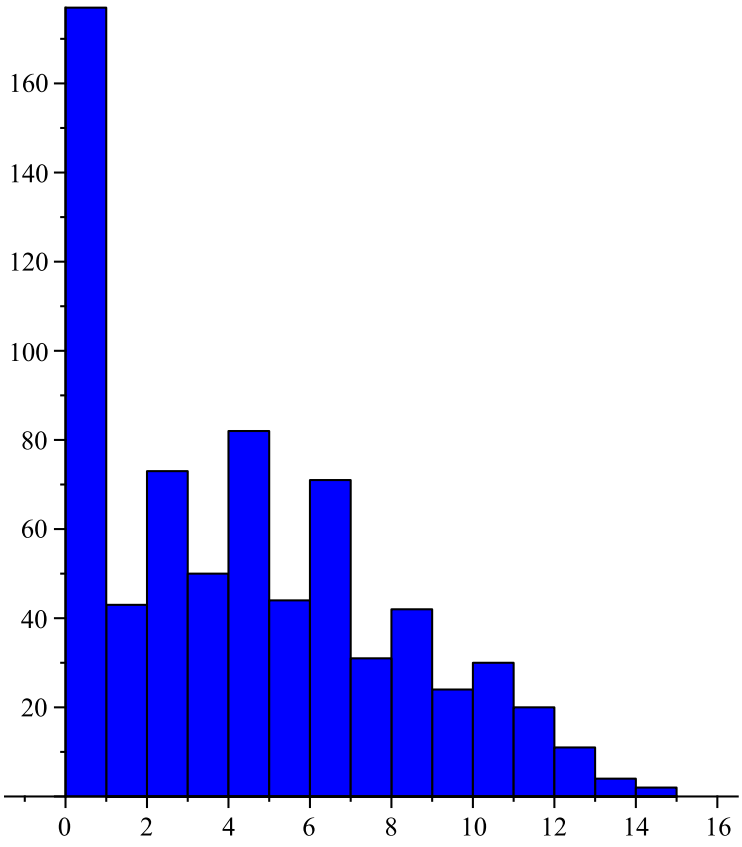


Figure 5.26: Distribution of delays of Mathematics exams.

6

CONCLUSIONS AND FUTURE ANALYSIS

In this thesis we proposed a data mining methodology to study the behavior of university students in terms of their careers, by analyzing the corresponding database, and, as a matter of fact, we found many interesting relationships among data. By introducing the concept of ideal career, that is, the career of the ideal student who takes each exams at the end of the corresponding course, without delay, we defined the distance between the career of a student and the ideal career. We defined the career of a student in different ways, by considering the temporal information or not, according to the method of analysis used; for traditional clustering analysis we introduced an ideal career that was a sequence of exams, without the temporal information. In our case study, illustrated in Chapter 5, we showed that the results obtained with one or the other representation of the ideal career are similar. For the frequent patterns analysis we represented a career as a sequence with temporal information, as this technique requires. We combined some data mining techniques to analyze students data and used also the classification technique, based on decision trees, to deepen the clustering analysis.

By applying clustering techniques on the student database of the Computer Science degree program at the University of Florence, we obtained significant results showing that students can be divided into two groups according to their distance (for example the **Bubblesort** distance defined in 4.1), from the ideal career: students who graduated relatively quickly and with high grades and students who obtained worse results. Luckily, we observed that students in the first group are characterized by *small* values of **Bubblesort** distance while students in the second group have *large* values. We point out that it is theoretically possible to be close to the ideal career and to have a large delay. In our case study this does not happen; however, if this happened, that is, if many good students had large distances from the ideal career, we could make other considerations, such as that the degree program probabily needs to be reorganized. The obtained results confirm that the more students follow the order taken by the ideal path, the more they obtain good performance in terms of graduation time and final grade. As an important consequence of this, we can conclude that the degree program in the period under examination was well structured. Also the results obtained with classification techniques, with different choices of attributes and class, confirmed the previous results: for example, classifying students with respect to *small* ($\leqslant 100$) and *large* (> 100) values of **Bubblesort** distance, we obtained that students with small distance have better career than students with large distance, in term of final grade and time to complete their

studies. We do not forget that the opposite conclusions could be true, that is, only the good students could complete the program as intended. This does not seem to be in our case study. We intend to deepen this aspect by using a random ideal career, that is, by performing different tests with different randomly generated careers. However we expect that the choise of the ideal career is conditioned by the data set being analyzed, that is, by the degree program under analysis.

By using the frequent pattern technique we investigated issues for mining sequential patterns in a database of university students. As far as we know and as we referred in Section 4.2, this is one of the first attempts to use the sequential pattern technique in the context of education. In particular, we proposed a methodology that consists in the following main steps:

i) the generation of the frequent patterns by using the `SPAM`, or `CloSpan`, algorithm;

ii) the selection of the most interesting patterns and, finally

iii) the cluster analysis based on the results of the previous step.

For what concerns the first step, we considered two possible types of temporal information: the *semester* in which students take exams and the *delay* with which exams are taken. A critical step concerns the selection of patterns and in Section 4.2.2 we give some hints to find the most interesting ones; however during this phase it is necessary a deep knowledge of the context under examination. For example, by searching frequent patterns for the database under analysis, we found the frequent pattern $\langle\{4\}\{8\}\{16,17\}\{12\}\rangle$ with the support 0.5, (that is, verified by the 50% of students) that allow us to understand which exams, or typology of exams, are considered difficult by students. In fact, by referring to Table 5.5, that contains the exam names corresponding to the codes involved in this pattern, where the ideal career corresponds to the exam codes sorted in ascending order, we can see that Mathematics exams tend to be considered more difficult than Computer Science exams (this explains why the exam with code 12 follows the exams with codes 16 and 17 in the pattern). In order to find if there is a relation between the curriculum of students and the patterns, we decided to use them as binary attributes of our database and tried to cluster students by using the `K-means` algorithm. We can wonder whether information such as the sex, the place of birth, the grade obtained at the high school level and the year of enrollment at the university have a correlation with the frequent pattern analysis. For example, in the case study examined in this thesis, the patterns P_1 and P_2 (defined in Section 5.4.1 and corresponding to the semester and to the delay as temporal information) divide students into *good* and *not so good*, according to the final grade and the length of studies. Pattern P_1 is a subsequence of the ideal career while P_2 corresponds to a 5-sequence of computer

science exams taken without delay. In other words, also by performing the frequent patterns analysis, we obtained that the good students have taken most exams according to the order planned by the degree program, and this can mean that it was well structured. Moreover, they have taken several Computer Science exams without delay, which did not happen for Mathematics exams.

We proposed also an analysis technique to study the data of university students by the prospective of courses. We presented in Section 5.5 an idea to approximate the distributions of exams delays, by starting from the observation that students are divided into two groups: *good* students who pass early each exam, *not so good* students which postpone most exams, especially if they are considered too difficult or technical. We illustrated the model and the algorithm by using a mixture of two Poisson distributions to determine the approximations of good students and not so good students, obtaining results, in terms of delays' medium values and their frequencies, showing that about 70% of students take Computer Science exams early, but Mathematics exams are delayed and often appear as the last exams taken before the final examination. Our approach is similar to that of the *Expectation Maximization* (`EM`) algorithm, which however is based on a different argument: given a guess for the parameter values, the `EM` algorithm calculates the probability that each point belongs to each distributons and then uses these probabilities to compute a new estimate for the parameters, which are the ones that maximize the likelihood. We find an implementation of this algorithm in `WEKA`; for a short tutorial of it we refer to [13], while for a complete discuss about it we refer to [12]. We are conscious that we have adopted some simplifications for what concerns the correct statistical approach. In fact we know that using the Poisson distribution implicates several hypothesis, such as having a large number of students with the same career in term of time (on the contrary our students graduated in different times), or having a large number of delays (in semester). We intend to deepen the delays distribution analysis by using the *Poisson Lognormal Distribution* [16], that seems to describe well this phenomenon. This distribution is characterized by a λ following a lognormal distribution and this feature allows to extend the characterization of students to various ranges of students (respect to good and not so good students) and to assign to each student a value of λ that characterizes him/her.

We deepened our students analysis by investigating the correlation indexes of clusters obtained by applying the clustering model, such as we did in Section 5.3; we evaluated clustering by using the *Pearson's correlation* and the *cosine similarity* applied to the linear representations of the proximity matrix and of the incidence matrix, but we can deep the evaluation step by considering other measures, such as it is discussed in [92]. In the same direction, we intend to analyze the student databases also using the `EM` algorithm.

We wish to point out that, although we tested our models on a small database, the results presented in this thesis are intended to illustrate a methodology

which can be applied to databases of any dimension containing various curriculum data of students and corresponding to different degree program programs.

6.1 THE EVOLUTION OF THE COMPUTER SCIENCE DEGREE AT THE UNIVERSITY OF FLORENCE

In Chapter 5 we presented the case study concerning the data analysis of graduate students in Computer Science at the University of Florence (Italy) enrolled during 2001-2002 and 2007-2008 academic years, according to Ministerial Decree n. 509/1999. We considered the *laurea triennale*, that is, the academic degree under analysis is structured in three years. After the period under examination the Computer Science degree is always organized in three years, but its scheduling has been modified and we had in this last year the first graduate students, that is, we have too few students and few data to perform a significant analysis. As we illustrated in Section 5.1, in the period under analysis the Computer Science degree had different organizations: during the academic years from 2001-2002 to 2003-2004 there were five curricula, in the other period students could choose between two curricula. This change was done because students enrolled in 2001-2003 mostly chose the first two curricula, that were merged in the first curriculum of the enrollment period 2004-2007. Our study analyzed data corresponding to different points of view, considering several subsets corresponding to a particular curriculum, students enrolled in different periods and choosing different curricula, besides analyzing all students, by considering only common exams. In every of these cases, we obtained the same important result: the more students follow the order of exams provided from the degree program scheduling, the more they obtain good performance in terms of graduation time and final grade. This fact could mean that the degree program is well structured. On the other hand, by analyzing the delays distributions of exams, we obtained that the Mathematics exams are delayed from a large part of students: perhaps the scheduling of these exams may be changed, or it is necessary that students come to the University with a better knowledge of Mathematics. We point out that in the analysis models proposed in this thesis, the results obtained by students at the highschool level are considered, such as, for example, in the clustering model, where we tried to use also the attribute corresponding to the grade obtained to highschool, and in the classification model, where we used the information about the type of highschool. We observe that, for a good university career, the type of highschool seems to be more important than the grade obtained at the highschool level; this also emerged from the correlation analysis illustrated in Section 5.3, that put in evidence low values of correlation between the attributes HighGrade and Time, and between HighGrade and FinalGrade.

We considered students careers with many degrees of freedom because in the period under analysis the degree program of Computer Science at the University of Florence did not have formal constraints between exams, but only a curriculm highly recommended. It would be interesting to apply our models to a degree program having some more constraints between exams to know if it possible to come up with similar results also when the ideal career has more constraints (for example, the Computer Science program degree in the last years, when some constraints are been fixed).

In order to deepen the analysis and the discussion about the university students performances, we have already began to perform other types of analysis on students data; in particular we intend to study the performance of students in Computer Science compared with the results of the entrance tests, and the performance of same students compared with the results of the evaluation of courses done by students. We expect that the Data Mining techinques can give interesting results also for these studies.

7

APPENDIX A

The following three figures show results with `K-means` algorithm by using the clustering attributes **Area** distance, **Time** and **FinalGrade** on students of Curriculum 1.

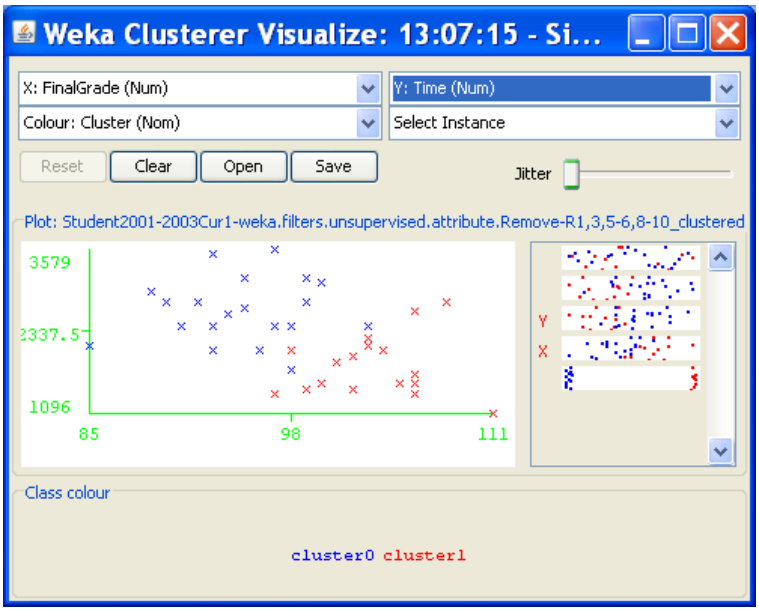


Figure 7.1: Students of Curriculum 1 with respect to `FinalGrade` and `Time`.

Figures 7.4 and 7.5 show results with `K-means` algorithm by using the clustering attributes **Area** distance, **Time** and **FinalGrade** on all students.

Figures from 7.6 to 7.11 show results with `K-means` algorithm by using the clustering attributes **BubblesortSem** distance, **Time** and **FinalGrade** on students of Curriculum 1 and 2.

For what concerns the results obtained with the `J48` algorithm, Figures 5.17 and 5.18 show the decision trees obtained on students data of Curriculum 1, by choosing **Bubblesort** and **Time** class respectively.

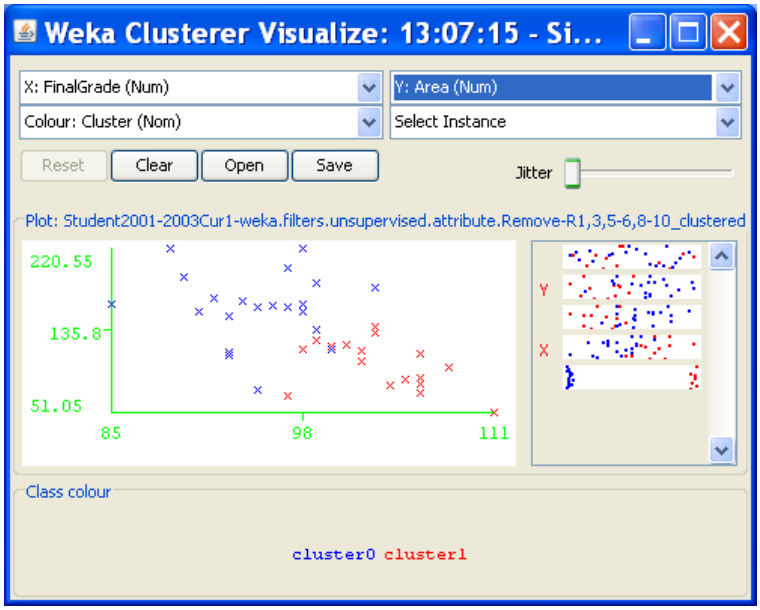


Figure 7.2: Students of Curriculum 1 with respect to `FinalGrade` and `Area` distance.

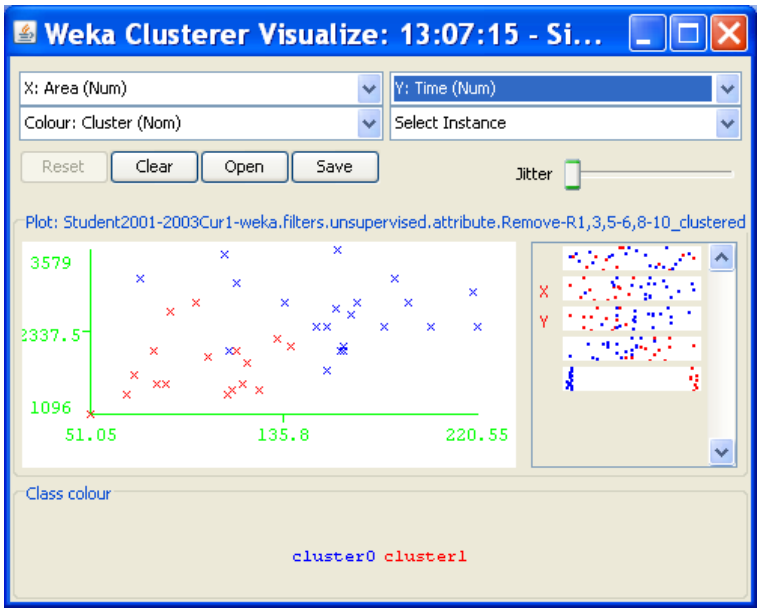


Figure 7.3: Students of Curriculum 1 with respect to `Area` distance and `Time`.

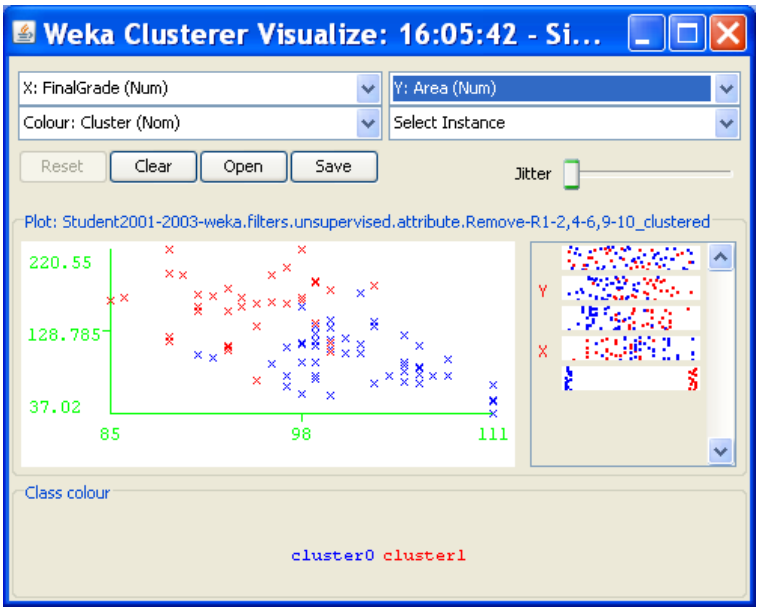


Figure 7.4: Students with respect to `FinalGrade` and `Area` distance.

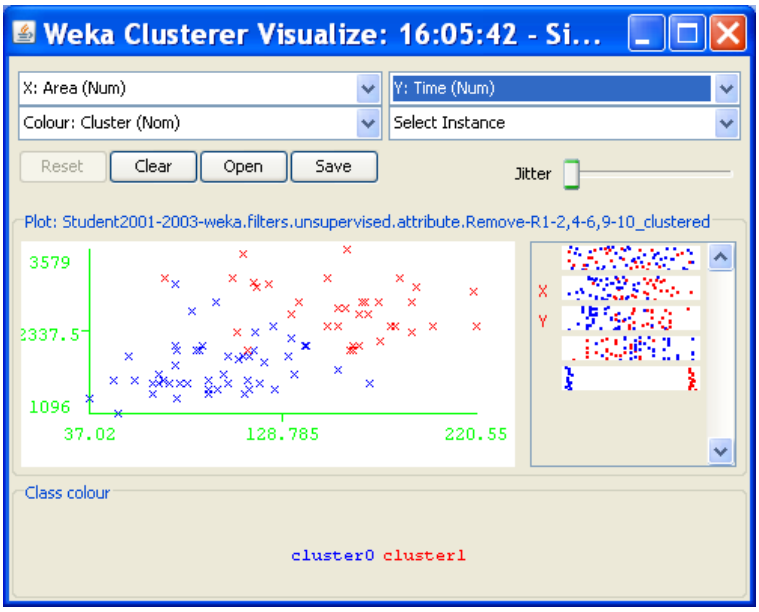


Figure 7.5: Students with respect to `Area` distance and `Time`.

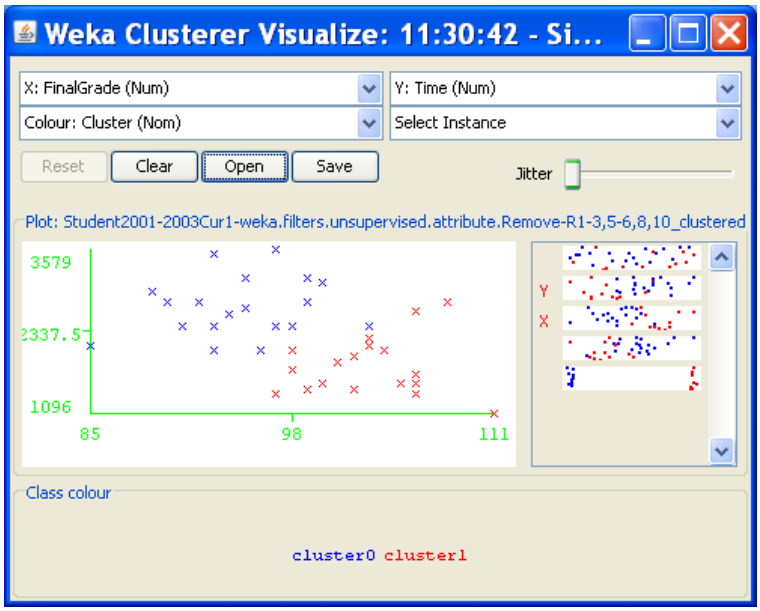


Figure 7.6: Students of Curriculum 1 with respect to `FinalGrade` and `Time` by using `BubblesortSem` distance.

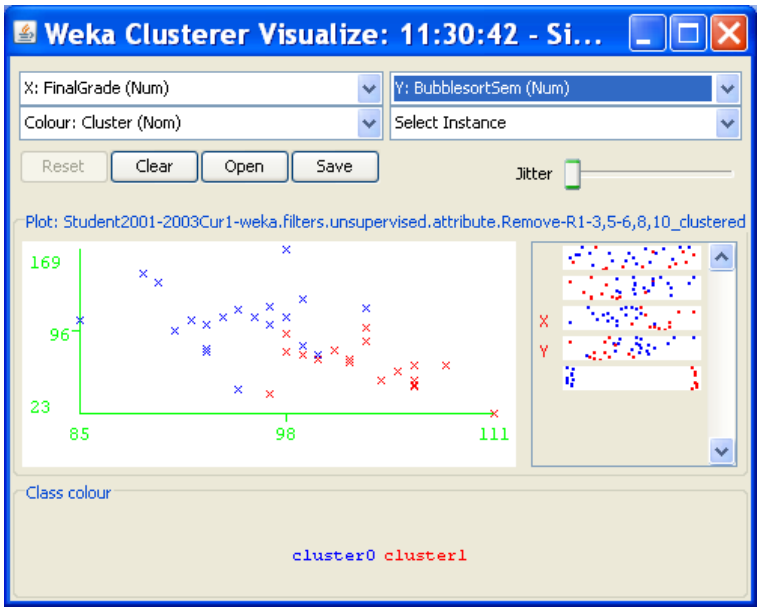


Figure 7.7: Students of Curriculum 1 with respect to `FinalGrade` and `BubblesortSem` distance.

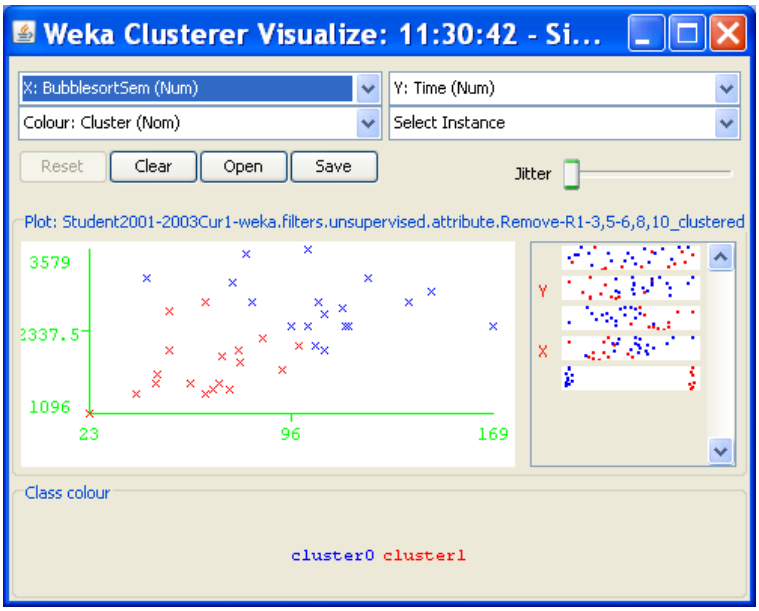


Figure 7.8: Students of Curriculum 1 with respect to `BubblesortSem` distance and `Time`.

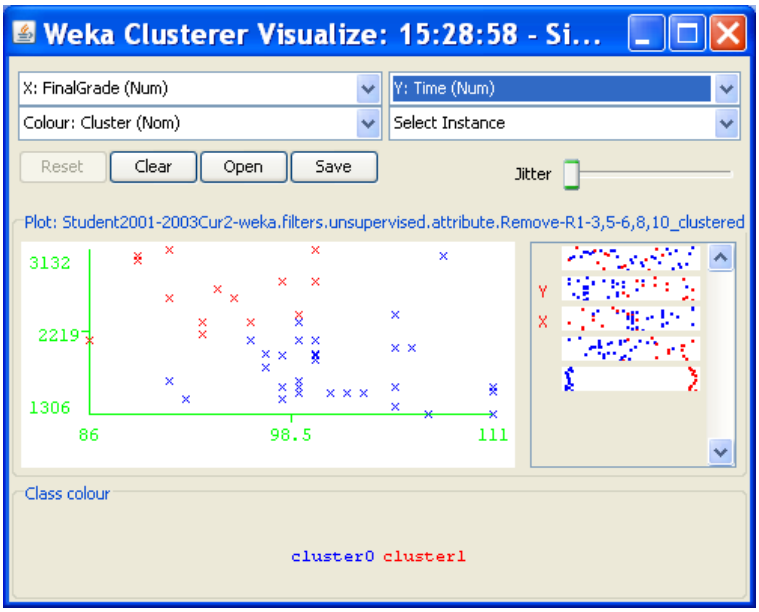


Figure 7.9: Students of Curriculum 2 with respect to `FinalGrade` and `Time` by using `BubblesortSem` distance.

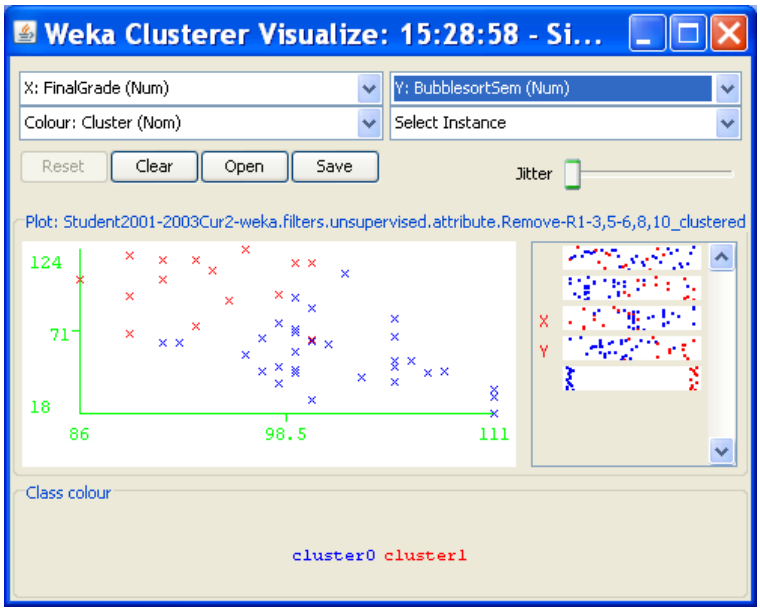


Figure 7.10: Students of Curriculum 2 with respect to `FinalGrade` and `BubblesortSem` distance.

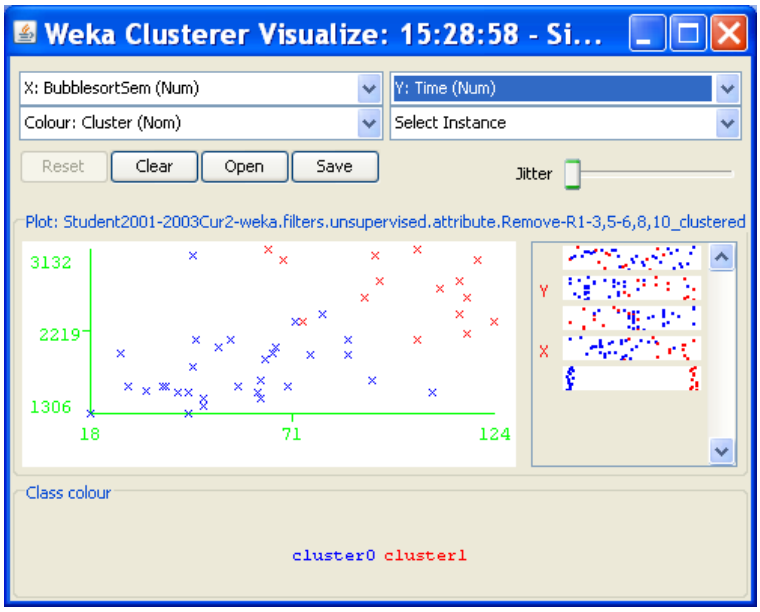


Figure 7.11: Students of Curriculum 2 with respect to `BubblesortSem` distance and `Time`.

8

APPENDIX B

This appendix collects figures showing the graphical representation of the delays distribution of some exams analyzed in Section 5.5.

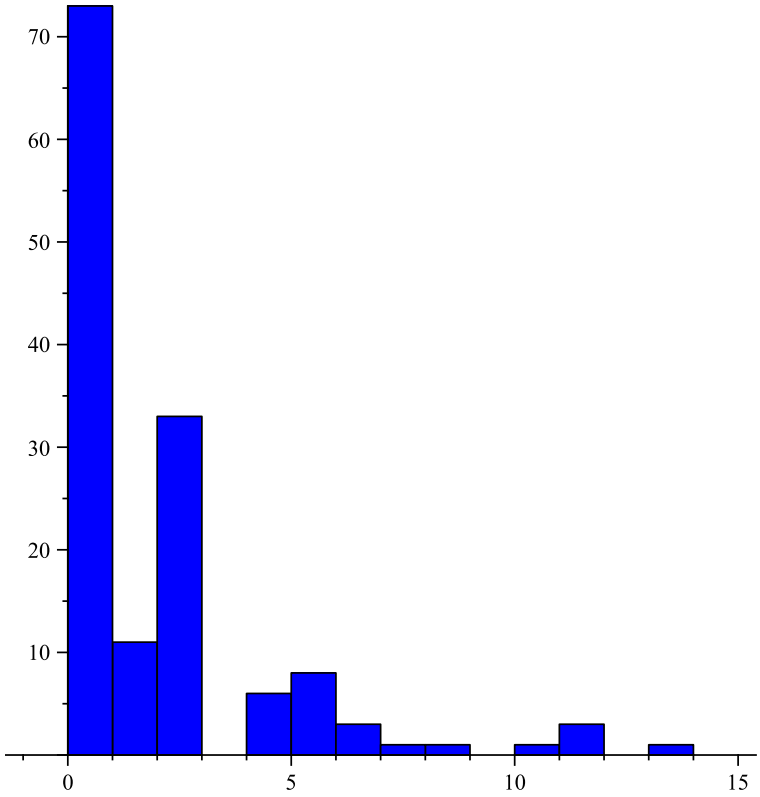


Figure 8.1: Delays distribution of exam Algorithms and Data Structures.

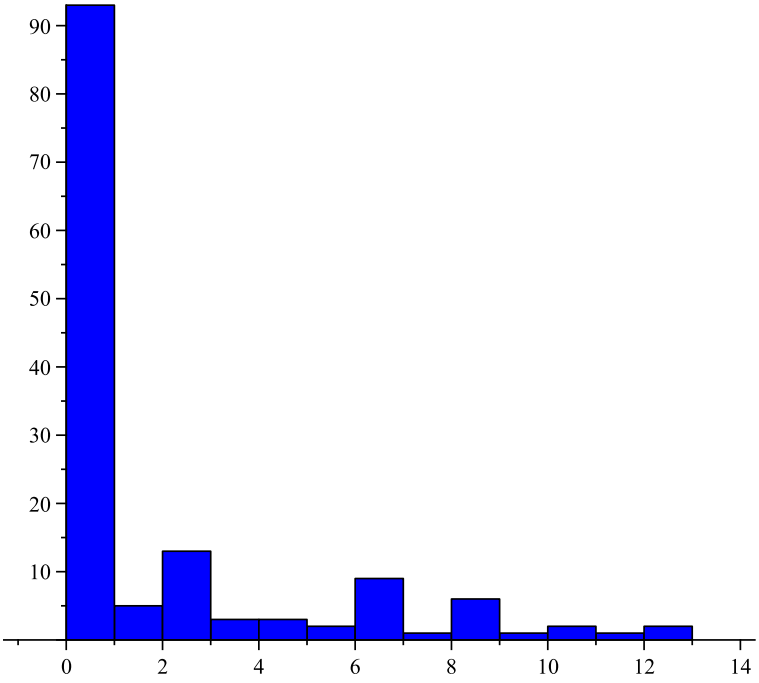


Figure 8.2: Delays distribution of exam Computer Architecture.

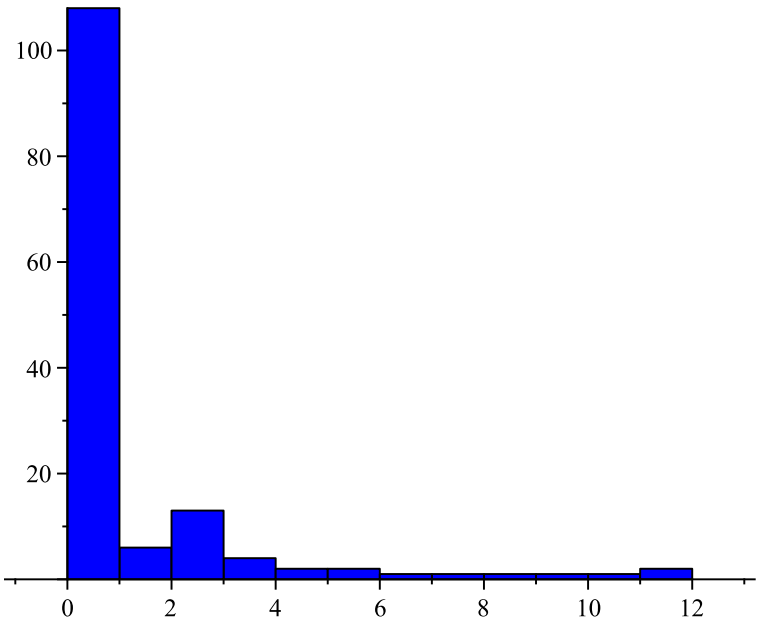


Figure 8.3: Delays distribution of exam Computer Networks.

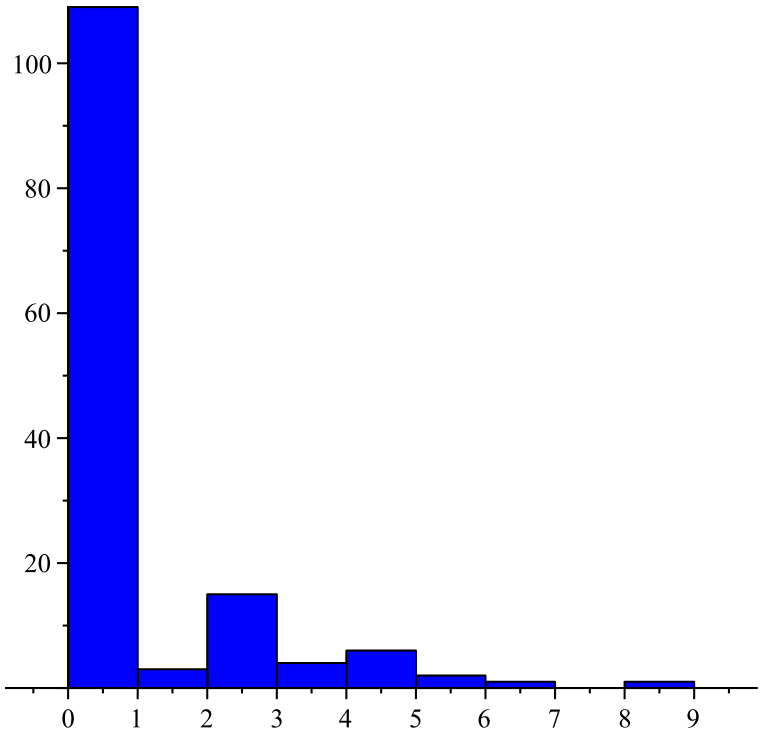


Figure 8.4: Delays distribution of exam Languages and Compilers.

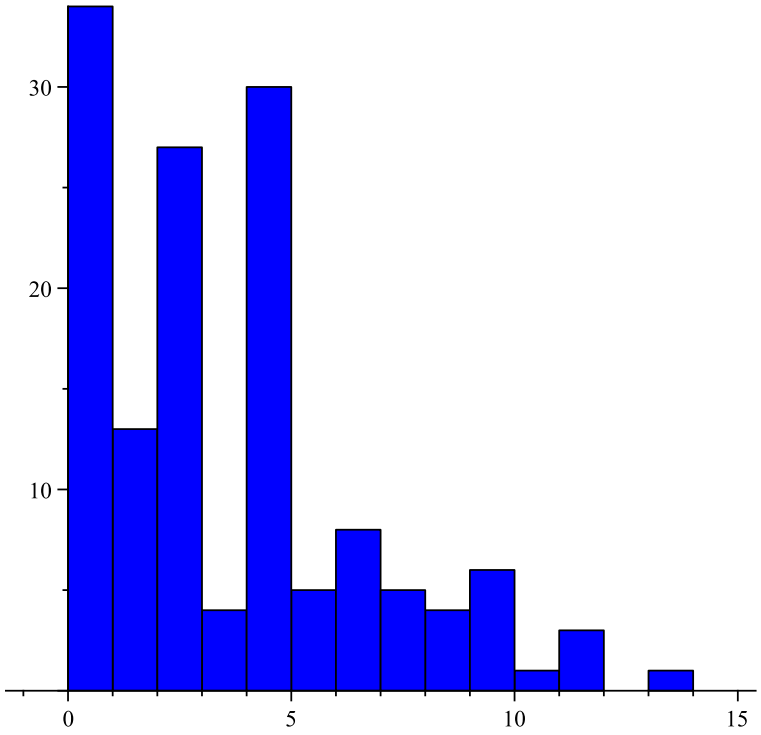


Figure 8.5: Delays distribution of exam Concurrent Programming.

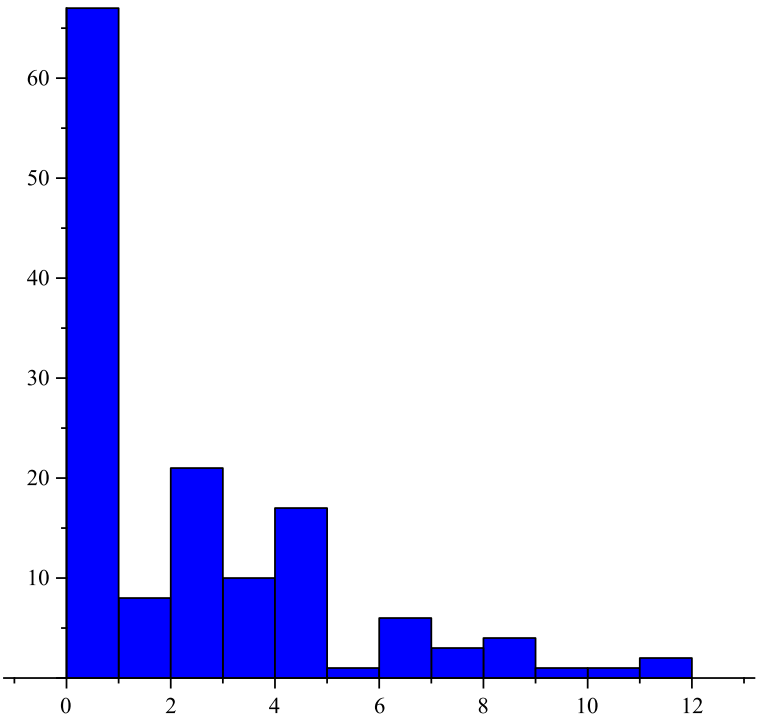


Figure 8.6: Delays distribution of exam Operating Sysyems.

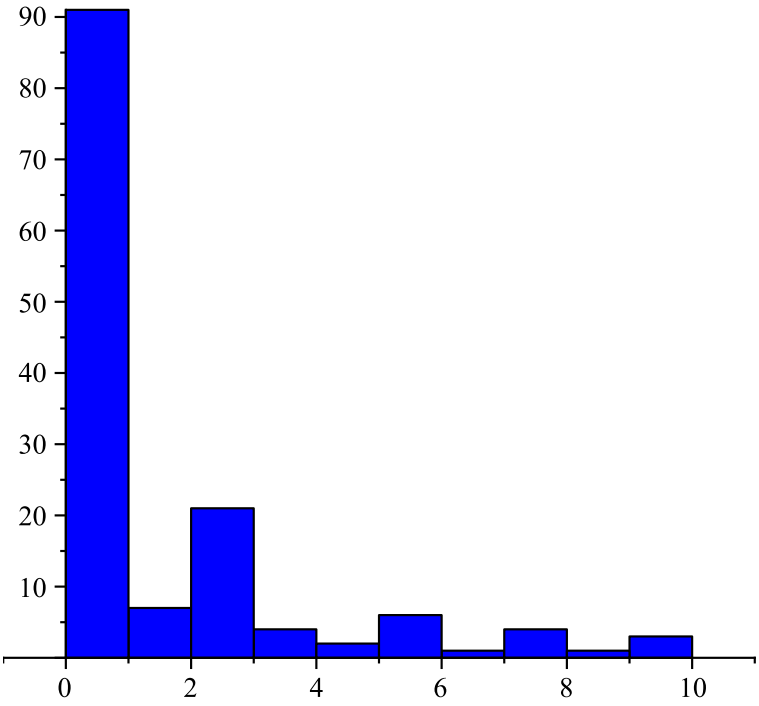


Figure 8.7: Delays distribution of exam Physics.

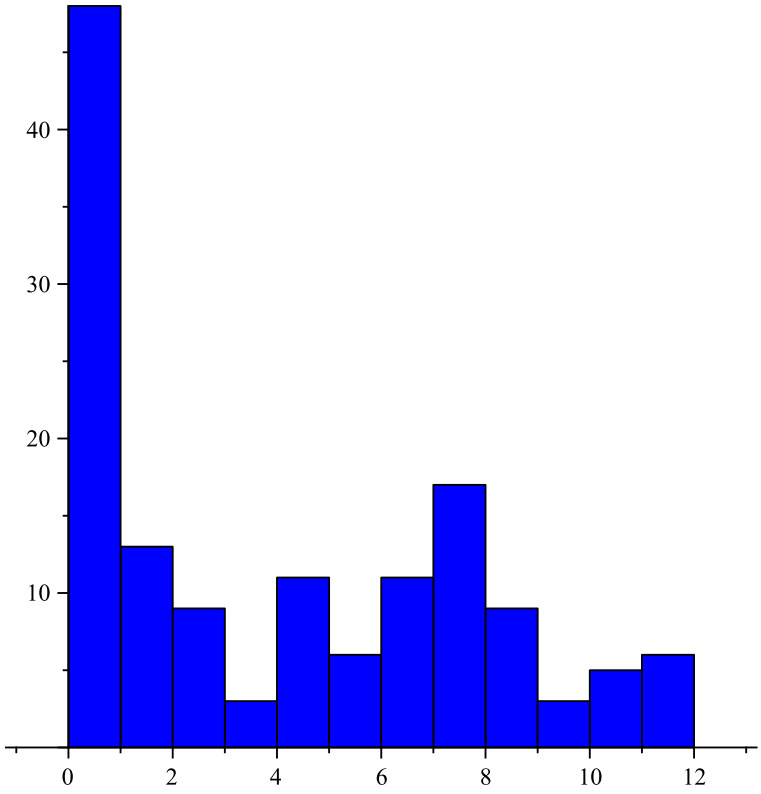


Figure 8.8: Delays distribution of exam Differential Calculus.

BIBLIOGRAPHY

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *The ACM-SIGMOD 1993 International Conference on Management of Data*, pages 207–216, Washington, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Technical Report RJ9839, IBM*, IBM Research Report RJ9839, 1993.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[4] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of 2000 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 439–450, Dallas, Texas, 2000. ACM Press.

[5] M. R. Anderberg. *Cluster Analysis for Applications*. Academic Press, New York, 1973.

[6] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. of 1999 ACM-SIGMOD Intl. Conf. on Management of Data*, pages 49–60, Philadelphia, Pennsylvania, 1999. ACM Press.

[7] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429–435, 2002.

[8] R. Baker. Data mining for education. In *International Encyclopedia of Education*, Oxford, U.K., 2010. Elsevier.

[9] R. S. J. D. Baker and K. Yacef. The state of educational data mining in 2009: a review and future visions. *Journal of educational Data Mining*, 1(1):3–17, 2009.

[10] P. Berkhin. Survey of clustering data mining techniques. In *Technical report*, San Jose, CA. Accrue software.

[11] M. J. A. Berry and G. Linoff. *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management*. 2004.

[12] J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Unpublished paper available at http://lasa.epfl.ch/teaching/lectures/ML-Phd/Notes/GP-GMM.pdf.

[13] S. Borman. The expectation maximization algorithm: A short tutorial. Unpublished paper available at http://www.seanborman.com/publications, 2004.

[14] L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1984.

[15] S. Brin, R. Motwani, and C. Silverstein. Beyound market baskets: Generalizing association rules to correlations. In *Proc.ACM SIGMOD Intl. Conf. Management of Data*, pages 265–276, Tucson, AZ, 1997.

[16] M. G. Bulmer. On fitting the poisson lognormal distribution to species-abundance data. *Biometrics*, 30(1):101–110, 1974.

[17] W. Buntine. Learning classification trees. In *Artificial Intelligence Frontiers in Statistics*, pages 182–201, London, 1993. Chapman & Hall.

[18] D. Burdick, M. Calimlim, and J. Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proc. 2001 Int. Conf. Data Engineering (ICDE'01)*, pages 443–452, Heidelberg, Germany, 2001.

[19] T. Calders. Ten-years award talk: Non-derivable frequent itemsets. In *Proc. The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases - ECML-PKDD 2012*, 2012.

[20] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *PKDD 2002 Proceedings*, pages 74–85, Helsinki, 2002.

[21] T. Calders and M. Pechenizkiy. Introduction to the special section on educational data mining. In *SIGKDD Explorations 13(2)*, pages 3–6, 2011.

[22] R. Campagni, D. Merlini, and R. Sprugnoli. Analyzing paths in a student database. In *The 5th International Conference on Educational Data Mining*, Chania, Greece, 2012.

[23] R. Campagni, D. Merlini, and R. Sprugnoli. Data mining for a student database. In *ICTCS 2012, 13th Italian Conference on Theoretical Computer Science*, Varese, Italy, 2012.

[24] R. Campagni, D. Merlini, and R. Sprugnoli. Sequential patterns analysis in a student database. In *ECML-PKDD Workshop: Mining and exploiting interpretable local patterns (IPat 2012)*, Bristol, UK, 2012.

[25] F. Castro, A. Vellido, A. Nebot, and F. Mugica. Applying data mining techniques to e-learning problems. In *Evolution of Teaching and Learning Paradigms in Intelligent Environment*, pages 183–221, New York, 2007. Springer-Verlag.

[26] S. Chakrabarti. *Mining the Web: Discovery Knowledge from Hypertext Data*. Morgan Kaufmann, San Francisco, CA, 2003.

[27] CLOSPAN. `http://www.cs.ucsb.edu/~xyan/software/Clospan.htm`.

[28] K. Daimi and R. Miller. Analyzing student retention with data mining. In *Proceedings of the 2009 International Conference on Data Mining*, pages 55–60, 2009.

[29] R. Damaševičius. Analysis of academic results for informatics course improvement using association rule mining. In *Information Systems Development*, pages 357–363. Springer, 2010.

[30] N. Delavari, M. R. A. Shirazi, and M. R. Beikzadeh. A new model for using data mining technology in higher educational systems. In *Proceedings of the Fifth International Conference on Information Technology Based Higher Education and Training*, 2004.

[31] N. Delavari, P. A. Somnuk, and M. R. Beikzadeh. Data mining application in higher learning institutions. *Informatics in Education*, 7(1):31–54, 2008.

[32] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley & Son, Inc., 2nd edition, New York, 2001.

[33] B. Dunkel and N. Soparkar. Data organization and access for efficient data mining. In *Proc. of the 15th Intl. Conf. on Data Engineering*, pages 522–529, Sydney, Australia, 1999.

[34] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Intl. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.

[35] U. Fayyad, G. P.-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, (17(3)):37–54, 1996.

[36] D. Fisher. Iterative optimization and simplification of hierarchical clustering. In *Journal of Artificial Intelligence Research, 4*, pages 147–179, 1996.

[37] F. Garvan. *The Maple Book*. Chapman & Hall, London, 2002.

[38] F. Giannotti and D. Pedreschi. *Mobility, Data Mining and Privacy - Geographic Knowledge Discovery*. Springer, 2008.

[39] M. Golfarelli and S. Rizzi. *Data Warehouse Design: Modern Principles and Methodologies*. Mc Graw Hill, 2009.

[40] M. S. Granovetter. The strenght of weak ties. pages 1360–1380. America Journal of Sociology, Vol. 78, 1973.

[41] F. H. Gaohua Gu and H. Liu. Sampling and its application in data mining: A survey. In *Technical Report TRA6/00*, Singapore. National University of Singapore.

[42] H. Guruler, A. Istanbullu, and M. Karahasan. A new student performance analysing system using knowledge discovery in higher educational databases. *Computers & Education*, 5(1):247–254, 2010.

[43] W. Hämäläinen, T. H. Laine, and E. Sutinen. Data mining in personalizing distance education courses. In C. Romero and S. Ventura, editors, *Data Mining in E-learning*, pages 157–171. WitPress, Southampton, UK, 2006.

[44] E. H. Han, G. Karipis, and V. Kumar. Min-Apriori: An algorithm for finding association rules in data with continuos attributes. `http://www.cs.umn.edu/han/`.

[45] J. Han and Y. Fu. Mining Multiple-level association rules in large databases. In *IEEE Trans. on Knowledge and Data Engineering, 11(5)*, pages 798–804, 1999.

[46] D. J. Hand. *Data Mining: Statistics and More?* The American Statistician, 1998.

[47] W. Härdle. *Smoothing Techniques with implementation in S*. Springer, 1991.

[48] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, Prediction*. Springer, New York, 2001.

[49] W. H. Inmon. *Building the data warehouse*. Wiley, 2005.

[50] A. K. Jain and R. C.Dubes. *Algorithms for Clustering Data*. Prentice Hall Advanced Reference Series. Prentice Hall, 1988.

[51] A. K. Jain, M. N. Murty, and P. J. Flynn. *Data Clustering: a review. Data Mining Techniques*. ACM Computing Surveys, 31.

[52] N. Jardine and R. Sibson. *Mathematical Taxonomy*. Wiley, New York, 1971.

[53] N. Jindal and B. Liu. Identifying comparative sentences in text documents. SIGIR-06, 2006.

[54] N. Jindal and B. Liu. Mining comparative sentences and relations. American Association for Artificial Intelligence, 2006.

[55] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. Wiley-IEEE Press, Piscataway, 2003.

[56] G. Karpis, E. H. Han, and V. Kumar. *Multilevel Refinement for Hierarchical Clustering*. Technical Report TR 99-020, University of Minnesota, Minneapolis, MN, 1999.

[57] K. Koedinger, K. Cunningham ad A. Skogsholm, and B. Leber. An open repository and analysis tools for fine-grained, longitudinal learner data. In *Proc. 1st Int. Conf. Educ. Data Mining*, pages 157–166, Montreal, Canada, 2008.

[58] B. Liu. Web Data Mining - Exploring hyperlinks, contents and usage data. A forthcoming book, 2006/2007.

[59] J. Luan. Data mining and its applications in higher education. *New Directions For Institutional Research*, Spring 2002.

[60] J. MacQueen. Some methods for classifications and analysis of multivariate observations. In *Proc. of the 5th Berkeley Symp. on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[61] H. W. Martin. Transformations between tree permutations and inversion tables. In *Proceeding CSC '90 Proceedings of the 1990 ACM annual conference on Cooperation*, pages 140–146, USA, 1990.

[62] A. Merceron and K. Yacef. Educational data mining: a case study. In *The 12th Conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, 2005.

[63] S. Milgram. The small word poblem. pages 62–67. Psycology Today 1, 1967.

[64] A. Monreale, R. Trasarti, C. Renso, D. Pedreschi, and V. Bogorny. Preserving privacy in semantic-rich trajectories of human mobility. *SPRINGL '10*, 2010.

[65] B. M . E. Moret. Decion trees and diagrams. In *Computing Surveys, 14(4)*, pages 593–623, 1982.

[66] J. Mostow and J. Beck. *Some useful tactics to modify, map and mine data from intelligent tutors*. J. Nat. Lang. Eng. vol.12, no.2, 2006.

[67] S. K . Murthy. Automatic construction of decision trees from data: A multidisciplinary survey. In *Data Mining and Knowledge Discovery, 2(4)*, pages 345–389, 1998.

[68] M. E. J. Newman. The structure and function of complex networks. pages 167–256. SIAM Review, Vol. 45, 2003.

[69] F. Olken and D. Rotem. Random sampling from databases - a survey. In *Statistics & Computing, 5(1)*, pages 25–42, 1995.

[70] J. S. Park, M. S. Chen, and P. S. Yu. *An effective has-based algorithm for mining association rules*. SIGMOD Record, 1995.

[71] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, pages 398–416, Jerusalem, Israel, 1999.

[72] M. Pechenizkiy, N. Trcka, P. De Bra, and P. Toledo. Currim: Curriculum mining. In *The 5th International Conference on Educational Data Mining*, Chania, Greece, 2012.

[73] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proc. 2000 ACM-SIGMOD Int. Workshop Data Mining and Knowledge Discovery (DMKD'00)*, pages 11–20, Dallas, TX, 2000.

[74] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dalay, and M.C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE '01 Proceedings of the 17th International Conference of Data Engineering*, pages 215–223, USA, 2001.

[75] MIT Total Data Quality Management Program. `http://we.mit.edu/tdqm/www/index.shtml`.

[76] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann Publisher, San Mateo, CA, 1993.

[77] C. Romero, J. R. Romero, J. M. Luna, and S. Ventura. Mining rare association rules from e-learning data. In *The 3rd International Conference on Educational Data Mining*, pages 171–180, 2010.

[78] C. Romero and S. Ventura. Educational Data Mining: A Review of the State of the Art. *IEEE Transactions on systems, man and cybernetics*, 40(6):601–618, 2010.

[79] C. Romero, S. Ventura, and P. De Bra. *Knowledge discovery with genetic programming for providing feedback to courseware author*. User Model. User-Adapted Interaction: J. Personalization Res., vol. 14, no. 5, 2004.

[80] C. Romero, S. Ventura, and E. García. Data mining in course management systems: Moodle case study and tutorial. *Computers & Education*, 51(1):368–384, 2008.

[81] S. R . Safavian and D. Landgrebe. A survey of decision tree classifier methodology. In *IEEE Trans. System, Man and Cybernetics, 22*, pages 660–674, 1998.

[82] J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Density based clustering in spatial databases: The algorithm GDBSCAN and its applications. In *Data Mining and Knowledge Discovery, 2(29*, 1998.

[83] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large database. In *Proc. of the 21th Int. Conf. on Very Large Databases (VLDB'95)*, pages 432–444, Zurich, Switzerland, 1995.

[84] A. Savasere, E. Omiecinski, and S. Navathe. Mining for strong negative associations in a large database of customer transactions. In *Proc. of the 14th Intl. Conf. on Data Engineering*, pages 494–502, Orlando, Florida, 1998.

[85] D. W. Scott. *Multivariate Density Estimation*. Wiley, 1992.

[86] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Addison-Wesley, Reading, MA, 1996.

[87] SPAM. `http://himalaya-tools.sourceforge.net/Spam/`.

[88] M. Spiliopoulou. Stream mining in education? Dealing with evolution. In *Proceeding EDM 2012 - 5th International Conference on Educational Data Mining*, Chania, Greece, 2012.

[89] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proc. of the 21st VLDB Conf.*, pages 407–419, Zurich, Switzerland, 1995.

[90] S. S. Stevens. Measurement. in g.m. maranell, editor. In *Scaling: A Sourcebook for Behavioral Scientists*, Chicago. Aldine Publishing Co.

[91] P. N. Tan, V. Kumar, and J. Srivastava. Indirect association: Mining higer order dependencies in data. In *Proc. of the 4th European Conf. of Principles and Practice of Knowledge Discovery in Databases*, pages 632–637, Lyon, France, 2000.

[92] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[93] R. Y. Wang, M. Ziad, Y. W. Lee, and Y. R. Wang. Data quality. In *The Kluwer International Series on Advances in Database Systems, Volume 23*. Kluwer Academic Publisher, 2001.

[94] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition*. Morgan Kaufmann, 2011.

[95] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large databases. In *SDM 2003*, San Francisco, 2003.

[96] C. T. Zahn. *Graph-Theoretical Methods for Detecting and Describing Gestalt Cluster*. IEEE Transaction on Computers, C-20, 1971.

[97] M. J. Zaki and C. J. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proc. 2002 SIAM Int. Conf. Data Mining*, pages 457–473, Arlington, VA, 2002.