Universidade Nova de Lisboa

Faculdade de Ciências e Tecnologia

Departamento de Informática

Mining Protein Structure Data

José Carlos Almeida Santos

Dissertação apresentada para obtenção de Grau de Mestre em Engenharia Informática , perfil de Inteligência Artificial, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientador: Prof. Doutor Pedro Barahona

Co-Orientador: Prof. Doutor Ludwig Krippahl

LISBOA

2006

To my grandmother Lucília and to my girlfriend Gilda

Acknowledgements

The very beginning of this thesis was in September 2004 when I had the first meeting with prof. Pedro Barahona and prof. Ludwig Krippahl where they explained me how promising the area of BioInformatics is. Their idea was to build a database of protein structure data and to try to discover interesting information from it applying data mining techniques.

At the time the curricular part of the master program was about to start and I was also working full time at Novabase Business Intelligence. Fortunately my managers at Novabase were understanbly and allowed me one day per week for the Masters. I thank Fernando Jesus and Vasco Lopes Paulo for that freedom and, more specially, for the professional and personal growth during the year I worked at Novabase.

During that day per week for the masters I had the lectures and ocasionaly meetings with profs. Barahona and Krippahl. A smaller version of the database was already built and some simple mining models were developed.

In the summer of 2005 I did a 3 month internship at Microsoft, in the US. During those 3 months my mind was in totally different projects but I had very fruitful conversations with my manager there, Galen Barbee, that made me see the world in a different perspective and had a very important influence in my decision of pursuing a scientific career. I thank him for that.

There are other professors-friends that did not have any direct relationship with this thesis but whose presence and influence in my education I would like to acknowledge here: Pedro Guerreiro, Artur Miguel Dias and Duarte Brito.

I would also like to acknowledge my two office mates, Marco Correia and Ludwig Krippahl. Marco did his master thesis in the same area and already had some experience with the pdb files, he has helped me retrieving data from the Protein Data Bank. Ludwig, besides the co-supervision of this work and many interesting conversations about a broad variety of subjects, also kindly conceded me the template of his Phd thesis from which the present is inspired.

Finally, I would like to acknowledge professor Pedro Barahona for his supervision of this project. His initial thesis proposal, scientific wisdom and patience with me made this work possible. I also thank him for the funding through REWERSE.

Sumário

O tema principal deste trabalho é a aplicação de técnicas de data mining, em particular de aprendizagem automática, para a descoberta de conhecimento numa base de dados de proteínas.

No primeiro capítulo da tese é feita uma introdução aos conceitos de base. Nomeadamente na secção 1.1 discute-se um pouco a metodologia de um projecto de Data Mining e descrevem-se os seus principais algoritmos. Na secção 1.2 é feita uma introdução às proteínas e aos formatos de ficheiro que lhe dão suporte. Este capítulo é concluído com a secção 1.3 que define o principal problema que pretendemos abordar neste trabalho: determinar se um amino ácido está exposto ou enterrado numa proteína, de forma discreta (i.e.: não contínua), para cinco classes de exposição: 2%, 10%, 20%, 25% e 30%.

No segundo capítulo, seguindo de perto a metodologia CRISP-DM, explica-se todo o processo de construção da base de dados que deu suporte a este trabalho. Nomeadamente, descreve-se o carregamento dos dados do Protein Data Bank, do DSSP e do SCOP. Depois faz-se uma exploração inicial dos dados e é introduzido um modelo simples de previsão (baseline) do nível de exposição de um amino ácido. É também introduzido o Data Mining Table Creator, um programa criado para produzir as tabelas de data mining necessárias a este problema.

No terceiro capítulo analisam-se os resultados obtidos recorrendo a testes de significância estatística. Inicialmente comparam-se os diversos classificadores usados (Redes Neuronais, C5.0, CART e Chaid) e conclui-se que o C5.0 é o mais adequado para o problema em causa. Também se compara a influência de parâmetros como o nível de informação do amino ácido, o tamanho da janela de vizinhança e o tipo de classe SCOP no grau de acerto dos modelos.

O quarto capítulo inicia-se com uma pequena revisão da literatura sobre a acessibilidade relativa de amino ácidos ao solvente. Depois é feito um sumário dos principais resultados atingidos e elicita-se possível trabalho futuro.

O quinto e último capítulo consiste num conjunto de anexos. O anexo A contém, o esquema da base de dados, o anexo B contém tabelas com informações auxiliares e

o anexo C descreve o software presente no DVD que acompanha a tese e que permite reconstruir todo o trabalho.

Palavras chave: Acessibilidade relativa de amino ácido ao solvente, Previsão da Estrutura de Proteínas, Data Mining, BioInformática, Inteligência Artificial

Abstract

The principal topic of this work is the application of data mining techniques, in particular of machine learning, to the discovery of knowledge in a protein database.

In the first chapter a general background is presented. Namely, in section 1.1 we overview the methodology of a Data Mining project and its main algorithms. In section 1.2 an introduction to the proteins and its supporting file formats is outlined. This chapter is concluded with section 1.3 which defines that main problem we pretend to address with this work: determine if an amino acid is exposed or buried in a protein, in a discrete way (i.e.: not continuous), for five exposition levels: 2%, 10%, 20%, 25% and 30%.

In the second chapter, following closely the CRISP-DM methodology, whole the process of construction the database that supported this work is presented. Namely, it is described the process of loading data from the Protein Data Bank, DSSP and SCOP. Then an initial data exploration is performed and a simple prediction model (baseline) of the relative solvent accessibility of an amino acid is introduced. It is also introduced the Data Mining Table Creator, a program developed to produce the data mining tables required for this problem.

In the third chapter the results obtained are analyzed with statistical significance tests. Initially the several used classifiers (Neural Networks, C5.0, CART and Chaid) are compared and it is concluded that C5.0 is the most suitable for the problem at stake. It is also compared the influence of parameters like the amino acid information level, the amino acid window size and the SCOP class type in the accuracy of the predictive models.

The fourth chapter starts with a brief revision of the literature about amino acid relative solvent accessibility. Then, we overview the main results achieved and finally discuss about possible future work.

The fifth and last chapter consists of appendices. Appendix A has the schema of the database that supported this thesis. Appendix B has a set of tables with additional information. Appendix C describes the software provided in the DVD accompanying this thesis that allows the reconstruction of the present work.

Keywords: Amino acid Relative Solvent Accessibility, Protein Structure Prediction, Data Mining, BioInformatics, Artificial Intelligence

Contents

List of Figures

List of Tables

Chapter

1

1 Introduction

In this chapter:

Section 1.1
Data Mining overview

Section 1.2
Protein Background

Section 1.3
Problem History

Chapter organization

In this chapter we aim to provide some basic knowledge that will make the reader familiar with the concepts used throughout the entire thesis.

In section 1.1, in an overview of data mining, we cover its methodologies with a strong focus on CRISP DM Methodology. This methodology is the most widely used for data mining projects and is the one we have used in this thesis. We also explain the main data mining algorithms, namely Neural Networks and the several types of Decision Trees. These will be the classifiers used in the learning phase of this project.

In section 1.2, we give a brief background explanation about proteins and amino acids. Then we introduce the structural classification of proteins, introducing the SCOP database. supporting files. We discuss what is SCOP, a pdb file and the DSSP program. These three concepts will be widely used thoroughly this thesis.

Finally, in section 1.3, we explain the initial motivation for this thesis main problem and the objectives to achieve with this work.

1.1 Data Mining overview

Data mining is a set of techniques to discover patterns, associations or, in general, interesting knowledge from large amounts of data.

In the last ten to twenty years, as the volumes of stored digital data, the memory capabilities and the computing power have grown, also has the need to take advantage of all that potential.

For instance, in several industries like communications or retail distribution (e.g. : supermarkets) there are huge databases of operational data that have plenty of hidden underlying information. The aim of data mining is to uncover that information and provide the decision makers with the knowledge to make better informed decisions.

In an academic environment, as is the case of this thesis, the aim is identical, it is to perform knowledge discovery in a huge database.

1.1.1 Data Mining software

Data Mining algorithms are very specific and hard to configure and use. In order to make the process of Data Mining more productive many tools aroused in the market in the late 90s. Those tools, besides supporting a wide variety of different purpose algorithms, integrate them in a friendly and easy to use environment that allows the user to develop full data mining solutions.

The most widely known commercial data mining software tools are Clementine from SPSS, Enterprise Miner from SAS, Intelligent Miner from IBM and Statistica from StatSoft.

There is also one open source Data Mining solution called Weka, which can be easily extendible – its Java source code is available - but it is not very friendly and has serious performance problems.

Commercial databases like Oracle 10g and SQL Server 2005 also have built-in data mining tools.

More from a programming perspective, there are Matlab and the R language (which is open source), although these two are not exactly Data Mining packages, but programming environments where it is easy to develop mining algorithms. (Some are already implemented)

For this project we have chosen Clementine (version 9.06) because of its overall good quality, ease of use and is the tool the author was more familiar with.

While the following sections are generic to most data mining tools, they are inspired by the author's experience with Clementine.

1.1.2 Data Mining possibilities

There are several ways to achieve the goal of data mining which is to extract new information from existing data. As we will see, there are two approaches to fulfil that goal: supervised learning and unsupervised learning. In the supervised learning approach for each input the desired output is known. In unsupervised learning, the algorithm classifies the input on its own.

1.1.2.1 Classification/Estimation

Both classification and estimation require a training phase where the attribute to predict is learned. The difference between classification and estimation is that the first deals with ordinal values and the latter with continuous values. In classification the output is a class (that already existed in training), in estimation the output is a real number. Sometimes it is interesting to reduce an estimation problem to a classification problem. That is done through binning (there are several binning methods, a simple one is to assign a class to values within a certain range).

An example of a classification algorithm is a decision tree like C5.0. An example of estimation algorithm is a regression model. Several algorithms, like neural

networks or classification and regression trees, can do both classification and estimation.

1.1.2.2 Clustering

Clustering consists in segmenting a population into several different subgroups called clusters. The difference between clustering and classification is that the former does not have any explicit information to which group the records belong as does a classification algorithm.

In clustering, the records are grouped together by a proximity criterion. It is the job of the analyst to determine if the discovered clusters have any underlying meaning. Hence, a cluster model is often used in data exploration phases and rarely an end by itself.

Sometimes a predictive model can be significantly improved by adding a cluster membership attribute or just by applying it to members of the same cluster.

1.1.2.3 Associating Rules

The task of associating rules is to determine what items go together (e.g. what usually goes together in a shopping cart at the supermarket). Associating rules can also be used to identify cross-selling opportunities and to design attractive packages or groupings of products and services.

1.1.2.4 Visualization

Sometimes the purpose of data mining is simply to describe what is going on in a complex database, in a way that increases our understanding of the people, the products, or the processes that produced the data in the first place. A good enough description of a behaviour will often suggest an explanation for it as well, or at least where to start looking for it.

Using some of the above techniques we can create predictive models. Whatever their application might be, the predictive models use experience to *assign scores* and *confidence levels*, to some relevant outcome in the future.

So, one of the keys to success is having enough data with the outcome already known to train the model. Simply stated, there are really two things to do with predictive models:

The first phase is training, where the model is created using data from the past, the second is scoring, where the created model is tested with unseen data to see how it scored.

One should never forget that the most important is to perform well in the unseen data and not in the training data. O*verfitting* is the situation that occurs when the model explains the training data but cannot generalize to test data.

To apply a predictive model we are assuming that part of the data is a good predictor of the remaining data (or in time series that the present is a good predictor of the future). We also assume that the patterns that are observed can be explained, at least partially, by the attributes we are considering.

1.1.3 CRISP-DM Methodology

The main methodology for Data Mining is CRISP-DM (CRoss Industry Standard Process for Data Mining). There are others like SEMMA [1.1-9] (Sample, Explore, Modify, Model, Assess) from SAS, but CRISP-DM is, by far ([1.1-12]), the most widely used. In this subsection we introduce the methodology and its phases.

In 2000, after several years of discussion, a consortium of data mining specialists from industry, and academia created the CRISP-DM methodology to apply to Data Mining projects.

According to the CRISP DM Methodology, the life cycle of a data mining project consists of six phases, with several subtasks at each phase. At the beginning of the project there is precedence between the phases, but as the data mining project evolves, and more insights are gathered from the data, more focused questions can be asked and hence it is often needed to return to the earlier phases.

The six phases of the CRISP-DM methodology are: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Deployment. The first three phases are the most time consuming and roughly 80% of a Data Mining project is spent on them. In the next pages we will discuss what is required at each phase.

1.1.3.1 Business Understanding

This is the first and the most crucial part of the data mining process. We need to identify and understand the problem to be solved. This step requires a good cooperation with someone with business skills. In the case of this thesis, this meant a close cooperation with someone with skills in biology, which was the thesis co-supervisor Ludwig Krippahl.

The most important goal at this phase is to define the business objectives in order to know what answers to seek.

Although in this phase we should not be too worried about what is required to solve the business objectives, after setting each objective we need to take into account if it is reasonable for the data that might be available. Often, the data available is not enough to predict what we want, whether because there are not enough observations or, more commonly, because we lack important attributes.

1.1.3.2 Data Understanding

The data understanding phase starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data or to detect interesting subsets to form hypotheses for hidden information.

Collecting data

Collecting the relevant data can be an easy task or a daunting one. Often there is some data available but scattered in many different places and formats. Collecting usually consists in gathering the data into a common place and in a common format: typically a relational database.

Describe data collected

One should describe the data which has been acquired, including: the format of the data, the quantity of data, for example number of records and fields in each table, the identities of the fields and any other surface features of the data which have been discovered. Does the data acquired satisfy the relevant requirements?

Ensure data quality

At this point it is important to answer questions like: *Are all the fields populated? Will missing values be a problem? Are the field values legal? Are numeric fields within proper bounds and are code fields all valid? Are the field values reasonable? Is the distribution of individual fields as expected?*

It is important to note that the outcome of data mining depends critically on the data, and data inaccuracies creep in from many different places.

Explore data

It is very important to explore and see some of the data properties before applying data mining algorithms. Some simple SQL queries and graphical visualization can be very helpful here. Typically, these queries will consist of simple statistical analysis and aggregations.

Visualization techniques are also very important at this stage. To check frequency of values and correlations between variables Histograms and Web graphs are particularly advised.

At the end of the data understanding phase we must be confident that the data acquired satisfy the requirements to fulfil the business objectives.

1.1.3.3 Data Preparation

The data preparation phase covers all activities to construct the final dataset (data that will be fed into the modeling tool(s)) from the initial raw data. Data preparation tasks are likely to be performed multiple times and not in any compulsory order. Tasks include table, record and attribute selection as well as transformation and cleaning of data for modeling tools.

Joining tables

Data mining algorithms expect as input a single table. Our databases are often normalized and hence information is scattered through, potentially, many tables. They need to be joined and treated so that they can serve as input for a mining algorithm.

Derive attributes

Often there is a need to create derived attributes. These attributes are created using information from other attributes. For instance in a DB of counties where we have the population and the area of the county we may want to introduce a new derived attribute, Density defined as Population/Area. This new attribute can help significantly the classification algorithms since these might not be able to relate the two fields and the new attribute introduces a concept that was not captured before.

Attribute and row selection

The data set may have many more variables than the mining algorithms can handle (or could handle but would be much slower). In these situations a pre-processing step where only the most relevant attributes are selected is very important. The choice of the most relevant attributes can be done by a human specialist in the business or automatically by specialized algorithms (such as Principal Components Analysis).

In addition, the data set may contain much more records than needed to build a model. The records with least quality (more incomplete if that has no special meaning for the problem) should be left out. If records have a date/time attribute often we want to discard the ones farthest from the period under consideration.

Balance data

Often there are imbalances in the data. For example, suppose that a data set has three classes: *low*, *medium* and *high*, occurring 10%, 20% and 70% respectively.

Many modeling techniques have trouble with such biased data because they will tend to learn only the commoner class and ignore the infrequent ones.

The solution is to balance the data so all classes have approximately the same number of observations. There are two ways to achieve this purpose: One option is to randomly eliminate observations from the larger classes, the other is to duplicate observations from the smaller class.

If the data is well-balanced models will have a better chance of finding patterns that distinguish the classes.

Dividing data for training and for test

Only after the above explained data preparation steps should the records be divided. Literature [1.1-7] says that if plenty of data is available the ideal division should be in 3 sets: A training set to build the model, a test set to further refine it (make sure it is sufficiently generalized) and a validation set that is only used in the end for evaluation purposes. If data is not sufficient only 2 sets should be used: A training set to build the model and a test set to evaluate the model generated. (see Holdout cross-validation). In certain cases, where data is scarce the training should be done with K-fold cross-validation or even Leave-one-out cross-validation.

According to [1.1-10] these are the main validation methods:

Holdout cross-validation

This is simplest type of cross-validation. Observations are chosen randomly from the initial sample to form the validation data, and the remaining observations are retained as the training data. Normally, about 2/3 of the initial sample is used for training and 1/3 for the validation.

K-fold cross-validation

In K-fold cross-validation, the original sample is partitioned into K sub-samples. Of the K sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining K−1 sub-samples are used as training data. The cross-validation process is then repeated K times (the folds), with each of the K sub-samples used exactly once as the validation data. The K results are then averaged to produce a single estimation.

Leave-one-out cross-validation

Leave-one-out cross validation uses a single observation from the original sample as the validation data and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the validation data. Note that this is the same as K-fold cross-validation where K is equal to the number of observations in the original sample.

Clementine has an easy to use partition which enables any classifier to do holdout cross validation easily. In addition, C5.0 supports K-fold cross validation directly.

1.1.3.4 Modeling

Before building a final model one needs to define how to test the quality of the generated model. For classification problems the measure of quality might be the percentage of correct predictions (giving more or less importance to the false positives and wrong negatives for the current problem context).

Only after assessing this preliminary model, one should focus on choosing the exact algorithm to use. For instance, for classification it may be C5.0, Neural Networks, for clustering it may be K-Means or Kohonen maps, for associative rules it may be Carma or GRI.

The decision for a specific algorithm depends on the problem at stake. Some characteristics to compare include robustness to null values, space complexity, time complexity and human readability of model.

There is much room for tuning on this phase since many algorithms require several parameters that can be calibrated to values more suited for the current data. In addition, it is often found that the data available is not enough and should be enriched and hence return to the data preparation phase.

1.1.3.5 Evaluation

At this stage one needs to evaluate the model(s) generated. When the goal is Predictive Modeling it is more or less straightforward to evaluate the quality of the model. The score we get from running the model in the test set is the indicator (the

test set was built in the data preparation phase). Note that a 100% success classifier is a utopia and often the objectives can be achieved with much more modest results. It is up to the business specialist to evaluate the quality of the model and determine if the results are within expectations.

It is also important to analyze if the model behave much worst in test data than in training data and if so to consider if the requisites of Predictive Modeling stand for the data considered.

It is very frequent that the new insight gained changes the understanding of what can be done and gives new ideas for new problems so it is frequent to return to the previous phases.

1.1.3.6 Deployment

A data mining project does not end when a final model is created. The application of the model, or of the knowledge gained from it, is what makes the data mining project worthwhile. The question "How can this information be useful to us?" needs to be answered and its answer incorporated in the decision process. A simple scenario, for predictive models, is its application to new data where we do not know the answer and hope the model give us an answer as accurate as it did in the test or validation sets.

1.1.4 Mining Algorithms

In this section we are going to highlight the objectives and main pros and cons for some of the most widely known data mining algorithms. For a more complete reference and in-depth knowledge of the specifics of each algorithm a good starting point is [1.1-7], [1.1-8] and chapter 3 of [1.1-4].

1.1.4.1 Neural networks

There are two main types of neural networks: Feed-forward and feedback (or recurrent). The difference between a feed-forward and a recurrent neural network is that in the feed-forward the direction of the signals is unidirectional from the input

to the output. In recurrent networks the output goes back to the input. (This is more powerful but with potential convergence problems too).

In this section we are going to briefly explain the feed-forward neural networks which are simpler and more common. Most of the explanation and characteristics, however, are valid for both types. In [1.1-14] there is an excellent explanation of both types and the main variations.

1.1.4.2 Concept

A neural network is composed of a potentially large number of neurons arranged in three different conceptual layers: an input layer representing the input variables, one or more hidden layers, and an output layer representing the output variables.

In a feed-forward neural network the connection between neurons only occurs with neurons in different layers (or in different sub-layers of the hidden layer). Each such connection has an associated weight.

A neuron is only responsible for determining the activation level and firing an output. Its activation level is the sum of the activation levels of the neurons connected to it from the previous layer weighted by the connection strength. The output is a function of the activation level. Typically the logistic function is used: $f(x) = 1 / (1 + e^x)$, where x is the activation level. Let us call the output of a neuron *i*, *Oi*.

Initially connection weights are set to random values and the aim of training a neural network is to determine the best weights for the neuron connections so that the sum of the squares of the errors is minimized. During the training process, as the weights are being adjusted, the prediction errors decrease and eventually the weights on the network no longer change significantly and so the error stabilizes.

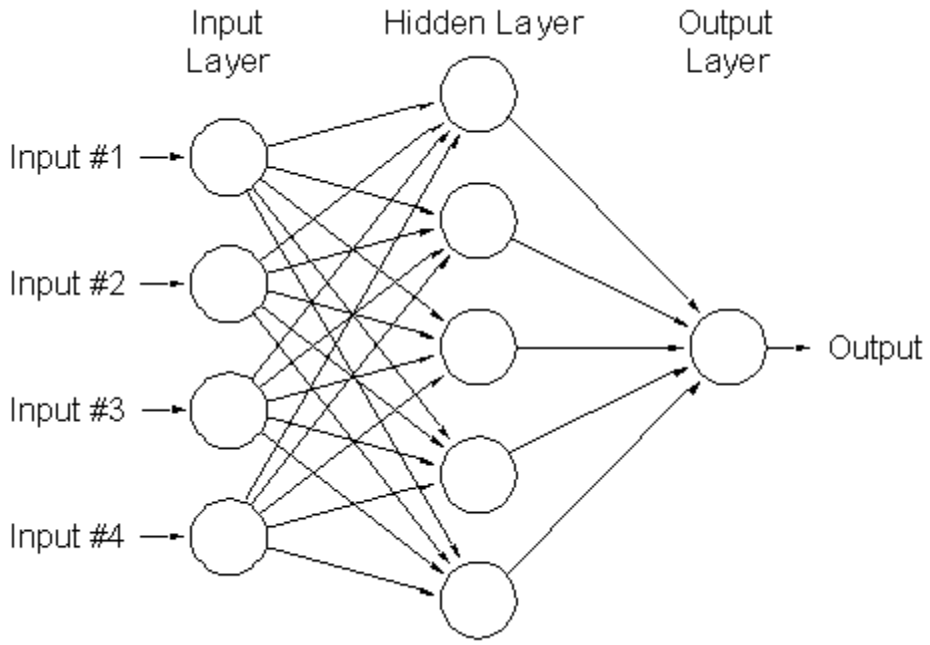


Figure 1.1-1 Architecture of a feed-forward Neural Network with a single hidden layer (figure taken from [1.1-3])

Note that a neural network with just a single hidden layer has severe learning limitations. The classical example is the incapacity of learning the logical Xor function. (This is easily solved by adding a second hidden layer)

A prediction is made when an input passes through the several layers as shown in Figure 1.1-1. The prediction is compared against the expected result and the error is computed. The error is the difference between the known value and predicted value. For situations where the target attribute is categorical a numeric encoding is used. (See *Numeric Coding of Symbolic Fields* in chapter 1 of [1.1-7])

The error assessed is back propagated to determine the responsibility of each neuron in the global error (see *Backpropagation calculations* in chapter 1 of [1.1-7]). Let us denote this error by *Ei*. The rule to update connection's weight takes *Ei* into account. This rule is also called the learning function of the neural network.

A learning function computes a delta to add to the connection weight between neurons *i* and *j* basically by doing the product of the learning rate by *Ej* and *Oi*. (This means going to direction that corrects the error at learning rate's pace). The most used learning function is the delta rule [1.1-15].

The learning rate is a global neural network parameter that defines the rate at which the neural networks adapts to new weights. If this value is too small the network converges very slowly, if the value is too big the weights may jump between extremes and never converge.

The default stop criteria is the persistence which checks if the network has not improve prediction for K (200 by default in Clementine) cycles then the training phase is terminated. Other stop criteria area: time elapsed, percentage of error obtained (may never be reached!) and number of records used.

The main advantages of a neural network are its robustness to data noise, its capacity of running in parallel as well as its capacity of approximation any function.

The more serious disadvantage of a neural network is that the model (eg: the weights of the neuron connections) is incomprehensible for a human and no business knowledge can be extracted from it.

Neural networks are, as presented, a supervised learning algorithm. However, Teuvo Kohonen proposed the Self Organizing Maps (SOM), which is a special type of neural network to perform clustering. (Unsupervised learning)

Clementine supports both SOM, Feed-forward and Recurrent neural networks.

1.1.4.3 Clustering Algorithms

Clustering models focus on identifying groups of similar records and labelling the records according to the group to which they belong. This is done without the benefit of prior knowledge about the groups and their characteristics. In fact, one may not even know exactly how many groups to look for. This is what distinguishes clustering models from the other machine-learning techniques - there is no predefined output or target field for the model to predict.

These models are often referred to as unsupervised learning models, since there is no external standard by which to judge the model's classification performance. There are no right or wrong answers for these models. Their value is determined by

their ability to capture interesting groupings in the data and provide useful descriptions of those groupings.

Clustering methods are based on measuring distances between records and between clusters. Records are assigned to clusters in a way that tend to minimize the distance between records belonging to the same cluster.

One of the most known clustering algorithms is K-Means, which works by defining a fixed number of clusters and iteratively assigning records to clusters and adjusting the cluster centers. This process of reassignment and cluster centre adjustment continues until further refinement can no longer improve the model significantly.

Besides K-Means, Clementine also includes SOM and TwoStep clustering.

1.1.4.4 Decision Trees

Decision trees are a very popular family of classification algorithms. There are 2 main types of decision trees: classification trees and regression trees.

The difference is that the first assigns the records a class (categorical value), while the latter estimates the value of a numeric variable.

Building the decision tree

A simple process for building a decision tree is a recursive greedy algorithm receiving as a parameter a dataset to analyze and returning a tree.

At each step we choose a split attribute (e.g.: outlook) and for each of the attribute's values (e.g.: sunny/overcast/rain) we call the algorithm recursively having as parameter the dataset where this attribute has always the same value (see Figure 1.1-2). In this example this would return 3 sub-trees- one for each state of the variable- that would be leaves of the previous node.

To minimize the *over fitting* of the generated tree a pruning stage is often needed. There are two types of pruning: pre-pruning and post-pruning. In pre-pruning, while the tree is being built, the build algorithm determines if the node should be

further expanded or become a leaf (Based on threshold of correct predictions at that level). In post-pruning the bottom splits that do not contribute significantly to the accuracy of the tree are condensed.

Whatever pruning technique used the result will be a simpler and more generalized tree that is easier to interpret. The performance will be worse in the training data but it is more likely it will perform better than the original tree in test data.

The central point in building a decision tree is the choice of the split attribute. There are several split criteria as shown in Table 1.1-1. The simplest split criteria is as this: Select an attribute, *Ai*, (to simplify lets consider all attributes and target are binary) and divide the data in 2 sets: one where *Ai* is 0 and other where *Ai* is 1. For first set let y0 be the most common value of the target value in it. For second set define y1 the same way. The number of errors by choosing *Ai* as the split attribute is then the number of examples where the target value is different from y0 and y1 in the respective sets. The selected attribute at each step is the one that minimizes this error.

The described split criteria is very simple and doesn't give very good results. Other split criteria are shown in Table 1.1-1. To know how they work please read chapter 3 of [1.1-7] and section 3.2.1 of [1.1-4].

As with neural networks there are many parameters that affect the result of a decision tree and often we need to play with them to see how they influence accuracy. In Clementine we can choose among several splitting criteria and pruning rules and even control parameters such as minimum number of observations for a split, minimum number of observations in a leaf and maximum tree depth.

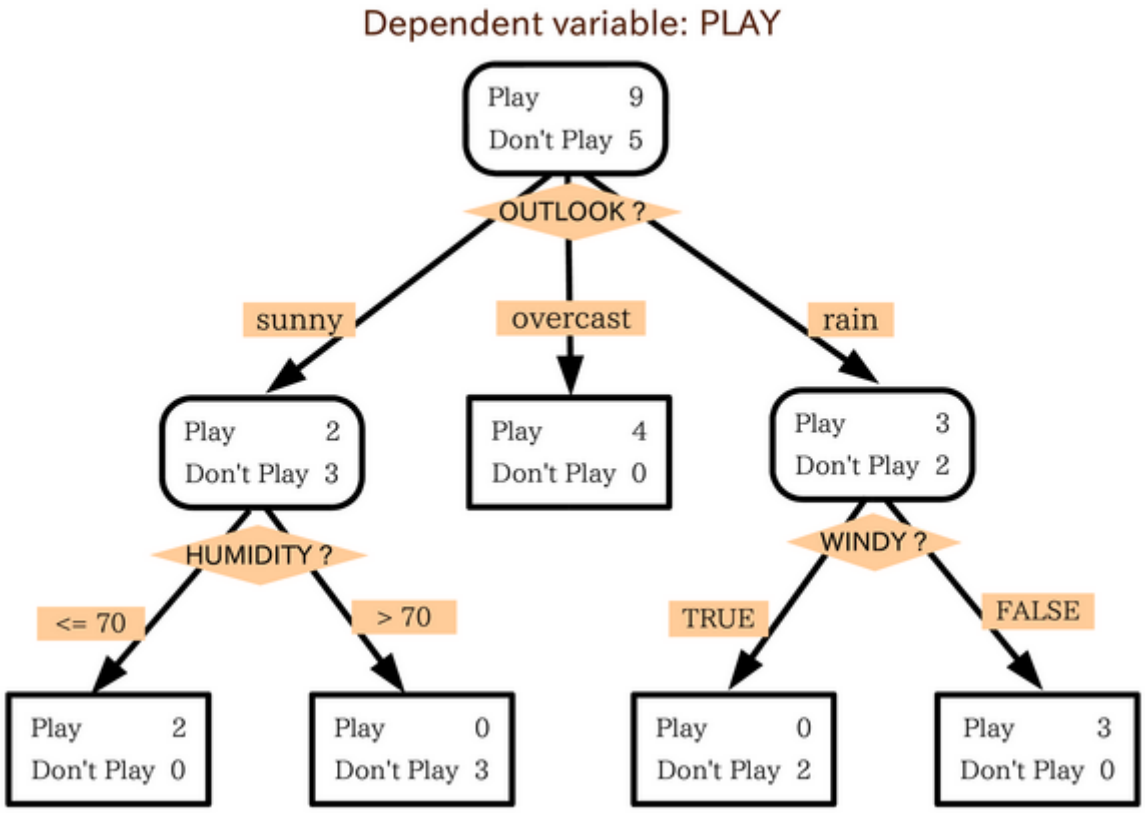


Figure 1.1-2 Sample decision tree (taken from [1.1-11])

Understanding a decision tree is straightforward. At each node, starting by the root node, some attribute is being tested. If it is of a certain value we follow a certain branch of the tree, if not we follow other branch. This is done recursively until a leaf node is reached. At a leaf node a certain prediction is made. One can easily rewrite a decision tree as a set of if-then rules to improve its readability.

The main advantage of a decision tree is precisely its readability (in opposition to a neural network where a human cannot understand the reasoning behind a prediction).

Clementine supports C5.0 (which is a classification tree), C&RT (Classification and regression tree), CHAID and Quest. The following table illustrates the differences between the different decision tree algorithms:

Algorithm	Split criteria	Leaves per node	Predictor types	Target types
Quest	Statistical tests	2	Categorical/Numeric	Categorical
CHAID	Chi Squared test	>=2	Categorical/Numeric	Categorical/Numeric
C5.0	Information gain	>=2	Categorical/Numeric	Categorical
C&RT	Several impurity measures	2	Categorical/Numeric	Categorical/Numeric

Table 1.1-1 Decision tree algorithm comparison

Improving accuracy through boosting

Boosting consists of building several models -the number of models to build is a parameter of the boost mode- each one specialized in the records the previous model failed to classify correctly. The whole set of models is the final model and a voting system is used to decide the final prediction. Boosting frequently improves the accuracy but takes much longer to train.

The only classifier in Clementine 9 to support directly this boost mode is C 5.0.

1.2 Protein Background

Proteins are organic molecules composed of subunits called amino acids. There are twenty different amino acids types available in proteins (list in Appendix B, AII. 1). Amino acids are connected to make proteins by a chemical reaction in which a molecule of water (H_2O) is removed, leaving two amino acids residues (i.e. the part of the amino acid that is left when the water molecule is removed) connected by a peptide bond. Connecting multiple amino acids in this way produces a polypeptide [1.2-7]. A protein might be composed of several polypeptides (i.e. chains).

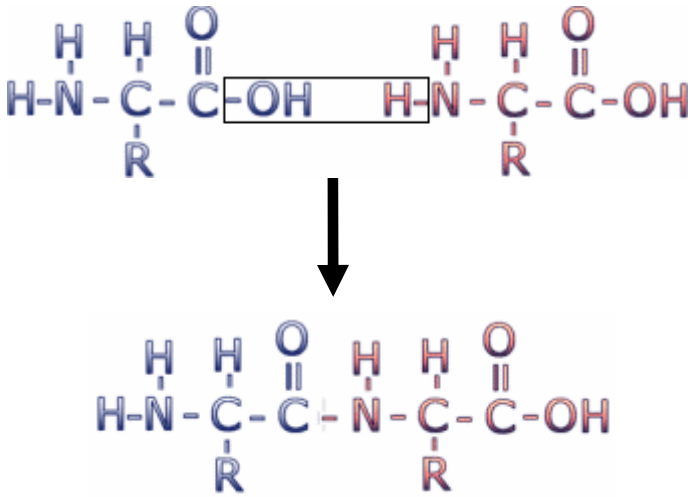


Figure 1.2-1 Process of connecting amino acids (picture adapted from [1.2-7])

Figure 1.2-1 shows the process of connecting amino acids as has been described above. As we will see during this thesis, amino acids are mainly divided in two categories, hydrophilic (i.e.: tend to be in contact with the solvent) and hydrophobic (i.e.: avoid being in contact with the solvent).

Figure 1.2-2 shows the image of the 1n2o protein as rendered by KiNG, one of the molecule viewer applets available at the RSCB (www.rcsb.org) site.

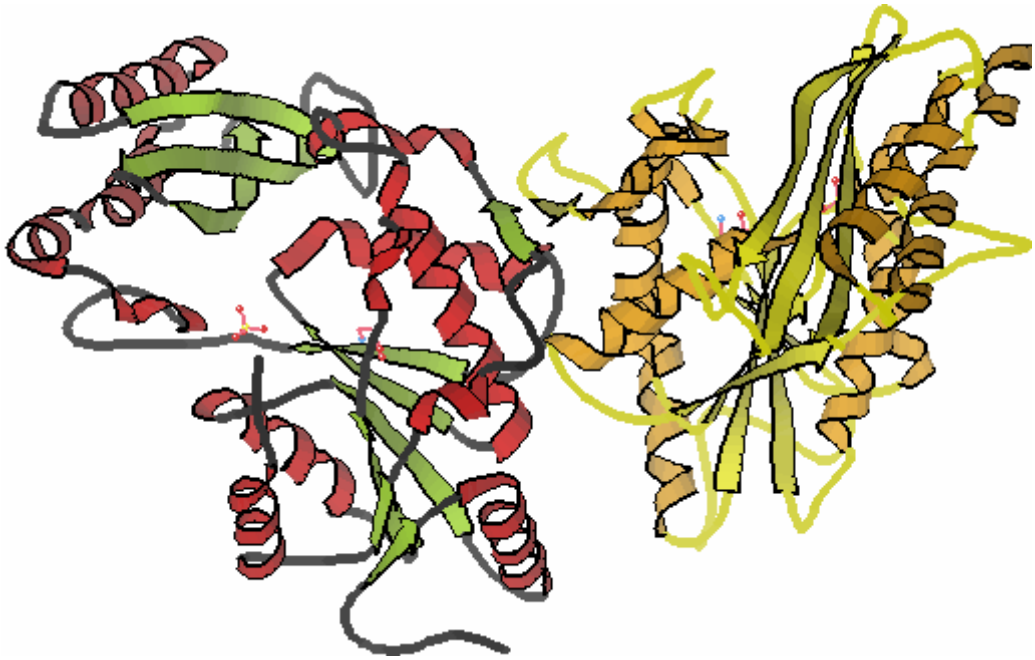


Figure 1.2-2 Image of protein with pdb_id 1n2o

Proteins are an essential part of any living organism and they participate in every process of a cell, many of them being enzymes catalyzing chemical reactions.

The amino acid sequence (simply a string of single letters) is called the primary structure of a protein and uniquely determines the folding of the protein. The final folded shape of a protein (its tertiary structure) is strongly correlated with the protein function in the organism.

One of the major open problems in biology is precisely how to determine the folding of a protein given its amino acid sequence. The solution for this problem will cause a revolution in biology and medicine as it make possible to develop custom assembled proteins to achieve certain functions.

The secondary structure of a protein consists in the repeating patterns formed by groups of amino acids (see subsection Definition of Secondary Structure of Proteins). The tertiary structure, as we have already mentioned, is the final folded shape of a protein, with the knowledge of the 3D position of each amino acid. The quaternary structure is the shape that results from the interaction of more than one protein molecule [1.2-9].

For simplicity, during this thesis, we will often use the term amino acid when technically we should have used the term amino acid residue.

1.2.1 Methods of Structural Classification of Proteins

Proteins have structural similarities with other proteins. That fact created the necessity of developing methods to classify proteins according to their structure.

There are three main structural classification techniques (SCOP, CATH and Dali). In [1.2-6] an exhaustive comparison between the three is made:

- SCOP (Structural Classification of Proteins) is manually maintained by a group of experts.
- CATH (Class Architecture Topology Homologous) uses a combination of manual and automated methods to define and classify domains.
- Dali (Distance mAtrix aLIgnement) is a fully automated method to define and classify classifying domains.

It is known that automatic assignment algorithms cannot identify all structural and evolutionary relationship between proteins. In this work, we have opted for SCOP because it is considered the more reliable classification method and the “de facto” standard. The release of SCOP used on this thesis was 1.67 from February 2005.

1.2.2 About SCOP

SCOP aims to provide a detailed and comprehensive description of the structural and evolutionary relationships between all proteins whose structure is known. It is updated with irregular frequency and hence is not always up to date with the Protein Data Bank. It is made available in simple text files. Its parsing into our database is described in subsection 2.1.3.

Protein chains in SCOP appear as leafs of a tree with four levels. The levels of the hierarchy are, in this order: class, fold, super family and family. These hierarchy levels pretend to reflect the structural and evolutionary similarities between the proteins.

As quoted from [1.2-10], the major levels in the SCOP hierarchy are:

- class - a very broad description of the structural content of the protein.
- fold - broad structural similarity but with no evidence of a homologous relationship.
- superfamily - sufficient structural similarity to infer a divergent evolutionary relationship but no detectable sequence similarity.
- family - significant sequence similarity.

There are 11 SCOP classes: a, b, c, d, e, f, g, h, i, j, k and l. Each class describes the major structural characteristics of its proteins. Below is the characterization of each of the 11 classes, as quoted from [1.2-10]:

a) proteins with only α-helices
b) proteins with only β-sheets
c) proteins with both α-helices and mainly parallel β-sheets (as beta-alpha-beta units)
d) proteins with both α-helices and mainly anti parallel β-sheets (as separate alpha and beta domains)
e) multi domain proteins
f) membrane and cell surface proteins and peptides (not including those involved in the immune system)
g) small proteins with well-defined structure
h) coiled-coil proteins
i) low-resolution protein structures
j) peptides and fragments
k) designed proteins of non-natural sequence

1.2.3 PDB Files Format

PDB is one of the most used formats to store molecule structure related data. It's a semi structured text file and its format is described in [1.2-5]. A PDB file specifies the position in space of every atom of every amino acid of a given molecule. A protein is a special case of molecule that we are interested in this project.

Figure 1.2-3 shows the beginning of a pdb file.

```
HEADER    HYDROLASE(ACID PROTEINASE)              14-FEB-94   XXXX
TITLE     RATIONAL DESIGN OF POTENT, BIOAVAILABLE, NONPEPTIDE CYCLIC
TITLE    2 UREAS AS HIV PROTEASE INHIBITORS
COMPND    HUMAN IMMUNODEFICIENCY VIRUS TYPE 1 (HIV-1) PROTEASE
COMPND   2 COMPLEXED WITH XK263 OF DUPONT MERCK
KEYWDS    HYDROLASE(ACID PROTEINASE)
EXPDTA    X-RAY DIFFRACTION
AUTHOR    C.-H.CHANG
```

Figure 1.2-3 Excerpt of the header of 1hvr pdb file

The first rows of a pdb file describe generic information about a protein. The first column of a pdb file describes the type of the row that follows. For instance, the *HEADER* row contains the name of the molecule and the date it was deposited at the Protein Data Bank and the *EXPDTA* row contains the experimental technique used to obtain the protein.

The majority of information in a pdb file is the description of the 3D coordinates of each atom of the molecule (the *ATOM* rows). Figure 1.2-4 shows an excerpt of it.

```
ATOM     22  N   VAL A   3     -13.253  35.366  26.117  1.00 35.97
ATOM     23  CA  VAL A   3     -12.269  34.588  25.374  1.00 34.64
ATOM     24  C   VAL A   3     -13.029  33.962  24.196  1.00 32.45
ATOM     25  O   VAL A   3     -14.003  33.234  24.394  1.00 31.41
ATOM     26  CB  VAL A   3     -11.730  33.474  26.320  1.00 35.21
ATOM     27  CG1 VAL A   3     -10.597  32.691  25.660  1.00 36.15
ATOM     28  CG2 VAL A   3     -11.293  33.952  27.723  1.00 34.78
ATOM     29  H   VAL A   3     -14.058  34.893  26.477  0.00 15.00
ATOM     30  N   THR A   4     -12.612  34.301  22.976  1.00 29.04
ATOM     31  CA  THR A   4     -13.127  33.558  21.845  1.00 25.02
ATOM     32  C   THR A   4     -12.159  32.418  21.669  1.00 25.43
ATOM     33  O   THR A   4     -11.038  32.479  22.141  1.00 27.66
ATOM     34  CB  THR A   4     -13.237  34.425  20.555  1.00 26.90
ATOM     35  OG1 THR A   4     -11.953  34.894  20.130  1.00 27.46
ATOM     36  CG2 THR A   4     -14.202  35.617  20.673  1.00 25.56
ATOM     37  H   THR A   4     -11.801  34.863  22.831  0.00 15.00
ATOM     38  HG1 THR A   4     -12.071  35.428  19.325  0.00 15.00
```

Figure 1.2-4 Excerpt of the body of 1hvr pdb file

The second column in an *ATOM* row is the atom number, the third column the atom description, the fourth column the three letter amino acid name, the fifth column the chain letter and the sixth column the amino acid number (not necessarily sequential).

Sometimes a letter follows the amino acid number, which we refer as the *CodeResid* column. It is this column in conjunction with the amino acid number that uniquely identify an amino acid inside a pdb file because, although rare, in a pdb file two amino acids may have the same amino acid number but different *CodeResid*s.

The seventh, eighth and ninth columns are the *X*, *Y* and *Z* coordinates of the atom inside the molecule (in angstroms). The last two columns are the occupancy and temperature factor.

In early 2005, when we started the loading process of the pdb files, there was already a beta version of this format in XML. The reason we have choose the txt format was because, despite its problems, it still was better documented. Besides, it occupies much less disk space and is much faster to parse and load.

A pdb file deposited in the Protein Data Bank generally contains an asymmetric unit. An asymmetric unit is the smallest portion of a crystal structure to which

crystallographic symmetry operations (e.g. : rotations, translations) can be applied to generate a unit cell. The unit cell is the component that is stacked multiple times to generate the entire crystal.

The pdb files we will use during this thesis are biological units (the macromolecules that are believed to be functional). A biological unit (biounit) pdb file is derived from the original asymmetric unit and might be part of the asymmetric unit, the whole asymmetric unit or several asymmetric units. A complete explanation of biological and asymmetric units can be found in [1.2-11].

In sub section 2.1.2 we will thoroughly explain the loading process of the pdb files into our database.

1.2.4 Definition of Secondary Structure of Proteins

DSSP stands for Definition of Secondary Structure of Proteins. It is a widely used program to calculate, among other things, the secondary structure (e.g. : alfa helix, beta sheets) of a pdb file. It is available for several platforms from [1.2-1] and is free for academic purposes.

DSSP receives as input a pdb file and outputs a dssp file with information about each amino acid. For each amino acid the DSSP program computes its accessible solvent area (the value we are most interested in this thesis), the secondary structure element to which it belongs, the angles the amino acid forms with its predecessor and successor.

Figure 1.2-5 shows an excerpt of the DSSP classification for the *1hvr* pdb.

#	RESIDUE	AA	STRUCTURE	ACC	KAPPA	ALPHA	PHI	PSI	X-CA	Y-CA	Z-CA
1	1	A	P	87	360.0	360.0	360.0	173.8	-12.7	39.1	29.8
2	2	A	Q E -A	126	360.0	-158.1	-106.1	137.2	-14.3	37.3	27.0
3	3	A	V E -A	16	1.0	-157.4	-117.2	119.2	-12.3	34.6	25.4
4	4	A	T E > -A	59	22.5	-126.5	-89.8	159.4	-13.1	33.6	21.8
5	5	A	L T 3 S+	1	80.0	104.1	-107.5	33.9	-11.9	30.1	21.0
6	6	A	W T 3 S+	182	89.7	41.0	-81.4	-17.1	-9.9	30.5	17.9
7	7	A	Q S < S-	86	110.1	-84.8	-117.1	157.3	-6.8	30.2	20.1
8	8	A	R -	109	48.3	-118.6	-58.0	131.9	-6.2	27.8	22.9
9	9	A	P +	1	51.5	168.0	-76.2	86.9	-7.8	29.6	25.9
10	10	A	L E -D	74	11.3	-177.5	-101.9	145.5	-4.6	29.9	27.9
11	11	A	V E -D	17	29.0	-107.6	-134.1	158.4	-4.2	32.0	30.9
12	12	A	T E -D	93	37.8	-176.5	-81.6	133.9	-1.4	33.0	33.3

Figure 1.2-5 Excerpt of DSSP classification for 1hvr pdb

In order for the figure to fit the page width, we have eliminated some columns that we do not use in our work. For a complete description of the DSSP output file format please read [1.2-2].

The first column of the DSSP file is a sequential amino acid number, the second is the pdb amino acid number, possible followed by a CodeResid letter. The third column is the chain letter and the fourth column the single letter amino acid code. The fifth column is also a single letter representing the motif to which the amino acid belongs. The most common motifs are H: Alfa Helix (31.8% of all amino acids) and E: Extended Strand in a Beta Ladder (21.8%).

Column *ACC* contains the amino acid exposed area in Angstroms^2, and the KAPPA, ALPHA, PHI and PSI are bond and torsion angles with the neighbouring amino acids. The X-CA, Y-CA and Z-CA columns are simply an output of the coordinates of the alpha carbon atom as found in the pdb file.

In sub section 2.1.2 we will explain how, in conjunction with the pdb files, DSSP information was loaded into our database.

1.3 Problem History

The main problem solved in this thesis is the prediction of residue solvent accessibility. This is an interesting problem by itself that can aid in understanding the 3D protein structure.

The initial motivation to tackle this problem was due to the work of other researcher from our group, Marco Correia. In his Master thesis ([1.3-1]) he tested some heuristics to improve the PSICO (Processing Structural Information with Constraint Programming and Optimization) algorithm [1.3-2]. In an important part of his heuristics, the algorithm needs to decide which half of an atom domain should be excluded.

The initial idea behind this thesis was to develop a better heuristic for the placement of atoms inside a protein since the original heuristics only considered geometrical features, not taking into account bio-chemical properties of the atoms or amino acids. Because treating information at the atom level is much more complex than at the amino acid level, we simplified the problem by assuming that all atoms of the same amino acid would have the same burial status of its amino acid.

The burial status of an amino acid is defined as the ratio between its exposed area to the solvent (depends on the protein conformation) and its total area (a constant).

In the literature this problem is called Relative Solvent Accessibility (RSA) prediction. The common values used are twenty and twenty five percent since, from the biochemical point of view, that is the lower bound for chemical interaction.

For our purposes, the most interesting cut off point is only two percent, because we just want to consider an amino acid buried if it has almost no (i.e.: less than 2%) surface exposed to the solvent. We have also defined the 10, 20, 25 and 30 percent

threshold for comparison with the literature (although, as we will later see why, the results are not directly comparable).

The objectives for this thesis are: 1) Build a relational database of protein structure data, easily updated, 2) Some insight in Protein Structure data and 3) Models to predict the buried status of amino acids.

Chapter

2

2 Development of a Protein Database

In this chapter:

Chapter Organization

In this chapter we describe the creation of a protein database that contains all the relevant information to solve the problems proposed before. The database, from now on called *ProteinsDB*, was developed in SQL Server 2005 Developer Edition running over Windows XP Service Pack 2.

The following sections explain the process of building *ProteinsDB* and the preparation for the data mining phase. We follow the CRISP-DM Methodology described in 1.1.3.

Previous section 1.3, Problem History, roughly corresponds to the first phase of CRISP-DM, the Business Understanding, where the goals for this mining project are described.

Sections 2.1 (Loading PDBs), 2.2 (Cleaning data) and 2.3 (Exploring the database) correspond to the second phase of CRISP, the Data Understanding, where collecting the data, ensurance of data quality and the first insights on the data are performed.

Section 2.4, Preparing data, corresponds to the third phase, Data Preparation, where the requirements and construction of the datasets for mining is described. A specific program with the sole purpose of building these datasets was developed to meet the requirements.

In section 2.5, Mining with Clementine, a description of the data mining stream, starting from the data mining table generated in the previous phase and ending with several models for predicting the burial status of an amino acid is introduced. This roughly corresponds to the fourth phase of CRISP, Modeling.

Finally, in section 2.5, Evaluating Mining Results, we show how to evaluate the results generated from the previous phase and introduce a baseline for which our results should be compared. This corresponds to the sixth phase of the CRISP-DM Methodology, Evaluation.

The seventh and last phase of the CRISP-DM Methodology, Deployment, which is the application of the knowledge gained during the mining process, is not the aim of this thesis. Nevertheless, in Future work, we suggest some applications.

2.1 Loading data

Loading data into a database is often one of the most time consuming issues, mainly because it is rare that the data might be added promptly to the database without any transformation. This project was no exception and the phase of data loading was particularly time consuming due the amount of data to load and because the not so friendly format of some data sources. There are four main data sources: Tables with amino acid information, PDB files, the DSSP program and the SCOP 1.67 table.

The following sections describe the process of loading those several sources.

2.1.1 Loading amino acid information

As shown before, proteins are composed of sequences of amino acids, and there are twenty different amino acids types. Each amino acid type is described, more or less accurately, via a number of different features. For the purpose of this work we have used two amino acid information sources.

The first source, [2.1-1], is very simple and has only six properties for each amino acid (Hydropathy, Polarity, H_Donor, H_Acceptor, Aromaticity, Charge at Ph7). This data was loaded to table *Aminoacids*. We will refer to these features of amino acids as information *Simple*.

The second, [2.1-2], is the most complete we found and has 37 properties (full list in Appendix A, Figure A-8) for each amino acid. This data was loaded to table *Aminoacids_Complete*. We will refer to these features of amino acids as information *Complete*.

Another information level exists which consists only on the amino acid name and its total area. We will refer to it as information *Minimal*.

Recall that the main aim of this thesis is to predict the Residue Solvent Accessibility (RSA) exposure state. RSA is simply the ratio between Residue Exposed Area (REA) and Residue Total Area (RTA). REA varies with the position of the amino acid inside the protein (and is one of the outputs of DSSP) and RTA is roughly constant for each of the twenty amino acids types.

To calculate the residue total area (RTA) of an amino acid, most papers in the literature use their surface areas available in several tables. However, each table has its own values. For instance, in papers [4.1-2], [4.1-4] and [4.1-1], RTA values are taken from experimental data published on the 1970s or 1980s (Shrake and Rupley(1973) [2.1-3], Chotia(1976) [2.1-4],and Rose et al(1985) [2.1-5] respectively).

We have opted not to use any of those tables and calculate the RTA using also DSSP because all papers consensually calculate REA with DSSP. We calculate the RTA values by using DSSP in a special way. As said before DSSP gives the exposed area of an amino acid inside a protein (which is given in a PDB file). We developed a program to parse a pdb file and, for each amino acid present, isolate it to a new pseudo pdb file. Then, by running DSSP on those new small pseudo PDB files, we have, among other things, the exposed area to the solvent of the selected amino acids (the value is not constant because the amino acid might be taken from different conformations). This computation is performed for many amino acids taken from about two hundred randomly selected pdb files. The results are then averaged to find the solvent accessibility area for each amino acid type.

The exposed area to the solvent should be very close to the total residue area because the protein now is only the amino acid and so its area is totally exposed to the solvent. However, there are discrepancies with the values of the literature tables because the values on those tables are calculated considering the residue surface area and DSSP considers the residue accessible area to solvent (water). The residue surface area is calculated by the total amino acid surface area, while the residue accessible area (RAA) is the amino acid surface the center a molecule of water, approximated by a sphere with a radius of 1.4 Angstroms, can reach.

The fact that each paper uses its own amino acid area tables and data sets makes the prediction values not directly comparable. Regarding the PEA level, however, for papers using amino acid area with table [2.1-4], like [4.1-4], as the denominator for the REA calculation it is possible to do some indirect comparison. Since the values in table s surface area are about 1.7 times smaller than the solvent accessible areas, our percentage of exposed area (PEA) roughly corresponds to other papers PEA/1.7. For instance, if other papers are using a 20% PEA value, we should compare it with our 11.7% PEA level (our closest PEA level is 10%).

In Table 2.1-1 we summarize the results of the described computations, with also the Relative Frequency of each amino acid and the Minimum, Maximum and Standard Deviation of the solvent accessible areas.

A. acid	Rel. Freq.	Min SA	Max SA	Avg SA	Std. Dev	SA [2.1-4]
ALA	8,8%	189	234	214	2,02	115
ARG	5,5%	188	379	352	15,01	225
ASN	4,9%	213	300	275	6,54	150
ASP	6,3%	212	293	274	5,84	160
CYS	1,0%	234	266	243	3,41	135
GLN	3,7%	192	324	303	8,99	190
GLU	5,8%	211	328	301	11,75	180
GLY	7,7%	185	214	191	2,76	75
HIS	1,9%	289	326	302	4,28	195
ILE	5,3%	213	302	281	4,33	175
LEU	9,7%	267	312	285	3,73	170
LYS	7,2%	210	351	315	13,78	200
MET	2,4%	212	323	298	5,87	185
PHE	3,7%	305	337	321	4,6	210
PRO	3,3%	213	266	243	2,96	145
SER	5,4%	211	257	234	3,15	115
THR	6,1%	214	278	254	3,26	140
TRP	1,6%	344	379	363	5,5	255
TYR	3,9%	211	362	343	6,23	230
VAL	5,9%	213	277	257	2,83	155

Table 2.1-1 Amino acids area statistics

The last column, the surface area as calculated in paper [2.1-4], is presented to give an idea of the values calculated by other methods. Although the results differ by the reasons we explained above, there is a 0.985 correlation between the surface area values and the residue solvent accessibility values.

We can confirm that the standard deviation is small which indicates the average area is a good approximation of the correct area. All these indicators were added to the *Aminoacids* table although only the Average Solvent Accessible Area (ASAA) will be used on calculations. The ASAA here forth is a synonymous for the amino acid exposable area and is the denominator in the percentage of exposed area (PEA) equation: *Current amino acid accessible area to solvent*/*Total amino acid accessible area to solvent*.

2.1.2 Loading PDBs & DSSPs

The source for the pdb files was the Protein Data Bank (www.rscb.org). We did a local mirror from their FTP server (ftp.rcsb.org) in April 2005. The mirror gave us about 20 gigabytes of compressed data. From those 20 gigabytes, for the context of this work, we are only interested in the biological units (available here: ftp://ftp.rcsb.org/pub/pdb/data/biounit/coordinates/all). These biological units, as of April 2005, were 5.24 gigabytes of compressed data and about 35.000 files, each file representing a bio unit (a protein or a protein complex).

The file name of a protein file is of the form PDB_ID.pdbX, being *pdb_id* the four-letter code of the PDB and *X* a variable we called *biounit_id*, which is usually 1. Only 30% of the *pdb_id*s have more than one bio unit.

Processing a pdb file is a complex task (the format of a pdb file was presented in sub section 1.2.3). First, we verify if the pdb file represents a pure protein instead of molecule with DNA/RNA fragment. If it is pure, we run it to a custom developed parser that decomposes its components into three CSV files. The first CSV file holds the general protein data, such as its id, the bio unit, the date it was added to the PDB repository, its resolutions in angstroms, etc. The second CSV file holds general information about the protein chains, such as to which protein it refers, the number of amino acids it has, the data it was added to our database, etc. Finally, the third CSV file has the description of all atoms of the protein (e.g.: their 3D coordinates, to which amino acid they belong, etc).

To load the DSSP information we do not need any of these CSV files or tables. It just requires as input the original pdb file. The output of the DSSP program is a text file (its format was already presented in sub section 1.2.4), which needs to be parsed to be added to the database. We did a simple parser that converts that text file into a CSV file in order to be then easily loaded to the database.

To load the set of CSV files to the *ProteinsDB*, we have used the SQL Server 2005 command line utility, BCP (Bulk Copy Program), which is quite efficient. To make the loading more efficient we have also turned off the foreign key constraints on the tables involved in this process. The total loading time for a pdb file is highly dependant on the protein size, but on average, it takes about 0.3 seconds, with DSSP being responsible for the majority of the cpu time. The loading itself is mainly an I/O bound task with little cpu overhead.

After all the pdb files are loaded, so is the majority *ProteinsDB*. The database occupies about 15 gigabytes, with the *pdb_data* table just by itself occupying near 14 gigabytes. It has approximately 217 million rows, each describing one atom of some amino acid. The *dssp_data* is the second largest one with near 800 megabytes. It has approximately 12 million rows, each describing an amino acid.

The remaining tables are very small compared to these two. *Chain_Headers* has 55.476 rows with one per chain and *PDB_Headers* has 29.841 rows, one per PDB (these numbers do not consider the new pdb files loaded in June 2006 for validation purposes).

Figure 2.1-1 illustrates the process of loading a single pdb file into *ProteinsDB*.

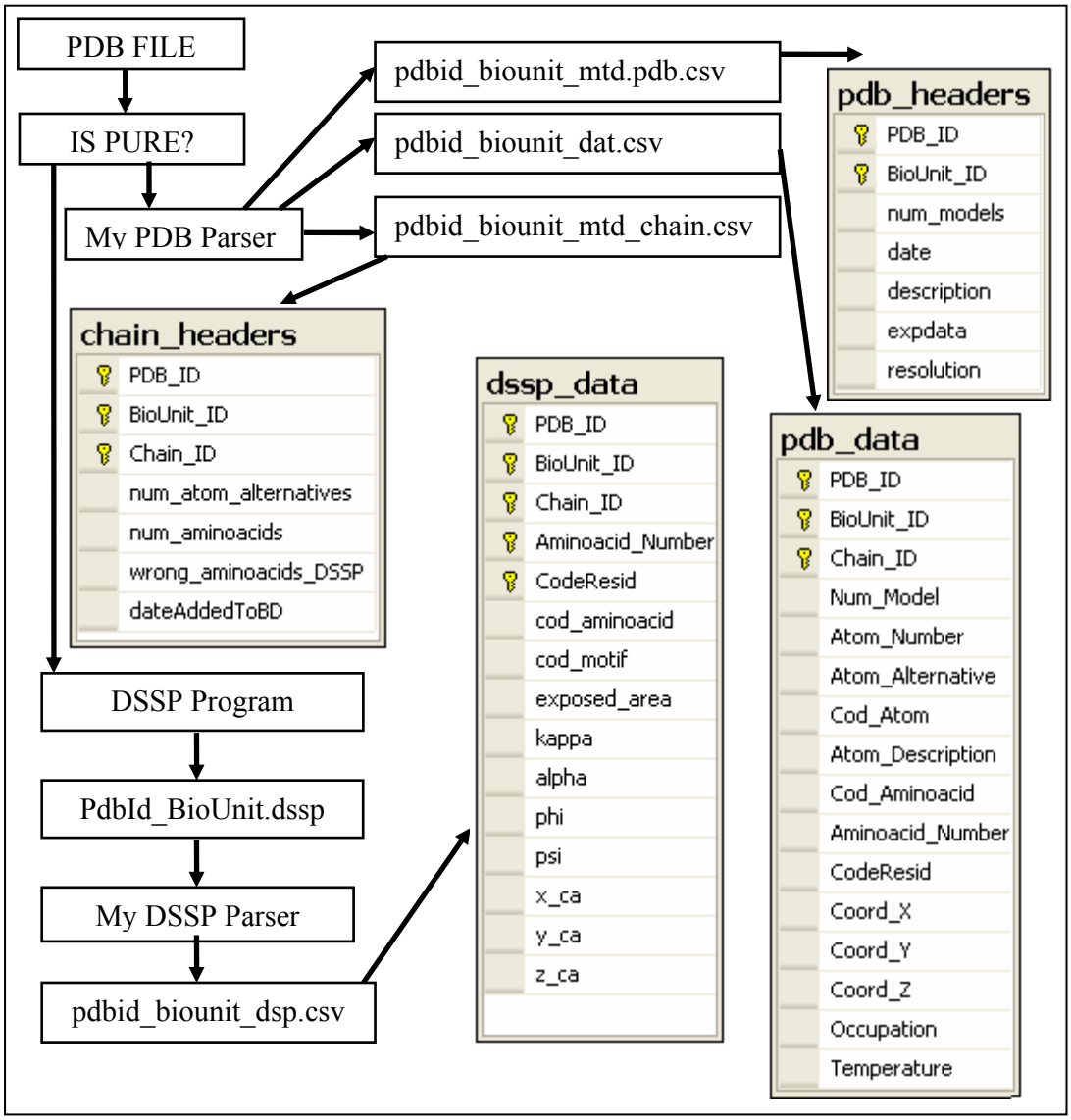


Figure 2.1-1 Scheme of processing a single PDB File

For the main purpose of this project, to build a model to determine if an amino acid residue is buried or exposed inside a protein, the *pdb_data* table is not needed. Nevertheless, this table can be of great use for other projects where information at the atom level is required.

In the same way, of all the information in the *dssp_data* table, for this project we only use the *exposed_area* column. The remaining information on this table, namely the secondary structure element to which the current amino acid belongs to (given by the *cod_motif* column) can be very useful for other projects where information of the secondary structure is required.

2.1.3 Loading SCOP

SCOP is available as a set of four text files: *dir.des.scop.txt_VER*, *dir.cla.scop.txt_VER*, *dir.hie.scop.txt_VER* and *dir.com.scop.txt_VER* with *VER* being the name of the scop version. The only file that we need to parse is the classification file-*dir.cla.scop.txt*. The figure below shows an excerpt of it.

```
d1c7ca1 1c7c  A:1-142   a.1.1.2 15261 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15261
d1c7ca2 1c7c  A:143-283 a.1.1.2 15262 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15262
d1o1oa_ 1o1o  A:        a.1.1.2 81068 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81068
d1o1oc_ 1o1o  C:        a.1.1.2 81070 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81070
d1dxta_ 1dxt  A:        a.1.1.2 15267 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15267
d1dxtc_ 1dxt  C:        a.1.1.2 15268 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15268
d1sdka_ 1sdk  A:        a.1.1.2 15265 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15265
d1sdkc_ 1sdk  C:        a.1.1.2 15266 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15266
d1a01a_ 1a01  A:        a.1.1.2 15271 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15271
d1a01c_ 1a01  C:        a.1.1.2 15272 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15272
d1o1pa1 1o1p  A:1-142   a.1.1.2 81072 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81072
d1o1pa2 1o1p  A:143-283 a.1.1.2 81073 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81073
d1a3oa_ 1a3o  A:        a.1.1.2 15274 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15274
d1a3oc_ 1a3o  C:        a.1.1.2 15275 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=15275
d1o1ja1 1o1j  A:1-142   a.1.1.2 81048 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81048
d1o1ja2 1o1j  A:143-283 a.1.1.2 81049 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81049
d1o1na1 1o1n  A:1-141   a.1.1.2 81064 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81064
d1o1na2 1o1n  A:145-285 a.1.1.2 81065 cl=46456,cf=46457,sf=46458,fa=46463,dm=46486,sp=46487,px=81065
```

Figure 2.1-2 Excerpt of the SCOP dir.cla.scop.txt_1.67 file

Each line of the scop text file represents the classification of an interval of amino acids inside a chain. The first column is a string with six characters identifying the line, but we will ignore it. The second column is the four letter PDB code (the pdb_id column in our tables), the third column is the chain letter (the chain_id column) plus an optional amino acid interval. Finally, in the fourth column, there are, separated by dots, the class, fold, super family and family classification.

In the third column, after the two dots ':' following the chain identifier, there might be, the amino acid interval for which the family classification applies. If no interval is specified, the family classification applies to the whole chain, which is the more common situation.

In the case of chain A of pdb_id 1o1p, the first family classification is valid from amino acid 1 to 142 and the second from amino acid 143 to 283. In this case, for both the intervals, the family classification is the same (i.e.: class=a, fold=1, super family=1 and family=1). This happens because the chain is split in two equal domains. Figure 2.3-2 in section Exploring the database shows this situation.

Since the scop file is ordered by family classification, Figure 2.1-2 does not show cases where the same chain belongs to different families. We will illustrate those cases, along with further comments to scop, in section Cleaning data.

The *cl*, *cf*, *sf*, *fa*, *dm* and *sp*.numbers are not relevant to our problem but we still loaded them because they might be useful in further projects supported by this database. These numbers are related to the domain classification of the chains.

The *px* column is a unique identifier of the scop line and is used as the key for the *SCOP* table. The table scheme can be seen in Appendix A, Figure A-4.

We developed a simple parser to convert the scop text file into a CSV file and then this CSV file was loaded to the *SCOP* table in the database throughout the same procedure described in the previous sub section.

From this point onwards, when we refer to the SCOP classification of a chain we always mean the concatenation of the full path from class, fold, super family and family, instead of just the smallest hierarchy member family.

2.2 Cleaning data

At this stage we have the database created and loaded with the needed data but we still need to join and clean the information gathered. The main task is to clean the *scop* table and merge it with the *dssp_data* table. We will describe the process in this section.

Let us consider first the *scop* table. Recall that there are cases where parts of the chain are separately classified. When that happens, we will call the chain irregular. SCOP 1.67 has family information for 51.376 chains with 40.144 (78.5%) regular and 11.062 (27.5%) irregular. Figure 2.2-1 shows examples of some chains belonging to different families.

	pdb_id	chain_id	class	fold	superfamily	family	start_aminoacid_number	end_aminoacid_number
1	10gs	A	c	47	1	5	2	76
2	10gs	A	a	45	1	1	77	209
3	10gs	B	c	47	1	5	2	76
4	10gs	B	a	45	1	1	77	209
5	11gs	A	c	47	1	5	2	76
6	11gs	A	a	45	1	1	77	209
7	11gs	B	c	47	1	5	2	76
8	11gs	B	a	45	1	1	77	209
9	12e8	H	b	1	1	1	1	114
10	12e8	H	b	1	1	2	115	214

Figure 2.2-1 Some chains which have amino acids belonging to different families

What interests us most in SCOP is not the number of chains but more the number of families represented by them. The 51.376 chains in SCOP represent 2.886 families. The subset of regular chains represents 2.302 families, 79.8% of the total number of families.

A one to one relationship between a chain and its family classification simplifies the modeling of the problem and hence we opted to consider only the regular chains for the mining process. (This choice, decided in early 2005, might have been premature. It would not be very difficult to adapt the database views and Data Mining Table Creator to support irregular chains.)

In SCOP a very small number of chains, 65, appear twice with the exact same information (all columns equal except *px*). For those cases we eliminated the ones with lower *px*. In addition, a small number of chains, 157, although regular, appear as belonging to two different families. Figure 2.2-2 shows some regular chains which belong to several families.

	pdb_id	chain_id	class	fold	superfamily	family	start_aminoacid_number	end_aminoacid_number
1	1a0i	2	b	40	4	6	-1	-1
2	1a0i	2	d	142	2	1	-1	-1
3	1a16	1	d	127	1	1	-1	-1
4	1a16	1	c	55	2	1	-1	-1
5	1a47	4	b	71	1	1	-1	-1
6	1a47	4	b	1	18	2	-1	-1
7	1a6q	2	a	159	1	1	-1	-1
8	1a6q	2	d	219	1	1	-1	-1
9	1a72	1	b	35	1	2	-1	-1
10	1a72	1	c	2	1	1	-1	-1
11	1a76	2	c	120	1	2	-1	-1

Figure 2.2-2 Examples of regular chains that belong to several families

To maintain a one to one relationship between a chain and its family classification, we eliminated those chains from the data mining process. These two steps eliminated 65+2*157 chains, leaving us with 39.765 regular chains suitable for data mining but keeping the 2.302 families. The set of regular chains is available from the database view *PureScopChains*.

The *dssp_data* table, as of April 2005, has 54.365 chains, representing about 12 million amino acids. A small percentage of them, about 0.5%, appear as lower case letters in the *cod_aminoacid* column, which should only contain upper case single letter amino acid codes. In accordance to the DSSP manual, which states lower case amino acid letters mean a SS-bridge Cysteine, these lower case amino acid letters were converted to the upper case letter ‘C’, meaning a Cysteine amino acid.

Of the 54.365 chains present in the *dssp_data* table, 38.736 are in common with the *scop* table, representing 2.408 families. However, since we need to exclude the irregular chains, we have only 29.379 chains (75.8% of the maximum possible total) shared between the *dssp_data* table and the *PureScopChains* view.

This subset of 29.379 chains, from 13.441 different pdbs, represents 1.742 families, which is 72.3% of the total possible families if we had also considered the irregular chains. The suitable pdbs for data mining are available from the database view *Possible_PDBs_For_DM*.

2.3 Exploring the database

In this section we aim to explore the database to get better acquainted to the data before entering the core data mining process.

As we have seen from the previous section, there are 29.384 chains belonging to 1.742 distinct families on SCOP v1.67 that are suitable for data mining purposes. The distribution of the chains along these families is not homogeneous at all. Figure 2.3-1 shows the top 10 common families. The count column represents the number of chains that belong to the specified family.

	class	fold	superfamily	family	count
1	b	47	1	2	718
2	a	1	1	2	681
3	b	1	1	1	374
4	b	40	2	1	373
5	b	50	1	1	364
6	c	2	1	2	360
7	i	15	1	1	335
8	d	2	1	2	332
9	d	144	1	7	326
10	b	29	1	1	280

Figure 2.3-1 Top 10 SCOP families of all possible chains for DM

On the other hand, the large majority of those 1.742 families have very few chains representing each family. 1.180 families have less than 10 chains representing it, with 339 families represented by a single chain. Table 2.3-1 shows the class distribution of the suitable chains for data mining.

SCOP Class	D	C	B	A	G	J	F	H	K	E	I	Total
Chains count	364	362	303	289	115	107	75	41	38	33	15	1742

Table 2.3-1 Class distribution of suitable chains for DM

As we have previously discussed in section 2.1.3, sometimes a chain is divided but all divisions belong to the same family. Figure 2.3-2 shows some an example of this situation.

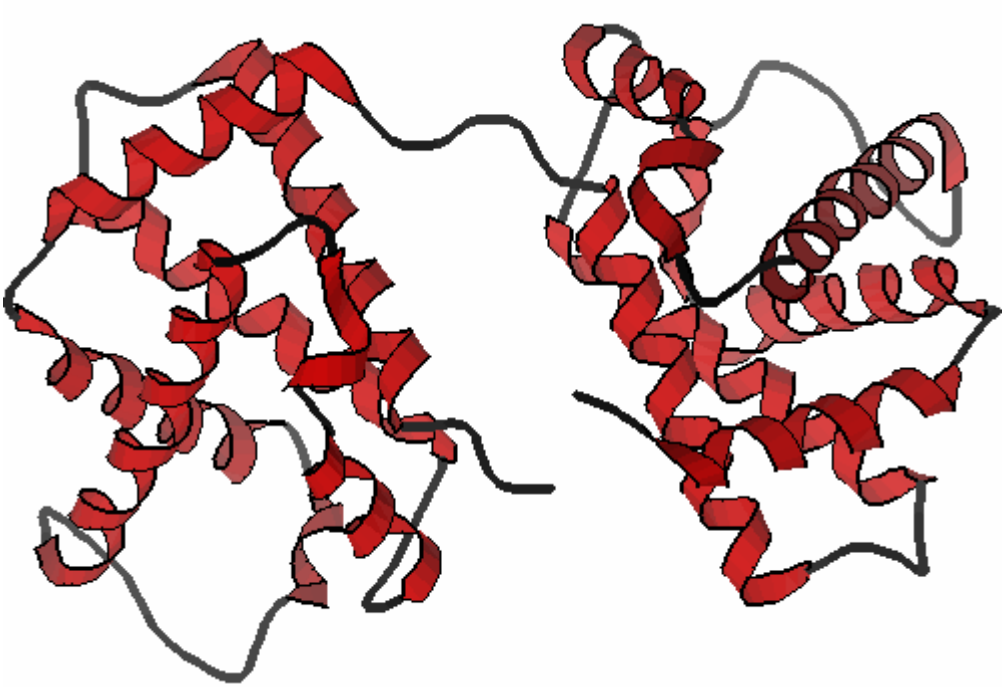


Figure 2.3-2 Visualization of chain A of pdb id 1o1p, which has two equal domains

An interesting insight for the residue solvent accessibility problem is the percentage of amino acids exposed at less than *X*%. Recall that the percentage of exposed area of an amino acid is given by *current exposed area/total amino acid exposable area* (see 2.1.1). Table 2.3-2 shows the percentage of amino acids exposed at less than each one of the 2, 10, 20, 25 and 30% thresholds, for all the chains up to April 2005.

Percentage of exposed area	2%	10%	20%	25%	30%
Percentage of amino acids exposed at <= X %	27.48%	48.88%	65.79%	73.17%	79.98%

Table 2.3-2 Percentage of amino acids exposed at <= X% in all chains up to April 2005

These results can help us build a trivial, yet not that bad, classifier. Let us call it PEAM (Percentage of Exposed Area Model), considering the percentage of all amino acids exposed at less than X% (PAE) in the training set and stating for the test set: *"if PAE(X)<0.5 return Buried else return Exposed"*. The accuracy of this classifier is the maximum between 1-PAE(X) and PAE(X). Table 2.3-3 shows its accuracy:

Percentage of exposed area	2%	10%	20%	25%	30%
PEAM model accuracy	72.52%	51.12%	65.79%	73.17%	79.98%

Table 2.3-3 PEA model accuracy for all chains up to April 2005

2.3.1 Baseline for residue solvent accessibility

The percentage of exposed area model shown just takes into account the percentage of exposed area and returns a unique answer regardless of the input. The idea of the baseline classifier is to improve it by taking into account the exposition of each of the twenty amino acid types.

In [4.1-2], Richardson and Barlow present a paper with a baseline for residue accessibility prediction models. Their baseline consists in assigning a residue into the particular exposure category in which it is most frequently found, not considering its local surrounding sequence. This baseline is the standard by which the literature measures its results. In chapter Experimental Results, we always present our models accuracy as improvement over this baseline classifier.

The baseline classifier works as follow: given a set of proteins as a dataset, it is split in training and test set as usual. In the training set we determine, for each amino acid, the frequency of its appearing in a certain class (i.e.: exposed/buried) for a certain degree of exposition (2%, 10%, 20%, 25% or 30%). The baseline model simple assigns an amino acid to the class where it appears more often (e.g.: if Alanine appears 60% of the time as exposed in the training set, our prediction for all Alanines in the test set is that they are exposed).

The baseline is hence dependent on the dataset and, more specially, on the cut off value for determining if a residue is exposed or buried. It can be considered as a special, very fast to calculate, classifier that has amino acid information *Minimal* and window size equal to zero.

Table 2.3-4 shows the average exposed area of each amino acid type for all amino acids in the *dssp_data* table (excludes amino acids added in June 2006 as explained in 3.4.1) and its corresponding Hydropathy as in [2.1-1].

Amino acid	Average exposed area	Hydropathy (from [2.1-1])
Lysine	29.94%	-69.24
Glutamic acid	26.91%	-79.12
Arginine	23.20%	-80.00
Aspartic	22.44%	-80.65
Glutamine	22.40%	-9.38
Asparagine	20.59%	-9.70
Proline	18.85%	0.00
Histidine	16.11%	-37.20
Serine	15.92%	-5.06
Threonine	15.25%	-4.88
Glycine	12.02%	0.00
Tyrosine	11.24%	-6.11
Alanine	10.33%	1.94
Tryptophan	9.74%	-5.88
Methionine	9.11%	-1.48
Leucine	7.95%	2.28
Phenylalanine	7.77%	-0.76
Valine	7.48%	1.99
Isoleucine	6.81%	2.15
Cysteine	5.38%	-1.24
Average	**15.02%**	**-19.12**

Table 2.3-4 Hydropathy and average exposed area of all amino acids in *ProteinsDB*

There is a correlation of -0.79 between the two columns, which indicates that there is a strong relationship between these two attributes. The correlation is negative because for high values of average exposed area (ASA) corresponds low values of hydropathy and vice-versa. Nevertheless, what matters in a correlation analysis is its absolute value and 0.79 is a high value for correlation. This relationship was expected and was also elicited by the data mining classifiers (see section 3.3.5). In paper [2.1-5] the authors also remark this correlation between ASA and hydropathy.

The baseline classifier can be seen as a switch statement with 20 branches (one per amino acid), returning 1 or 0 (buried/exposed) depending only on the amino acid type. A table with the Baseline model if the training set was the entire database, as it was in April 2005, is presented below.

Amino acid	Percentage of burial				
	2	10	20	25	30
Alanine	0	0	1	1	1
Arginine	0	0	0	1	1
Asparagine	0	0	0	1	1
Aspartic	0	0	0	1	1
Cysteine	0	1	1	1	1
Glutamic acid	0	0	0	0	1
Glutamine	0	0	0	1	1
Glycine	0	0	1	1	1
Histidine	0	0	1	1	1
Isoleucine	0	1	1	1	1
Leucine	0	1	1	1	1
Lysine	0	0	0	0	1
Methionine	0	1	1	1	1
Phenylalanine	0	1	1	1	1
Proline	0	0	1	1	1
Serine	0	0	1	1	1
Threonine	0	0	1	1	1
Tryptophan	0	1	1	1	1
Tyrosine	0	0	1	1	1
Valine	0	1	1	1	1

Table 2.3-5 The five percentage of exposed area baseline models

Surprisingly this very simple classifier is very accurate and performs almost as well as the much more sophisticated machine learning approaches.

2.3.2 Comparing PEAM with Baseline

It is trivial to see that the baseline accuracy can never be worst than PEAM because PEAM is just a particular case of the baseline (much less granular). How does this improvement in granularity increase the accuracy of baseline over PEAM? To answer that question we evaluated both classifiers on unseen data added in 2006 (this process is described in 3.4.1 Updating the database). Table 2.3-6 shows the accuracy of both classifiers for the new data:

Model	Percentage used to determine burial status				
	2%	10%	20%	25%	30%
Percentage of amino acids exposed at <= X %	27.53%	48.98%	65.99%	73.45%	80.26%
PEA Model accuracy	72.47%	51.02%	65.99%	73.45%	80.26%
Baseline accuracy	72.47%	65.15%	71.92%	75.59%	80.26%

Table 2.3-6 Relationship between baseline accuracy and percentage of exposed amino acids

Note that for constructing Table 2.3-3 we have used data up to April 2005 and in the above table data added in 2006. That is the reason the percentage of amino acids (and consequently the PEA Model) differ slightly from the results in Table 2.3-3.

In Table 2.3-6, for the extreme percentages of burial 2 and 30, baseline and PEAM are the same. This is obvious from the information in Table 2.3-5, where we can see that for both 2 and 30 the baseline model is, respectively, all 0s and all 1s, which coincides with the PEA model.

For PEA 10, 20 and 25 the base line significantly surpasses the simpler PEA model. However, at PEA 25 the difference is small because the BL and PEAM coincide in all except two amino acids (glutamic acid and glysine), which BL considers to be exposed. PEA 10 is where the Baseline improves more over PEAM. This is because at PEA 10 the proportion between buried and exposed amino acids is more evenly divided.

2.4 Preparing data

Preparing data for the mining process consists mainly in combining all the relevant data in one table so that it acts as the source for the learning algorithms. Not only we need to combine the data but also divide it properly between training and test sets. This process is called the construction of the datasets.

In this section we will discuss the problems of constructing the mining datasets and the requirements of such datasets. After, we will introduce Data Mining Table Creator, a program we developed to face those requirements.

2.4.1 Construction the mining datasets

Building the datasets is an important step in every data mining problem. Ideally they should be large enough so that existing patterns could emerge. However larger the sets are, the longer it will take to generate a model. The main problem, however, is to get "well suited" data to use in a dataset (i.e. data that avoids model over fitting).

In our work, the data are protein chains and, even considering the cleaning process performed in the previous section, we still have plenty of data available. The main problem, however, is that the vast majority of proteins in the Protein Data Bank (and hence in our database) belongs to clusters with a high degree of homology and that does not serve the purpose of building generic models to determine protein properties because, when the dataset is very similar, there is little that can be learned.

2.4.2 Requirements of the Data Mining table

Proteins in the same SCOP family have a high degree of homology, so we want training and test sets to have chains of distinct families. The data mining table should have chains that belong to different families but, optionally, with some

feature in common like all having the same SCOP class. To be able to distinguish between training, and test (and eventually validation) we also need a *Partition* column.

Each row of the data mining table represents a single amino acid of a chain and some constant and variable information about it. The variable information depends on the window size and information level parameters. We must ensure that all amino acids of the same chain have the same value in the *Partition* column (i.e. all amino acids of a chain must belong to the same set- training, testing or validation).

The total number of columns in the data mining table is given by $K+2*W*I+I$. With K being a small constant (about 10) that represents fixed information about the chain and the current amino acid. I is the number of columns needed to detail an amino acid features which depends on the level of amino acid information (explained in 2.1.1). W is the window size (0, 3, 6 or 10).

The fixed information of each row are attributes like pdb_id, chain_id, biounit_id, These attributes identify the chain to which the current amino acid belongs but are not used as inputs for the learning algorithms. The other fixed attributes serve to identify the amino acid and are its name and total area. These are the only attributes available for an amino acid at information level *Minimal*.

The data mining table must also have five Boolean attributes (the target attributes), which are the ones that the classifiers try to predict. We named them *isBuriedAt02pct*, *isBuriedAt10pct, isBuriedAt20pct, isBuriedAt25pct* and *isBuriedAt30pct*. They are calculated by testing if the ratio between the exposed area of the amino acid and its total area are below or above the specific threshold.

If the amino acid information is other than *Minimal* then, for the current amino acid, a list of attributes (as explained in section 2.1.1) appear prefixed by *Cur_*. If window size is N, for N greater than zero, the same attributes appear to the N amino acids both to the left and right of the current, prefixed by *LeftX_* and *RightX_* (with X between 1 and N).

Another important requirement is that it should be easy to generate similar tables but with different datasets (i.e.: different chains representing the families) so that different experiments could be performed and we could assess the stability of the results.

To build a data mining tables with these requirements a specific program needed to be developed, Data Mining Table Creator (DMTC), which will be explained in the following section.

2.4.3 Data Mining Table Creator

Data mining table creator (DMTC) is a program we developed, in C# 2.0, specifically to build the training, test and validation sets. We decided to develop it because it was neither practical nor easy to build the data sets with the characteristics we need directly in SQL.

So far, except for the small scripts for loading data into *ProteinsDB*, all the tasks-cleaning and exploration- were achieved just in SQL (the query language at the SGBD level). SQL is wonderful for many purposes and is, indeed, the right tool for exploring such a huge amount of information in an efficient and practical way. However, for the creation of a data mining table with the requirements specified in the previous section, SQL was not appropriate.

One of the problems is that, since we vary the amino acid size, the number of columns varies for each different window. Worst than that, it would require many non-trivial joins to generate even a simple table for an amino acid window size of just three. Even overcoming the problem of generating dynamically the several tables for different window sizes, the performance was very poor. With a window size of three and it required several hours to generate just one table, while DMTC creates the same table (with hundreds of thousands of records) in less than five minutes.

Other problem that DMTC solve easily but there was no simple solution in SQL was the creation of randomly generated datasets with control over those sequences.

When the random seed of DMTC changes the list of selected chains for mining is different but still represents the same set of families as the previous list. The selected chains only have in common the same SCOP class, which is a parameter of DMTC.

The determination whether a chain belongs to the training or test set is also a random choice. For each chain we determine if it belongs to the training set by generating a random number between 0 and 1 and testing if it is less than a specified percentage (by default 66%). Since the pseudo random generator is unique, changing the seed, besides changing the list of chains within the family, also changes the distribution of the chains between training and test set.

To summarize, the main purpose of DMTC is, given the *ProteinsDB* database, to find chains that obey to several user-selected criteria and generate CSV files with various level of information (dependant on the parameters) to be used later by Clementine (see section 2.5). The user-selected parameters are saved back to a database table, *DM_Test_Desc*, in order to facilitate the evaluation of the parameters in the mining results (see section 2.6). Figure 2.4-1 illustrates the described data flow.

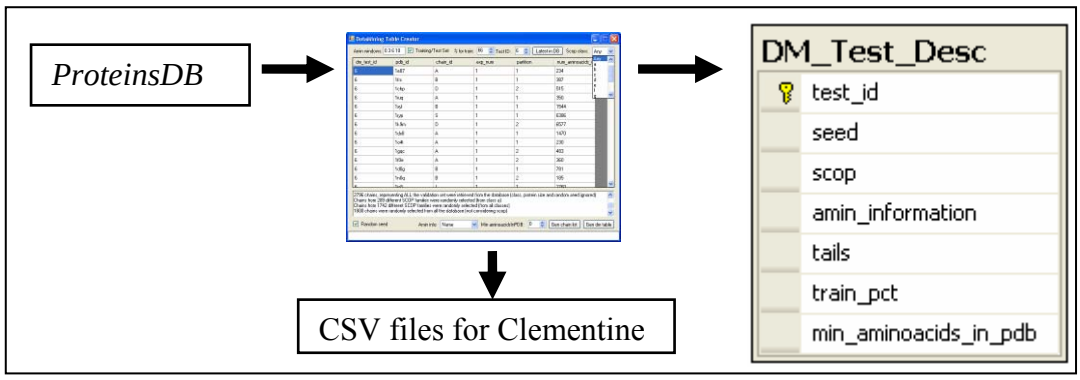


Figure 2.4-1 Data Mining Table Creator data flow

2.4.3.1 Data Mining Table Creator Options

As anticipated by the previous explanation, DMTC has several user selectable parameters. In this subsection we will explain its options. Figure 2.4-2 shows

DMTC's GUI. An identical command line version was also developed to be more easily integrated in batch scripts.

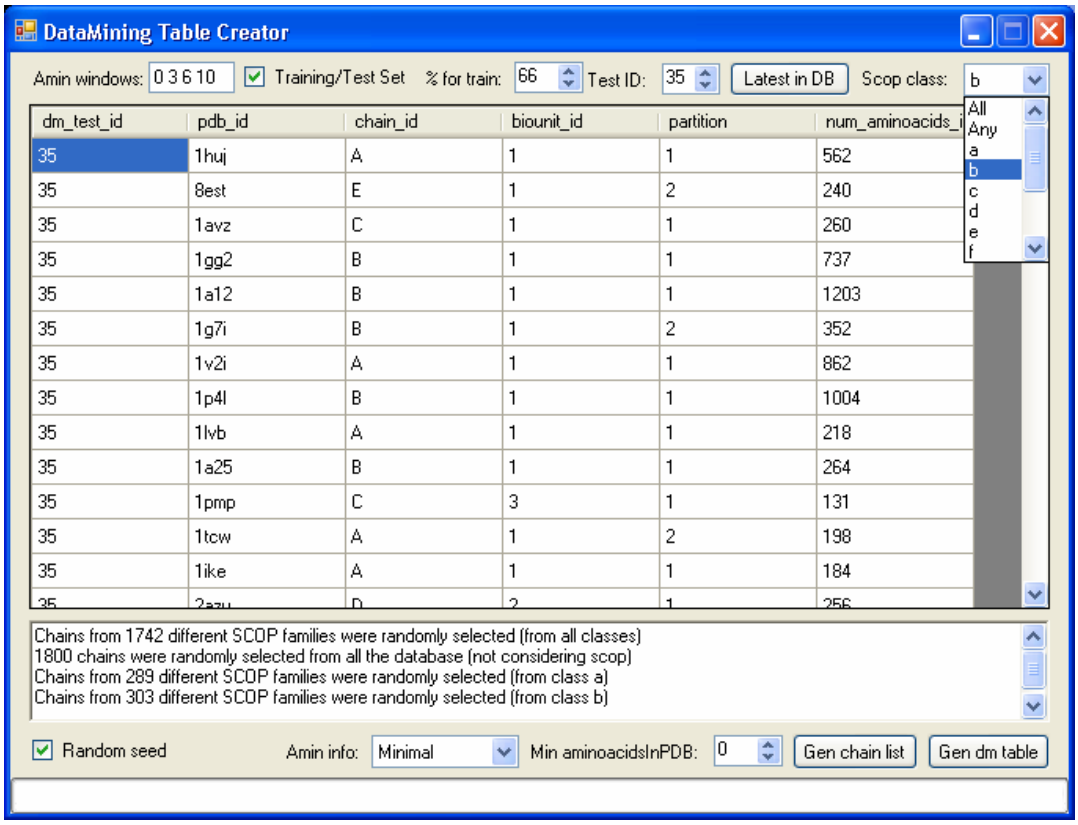


Figure 2.4-2 Data Mining Table Creator GUI

DM_Test Id: It is the ID of the experiment in the *DM_Test_Desc* table. A seed, a training percentage, a SCOP class and a minimum number of amino acids in PDB characterize each experiment.

Minimum number of amino acids in PDB: Sets the minimum number of amino acids that a PDB appearing in the dataset must have. The default is zero.

Amino acid information: Sets the amount of information to consider per amino acid: *Minimal*, *Simple* or *Complete*. The meaning of these information levels is explained in 2.1.1. This setting will be reflected in the number of columns characterizing each amino acid. Each amino acid is represented by 2, 8 or 37 columns in order of crescent level of information.

Random Seed: Seed used to randomly generate the chains of the selected scop class and to decide if a chain belongs to the training or test partition.

Scop class: Sets the scop class of the proteins that appear in the dataset. There are two special classes: *All* and *Any*. The *All* class means that the DMTC will iterate through all distinct families and choose, randomly, one chain representing it. The *Any* class does not care about SCOP. It will just randomly generate 1.800 pdb ids from the whole database, not caring to which scop families or classes they belong. Naturally, since the protein database is biased towards some families, there will be many chains from the same family.

If any other class is selected (i.e. a letter from a to k), all the generated chains will be from different families but will all belong to the selected class.

Amin windows: List of amino acid window sizes to be considered. The user can choose any values and not only 0, 3, 6 and 10. One CSV file is generated per window value, so with the default configuration, four CSV files are generated. Remember that this option has a strong impact in the number of columns of the CSV file and hence in its file size.

Percentage for training: Sets the approximate percentage of chains (not necessarily amino acids) that will belong to the training set. Since the average number of amino acids in chains is similar between training and test chains it is normal that this value also reflects the approximate percentage of amino acids in training and test set.

Training/Test set: If selected the chains are chosen from Scop v1.67. Otherwise, we are in validation mode and the chains are chosen from the new families of Scop 1.69 (see 3.4.3).

2.5 Mining with Clementine

In the previous section we have seen how the CSV files were generated. Those CSV files are the input files for Clementine. Clementine is a commercial data mining software from SPSS. We have chosen it for the reasons explained in subsection Data Mining software. The aim of this section is to explain how Clementine was used to generate the models for the prediction of the burial status of an amino acid.

Clementine can work in interactive or batch mode. We did the first experiments in interactive mode but then switched to batch mode so that all experiments could be performed without user intervention. Figure 2.5-1 shows the data mining stream we built to perform our experiments.

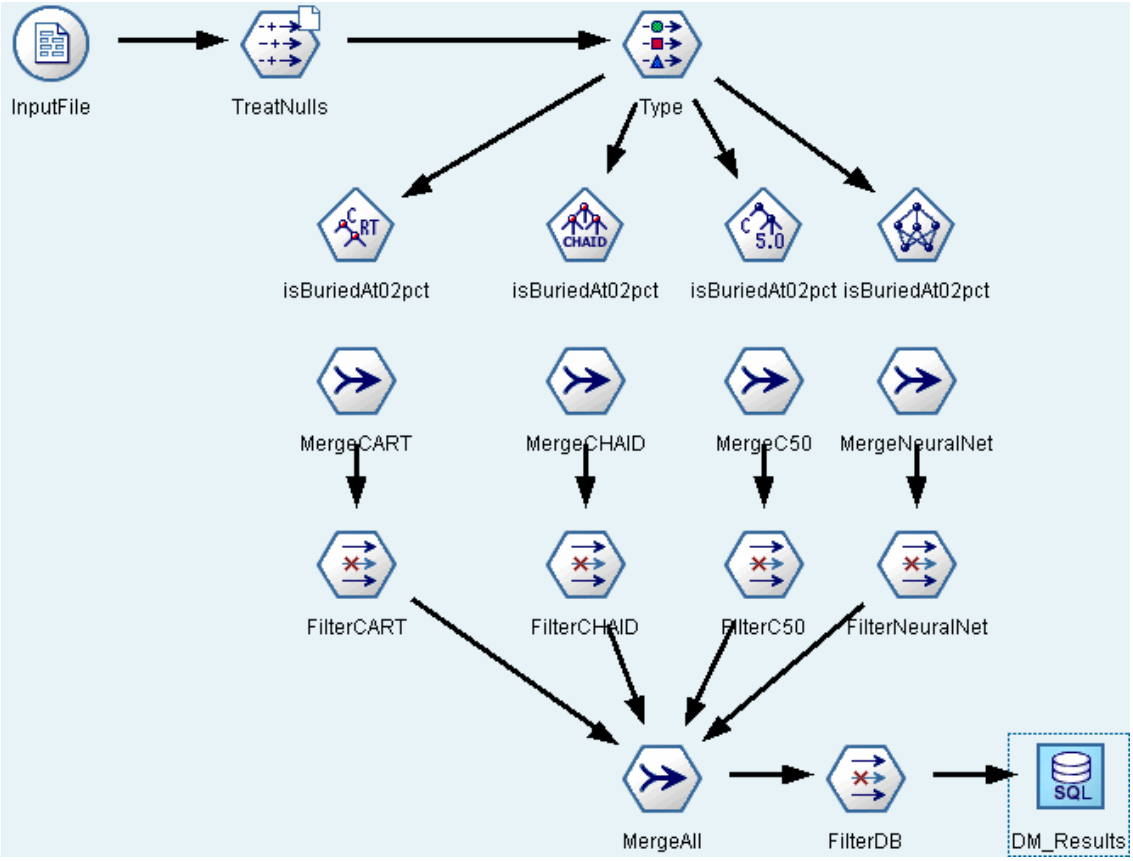


Figure 2.5-1 Base stream for data mining automatization

The input file is an alias for the CSV file we want to use. In the *Type* node we specify which are the input and target fields but also specifying the fields to ignore. For instance, when we want to predict the 2% PEA, *isBuriedAt02Pct* is set as the target field but all the others *isBuriedAtXPct* are set to be ignored. Otherwise, if they were used as input, the classifier would for sure use them because there is a great correlation between the several levels of exposition. If we have a protein for which we do not know the structure we do not know the values of other percentage of exposed areas.

After passing through the *Type* node the data goes to each one of the classifiers. Their label show what is the attribute they are predicting. After generating the model, which can take a long while depending on the data and on the classifier algorithm, a new object appears, a ClassifierXYModel, where X is the classifier type and Y is the PEA value.

The script we developed, executes the command line version of DMTC, varying its seed, window size, amino acid information level and scop class. Each of these variations is represented by a different CSV. Its number of columns is only dependant on the amino acid level information and amino acid size. For instance, with amino acid information *Complete* and a window size of 10 the number of columns is almost 800. The number of records is only dependant of the scop class (with a slight variation with the seed because it implies changing the selected chains and they might have slightly more or less amino acids). For the scop *All* class the CSV file has about 300.000 records and generate a file withis a CSV

For each CSV file generated by DMTC our script assigns it to the input file of the Clementine Stream, then changes the target fields and executes each one of the four classifiers automatically. When the models are generated it connects them to their respective merger nodes and finally execute the recently complete stream.

The execution of the final stream has the effect of writing all predictions for each one of the four classifiers and for every amino acid present in the input CSV file back to the database, into table *DM_Results*. The execution of this stream for a single CSV file, if all classifiers are used, may take up to a couple of hours.

By processing the *DM_Results* table the performance of the classifiers and the baseline is assessed. A brief description of that evaluation is the subject of the following section.

2.6 Evaluating Mining Results

When the script described in the previous section finishes its execution, we have, in the *DM_Results* table of the *ProteinsDB*, all the data needed to calculate the accuracy of the baseline and of all classifiers for the several percentages of exposed area (PEA) levels.

Figure 2.6-1 shows *DM_Results* table schema.

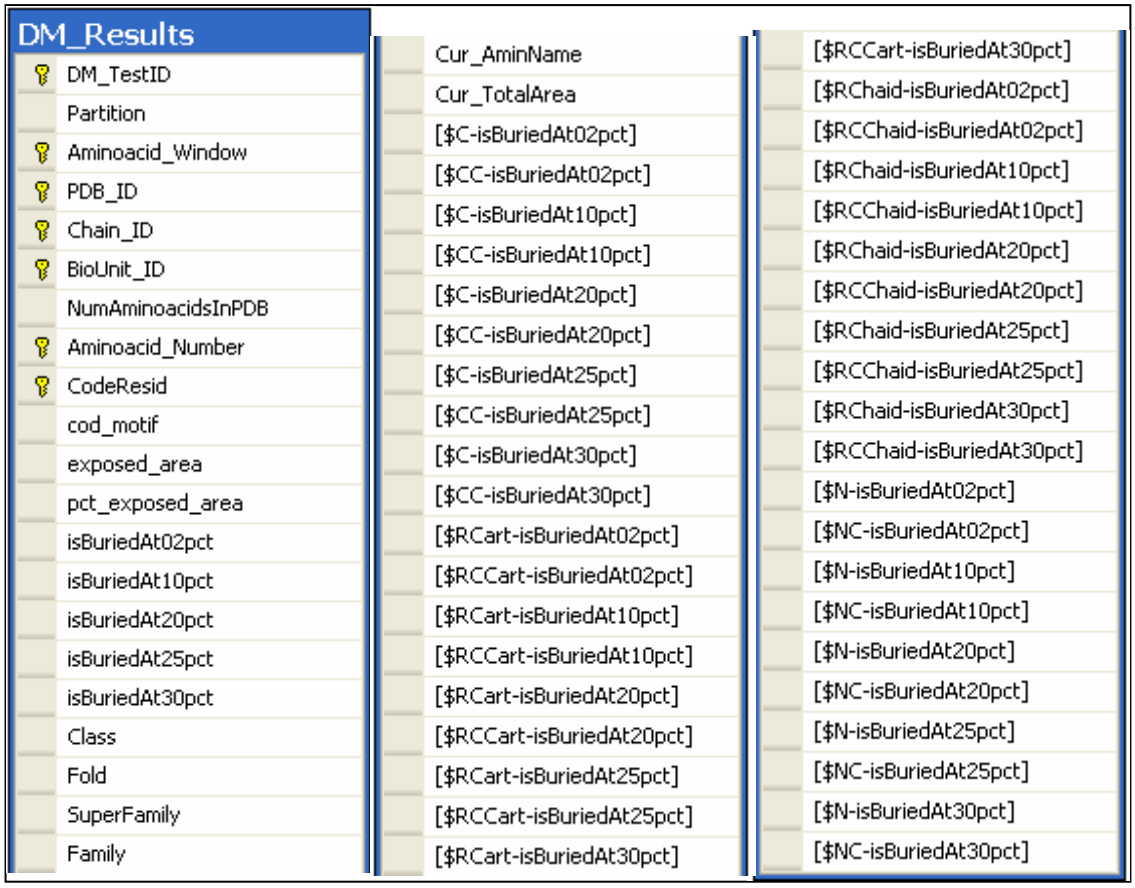


Figure 2.6-1 Data Mining results table

Each row of the *DM_Results* table, besides fully identify the amino acid (columns *PDB_ID*, *Chain_ID*, *BioUnit_ID*, *Aminocid_Number* and *CodeResid*) and the correct burial status (columns *isBuriedAtXpct*), has the prediction of the several classifiers for each amino acid of the dataset. For each classifier and each percentage of exposed area (PEA), we have two columns. One column is the buried

status at that PEA, the other is the confidence the classifier has that the prediction is correct.

We did not use the classifier's confidence column in any experiment. Others (see Related work) have used it and reported no significant gain. Since we have four classifiers and five PEA levels, with two columns per PEA per classifier, we have 4*5*2 columns generated by the classifiers. With other fixed columns, this makes the DM_Results table very wide, with 63 columns.

The columns prefixed by '$' are those created by Clementine. '$C' stands for C5.0 predictions, '$RCart' for Cart, '$RChaid' for Chaid and '$N' for Neural Networks. The columns with another C, like for instance '$CC' stands for the confidence of the classifier for that level of PEA. When, for some reason, the classifier was not executed in that experiment, the respective column appears all filled with *NULL*.

To deal with all this data and to convert it into useful information, many auxiliary views and functions were developed in the database side. By joining the *DM_Results* table, which has key *DM_Test_ID*, with the *DM_Test_Desc* table (presented in Figure 2.4-1) it is possible to calculate the baseline and each classifier performance, for each experiment. Recall that each line of the *DM_Test_Desc* table, also with *DM_TEST_ID* key, represents the characteristics of an experiment and is automatically kept up to date by Data Mining Table Creator.

The main process for assessing a classifier accuracy consists simply in comparing the *isBuriedAtXpct* column (the one that we are sure to be correct) with the respective classifier and level of burial column. For instance, to calculate the percentage of accuracy for the C5.0 classifier at 10% PEA, we compare how many times the value of *isBuriedAt10pct* matches the value of *$C-isBuriedAt10pct*.

We distinguish between the training, testing and validation rows by rows looking at the *Partition* column. We never evaluate the accuracy for the training set. The training set results are only used to build the Baseline model as described in 2.3.1.

An aggregated version of all the results in the *DM_Results* table is given by the *Results* function we developed in Transact SQL at the database side. In fact,

hundreds of thousands of records of the *DM_Results* table are condensed in a few lines. Each line has simply the experiment id, the characteristics of the experience (e.g.: amino acid information level, amino acid window size, min pdb size, random seed) and the accuracies of the classifiers, including the baseline and the simpler PEAM model, for each PEA level.

Next chapter is fully dedicated to present and analyze the results obtained from evaluating the tables returned by this *Results* function.

Chapter

3

3 Experimental Results

In this chapter:

Chapter Organization

In this chapter an analysis of the experimental results is presented. Several important questions will be answered in the next few sections.

We start the chapter by discussing the Experiment Methodology in section 3.1. This section explains the methodology used for the experiments in sections 3.3 and 3.4.

In section 3.2, Statistical Tests, we briefly explain what a Student's T-Test is and how they were used to assess the significance of the results we found for test and validation set experiments.

In section 3.3, Test Set Results, we evaluate the accuracy of the several classifiers with the various parameters at the several PEA levels. Throughout the section we show the importance of the several parameters and terminate it by presenting what we consider the best prediction model.

Finally, in section 3.4, Validation Set Results, we verify if the best model found in section 3.3, which was a choice between the several models for the various parameters and hence we could have biases it, remains accurate for the validation set. The validation set are new chains gathered in June 2006. We conclude that the best model is robust and performs as accurately in the validation set as in the testing set.

3.1 Experiment Methodology

For each percentage of exposed area (PEA), window size, amino acid information type and machine learning algorithm, the model building was repeated five times varying the seed on DMTC so the data sets would differ between tests but remain the same for all the other parameters. This is necessary to apply statistically significance tests. The number of five experiments was a compromise between being able to find statistical differences and speed of experiments.

There are six variables involved in the tests: Classifier, SCOP class, chain size, amino acid window size, amino acid level information and percentage of exposed area used to determine burial status. Each of these variables might assume four to six values, which gives an enormous number of possible experiments (if the average variable domain is five the number of experiments would be: five raised to six times five for the statistical tests, which is 78.125 experiments).

Although all the variables seemed important, we strongly suspected that many of its values were not necessary so we first focused on reducing the variables domain in order to drastically reduce the number of required experiments. As we will see in subsection 3.3.1, a very time consuming step is build a model with each of the four classifiers. In the same subsection we show that training with only one classifier (C5.0) is enough.

For the Test Set Results, the classifier's models and the baseline were built with the training data and evaluated with the test data. For the Validation Set Results, the classifier's models were also built with the training data but evaluated with the validation data. The validation data consists in newly retrieved proteins from the Protein Data Bank. The baseline (BL) was always calculated as explained in 2.3.1 and so did not take into account the neighbours of current amino acid nor has any attributes about the amino acid except is name.

The experiments were not necessarily realized in the order we present them. In fact, they were somewhat iterative as expected by the CRISP-DM Methodology.

The results of some experiments raised new questions that we answered by doing more experiments. The accuracy values presented on the next tables are always an average of the results of five identical experiments keeping all parameters constant and varying only the random seed.

3.2 Statistical Tests

To determine if the classifier's improvements are statistically relevant we have performed mainly Student's t-Test but also ANova tests using Excel's 2003 implementation.

A Student's t-test gives the probability of two random samples from the same population being at least as different as those observed, assuming that the two samples are random samples from the same normally distributed population.

When this probability, called the *p-value*, is small, we can reject the hypothesis that the samples have come from the same normally distributed population (this is called the *null* hypothesis). The alternative hypothesis is then chosen, stating that the samples belong to different populations.

Statisticians often consider a *p-value* lower than 1% or 5% as statistically significant, meaning it is unlikely those results happened by chance but rather because the samples are intrinsically different. In our case, the sample is the result of each experiment for each classifier and is simply a vector set of percentages (each percentage if the accuracy of a classifier for a given experiment).

A t-test receives the two distributions –in our case two vectors of classifier results- to compare and two more parameters, the tails and the type of t-test to perform. We performed a paired t-test instead of the more common independent test because we can pair each experiment to a unique classifier result. Hence, the experiment and result order is not arbitrary but a pair.

Table 3.2-1 shows, for each experiment, the accuracy of Baseline, C5.0 and Chaid for percentage of exposed area 10, and amino acid window size 0. Each experiment was generated with Data Mining Table Creator but with a different seed, which resulted in different data sets, and that is the reason results differ slightly.

Experiment	Accuracy		
	Baseline	C 5.0	Chaid
1	65,80%	68,77%	68,57%
2	65,81%	69,03%	68,83%
3	66,29%	68,92%	68,91%
4	66,09%	68,61%	68,25%
5	65,96%	68,77%	68,75%
Average	65,99%	68,82%	68,66%

Table 3.2-1 Accuracy for Baseline, C5.0 and Chaid at PEA 10, Window 0, Scop *All*

Looking at the averages, we can easily see that C5.0 is the most accurate although the absolute gain is small. The question that statistical tests help to answer is whether this gain may have happened due to chance or because C5.0 is really better than the baseline or Chaid.

This introduces the second parameter of a t-test, tails. A one-tailed test is used if we already know that something is better/worst than other and we are checking if that happened by chance or not. If we have no reason to suspect what is the direction of the change beforehand, we should only check if there is a difference, and hence consider a two-tailed test.

We have opted for a two-tailed test because, in general, we do not suspect a priori the direction of the change (i.e. if the results are going to be better or worst). Note that the results for a one-tailed test are exactly half of a two-tailed test.

The result of applying a two-tailed paired t-test to Baseline and C5.0 columns is 0.002%. This means that, if the null hypothesis is true, there is only a 0.002% probability that the samples are as different as observed. Since this *p-value* is lower than 5%, we reject the hypothesis that they are the same and will say they are difference.

The result of the same paired t-test for C5.0 and Chaid columns is 7.28%. Although C5.0 is consistently better than Chaid for all experiments, considering a *p-value* significant if lower than 5%, we cannot say that C5.0 is distinguishable from Chaid. Note that if we were specifically testing if C 5.0 was better than Chaid (i.e.: one-tailed t-test) the t-test result would be 3.64% and then we could say that C 5.0 was significantly better than Chaid.

If, in the above tests, instead of a paired two-tailed t-test we have used an independent two-tailed t-test, the respective *p-values* would have been 0.00% and 28.42%.

T-tests are quite useful to test whether the means between two groups are statistically significant but if we have several groups the number of t-test need to perform grow very fast (it is the number of combinations of N 2 by 2). This raises the problem that the probability value no longer has the same meaning, it should be multiplied by the number of t-tests performed, which rapidly increases it and renders the t-test useless for more than a few groups.

In this case we may use an ANova Single Factor, Analysis of Variance, test which considers all the groups at once and gives a unique *p-value* to accept or reject the *null* hypothesis that the values of the groups come from the same populations. The result of an ANova test with only two groups is the same as of a two-tailed independent t-test.

We have only needed to use ANova to compare the accuracy between classifiers, since they were four at the same time. For all the other experiments, we only had two groups of values at the same time, hence a student's t-test was simpler to apply and more appropriate because we should consider the experiments results as paired rather than independent.

More information on T-Test can be found in [3.2-1] and on ANova in [3.2-2].

3.3 Test Set Results

In subsection 3.3.1, we compare the several classifiers (C5.0, Chaid, Neural Networks and CART) to find the most adequate to our problem. We show that C 5.0 is much faster than any other classifier and has an average accuracy as good as any other.

In subsection 3.3.2, we assess the importance of SCOP classes for the accuracy of the models, and conclude that SCOP class All (i.e. all classes represented with one chain per family) is the best.

In subsection 3.3.3, we test the importance of protein size in prediction results and conclude that, although the length of the chain is an important attribute, there is no gain in building specific models for chains of a particular size.

In subsection 3.3.4 we compare the models accuracies between the four (0, 3, 6, 10) amino acid window sizes conjugated with the three levels of amino acid information (*Minimal*, *Simple* and *Complete*) and the baseline model. We conclude that the best window size is six, and amino acid information *Simple* has, clearly, the best trade off between accuracy and speed.

Finally, in subsection 3.3.5, we present and discuss what we consider the best prediction model.

3.3.1 Classifier comparison

Clementine 9.0 offers five classifiers: C5.0, Neural Networks, CART (Classification and Regression Trees), Chaid and Quest. Quest failed to build a model on most test cases and so we could not use it.

First, we did an experiment with the four classifiers for all the scenarios-five percentages of exposed areas times four amino acid window sizes times three levels of amino acid information.

The tests were performed on a Pentium IV at 3.06 Ghz with 2 gigabytes of RAM, running Windows XP Professional with Service Pack 2. The database used was SQL Server 2005 RTM and Clementine 9.06.

Chaid and CART were tested with their default parameters. C5.0 was used in boost mode (see Improving accuracy through boosting in section 1.1.4.4) with 10 trials, which increased slightly its accuracy but made its run time about five times slower. The favour mode was also changed to *Generality* instead of *Accuracy* so it would perform better in the test and validation sets. The Neural Network parameters were changed in order to converge faster. The number of layers was fixed to three with respectively 20, 15 and 10 neurons and the persistence decreased from 200 to 10.

The time required for building the model varies significantly with the level of amino acid information and with the amino acid window size. The following table summarizes the times taken by the several classifiers for the distinct amino acid information levels. The full times used to build Table 3.3-1 are found in AII. 2.

Classifier	Amino acid Information level		
	Minimal	Simple	Complete
C 5.0 (Boost)	0.5	2.5	8.5
Chaid	1	3.5	22.5
Cart	3	5.5	56
Neural Network	3.5	5.5	62

Table 3.3-1 Average time, in minutes, taken to build a SCOP *All* model with the four classifiers

Even in boost mode, C5.0 is always the fastest classifier by far and, often, the one with best accuracy. From the statistical significance tests performed, when C5.0 is not the best classifier, only rarely the difference between it and the best classifier was statistically significant at the level of 5%. When ANova tests were performed for the several groups of classifiers' accuracies there was no statistical significance between them- this is not surprising since all classifiers work with the same information theory concepts.

Table 3.3-2 summarizes the accuracies for the different classifiers, including the Baseline, for the special Scop *All* class, with amino acid window size of six and amino acid information level *Minimal*.

By observing the table we can confirm that C5.0 is the most accurate classifier for 10%, 20% and 25% percentage of exposed areas.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline	75.59%	65.99%	70.31%	72.73%	77.60%
C 5.0	75.59%	69.03%	70.83%	73.44%	77.96%
Chaid	75.80%	68.95%	70.74%	73.36%	77.99%
Cart	75.97%	68.88%	70.71%	73.36%	78.14%
Neural Network	75.12%	67.91%	70.18%	72.80%	77.10%

Table 3.3-2 Accuracies for the different classifiers for the Scop *All* model at window 6 and information *Minimal*

From the above, from now on, we have decided to use only C5.0 since it is the fastest classifier and is usually as good as (or sometimes even better than) the other classifiers.

3.3.2 Accuracies for different SCOP classes

One of the aims of this project was to test how the accuracy varies for the different SCOP classes and if there is any gain in building specific models for some SCOP classes.

Using Data Mining Table Creator we would be able to build training, test and validation sets for any of the 11 (letters *a* to *k*) SCOP classes but because of time and interest constraints, we investigated only the four more populous and interesting classes (*a*, *b*, *c* and *d*) which, together, represent about 87% of all the proteins.

Besides these four specific models, two more models were developed. One, called *All*, which has exactly one protein representing each SCOP family. The other, called *Any*, which does not care about SCOP and randomly chooses proteins from the whole database. In no situation SCOP columns (class, fold, superfamily and family) were used as input for classifiers.

Table 3.3-3 shows the baselines for the different SCOP classes at the different PEAs:

Scop Class	Percentage of exposed area				
	2%	10%	20%	25%	30%
All	75,59%	65,99%	70,31%	72,73%	77,60%
Any	72,61%	65,01%	71,87%	75,49%	80,31%
A	77,52%	69,87%	70,89%	71,66%	75,55%
B	76,16%	66,29%	69,34%	72,45%	77,81%
C	70,59%	66,71%	73,15%	76,25%	80,67%
D	76,78%	67,85%	70,31%	72,30%	76,41%

Table 3.3-3 Baselines for the different Scop classes and percentage of exposed areas

Although apparently the baselines do not differ much between SCOP classes, some of the differences are statistically significant. This gives evidence that the baseline is, at least for some Percentages of Exposed Areas (PEA), SCOP class dependent. For instance, at 10% PEA, the baseline for class A is almost 70% when other baselines are about 66%, and at 2% PEA, the baseline for class C is only 70% when most other baselines are about 76%.

More interesting than analysing the baselines are the classifier results for the different SCOP classes. In Table 3.3-4 the improvement over the baseline, using C 5.0 classifier, for amino acid window size six, with amino acid information *Simple* is presented:

Scop Class	Percentage of exposed area				
	2%	10%	20%	25%	30%
All	0.96%	3.75%	0.95%	1.25%	0.92%
Any	6.09%	8.69%	4.24%	3.26%	1.85%
A	1.72%	3.18%	0.85%	0.96%	0.45%
B	1.31%	3.03%	0.62%	0.98%	0.72%
C	3.29%	3.67%	0.69%	0.44%	0.46%
D	0.29%	2.73%	0.47%	0.33%	0.39%

Table 3.3-4 C 5.0 improvement over baseline with window 6 and information *Simple*

For SCOP class *Any* the improvement over the baseline is much larger than for any other class. This is not surprising since the protein database is biased towards some families and since the data contains randomly chosen proteins it is very likely that proteins in the test set belong to the same family as proteins in the training set and hence resulting in this extremely good results. We can also see that, except for 2% PEA, for all other SCOP models (*All, A, B, C* and *D*) the *All* classifier achieves the best accuracy.

We have built specific models for each of the Scop classes to test if they predicted better than the generic Scop *All* model predicts for the specific class. The results were similar. In fact, the Scop *All* model generally performed slightly better than the specific models for each class but the improvements were not statistically significant. Therefore, it is useless to build specific models for each SCOP class, so we will present only the results for the Scop *All* class.

3.3.3 Taking into account the chain length

A simple form to take into account the chain length (i.e.: its number of amino acids) in the learning process is to add it as a field of the dataset. We did that and it increased slightly the accuracy of our models and, more significantly, it is one of the most relevant fields selected by the several classifiers.

We would like to know if bigger proteins had better RSA prediction accuracies. For that purpose, for the same classes above except *Any*, we built specific datasets with Data Mining Table Creator for proteins in the percentile 75 on the number of amino acids. Those are proteins that their number of amino acids is larger than 1100, 700, 1300, 900 and 1050 for, respectively, classes a, b, c, d and the special class *All*.

Running the previously built models to these data sets results in slightly worse results but not statistically significant. The only noticeable fact, however, is that for these percentile 75 datasets, 10% PEA baseline is significantly worse (it is about 63% instead of about 66%) and since the classifiers keep the accuracy, they outperform more significantly the baseline.

We also built specific models for each of these percentile 75 datasets, aiming to see if they predicted better than the generic models, but the results were similar showing there is no gain in building specific models for larger proteins.

3.3.4 Influence of amino acid window size and information level

The experiments we have performed considered four different window sizes - 0, 3, 6 and 10. A window size of *X*, considers the *X* neighbouring amino acids to the left and the right of the current. It would be natural to think that the bigger the amino acid window size, the best the prediction of the burial status for the current, but that is not necessarily true. We also have three different levels of amino acid information: *Minimal*, *Simple* and *Complete*. The differences in these levels of information are explained in 2.1.1 but the number of features for each one is respectively 2, 8 and 37.

Recall from 2.4.2 that the total number of columns of the data mining table is given by $K+2*W*I+I$. With K being a small constant, W the window size and I the number of features of the current amino acid level information. This shows that the window size has a significant effect in the data mining table size and hence in the final time to build a model. Table 3.3-5 shows the average time, in minutes, considering all PEA levels (02, 10, 20, 25 and 30) required by C5.0 classifier (in boost mode), to build a SCOP All model:

Window size	Amino acid Information level		
	Name	Simple	Complete
0	0	0.5	1
3	0.5	2	5
6	0.5	3	10.5
10	1	4.5	18.5

Table 3.3-5 Average time, in minutes, taken to build a model with C5.0 and SCOP *All*

The time required to build a model increases more markedly with the increase in information per amino acid than with the increase in window size. It would be interesting to see if this extra time has some reflection in the model accuracy. Table 3.3-6 shows the increase in accuracy over the baseline gained by applying the C5.0 model for 25% PEA.

Percentage of exposed area: 25% SCOP All (Baseline: 72.73%)				
Amino acid Information	Amino acid window size			
	0	3	6	10
Minimal	0.49%	0.64%	0.71%	0.69%
Simple	0.47%	0.92%	1.25%	1.17%
Complete	0.28%	0.89%	1.18%	1.24%

Table 3.3-6 C5.0 model improvement over SCOP *All*, 25% PEA baseline for the various levels of amino acid information and window sizes

The overall improvements are very small, less than 1.30%in absolute value, showing that the baseline is a good estimate. Still, the biggest improvement is for amino acid information *Simple* and a window size of six. Increasing the window size seems to help although, after window size six, the improvement seems to cease and even regress. In the amino acid information side, adding more information seems to help from *Minimal* to *Complete* but not from *Simple* to *Complete*.

In order to clarify the relevance of the results, a student's t-test over the effect of the change in amino acid window size and information level was performed as explained in 3.2. Recall that the results above are an average of five different experiments varying the datasets and that the statistically analysis was performed on the original data and not on the summaries.

Table 3.3-7 shows the *p-values* for the changes in previous results, due to increased window size. Note that, since we are considering the change and not the improvement, this is a two-tailed test.

Amino acid Information	Change in amino acid window size		
	From 0 to 3	From 3 to 6	From 6 to 10
Minimal	0.26%	46.48%	73.14%
Simple	1.36%	1.12%	44.56%
Complete	0.10%	0.80%	37.06%

Table 3.3-7 *P-values* for change in results, because increased window size for SCOP *All*, 25% PEA

The columns in this table represent the change in window, while the rows represent the amino acid level information. For instance, the cell (*Simple*, From 3 to 6), has a *p-value* of 1.12%. This means that the change in accuracy from 0.92% to 1.25% in Table 3.3-6 has a *p-value* of 1.12%. Being lower than 5% we will reject the null hypothesis that the results (i.e. accuracies for window 3 and window 6) have come

from the same population and accept the alternative hypothesis that the results are from different populations (i.e.: the results are intrinsically different).

We can conclude that increasing the window size from 0 to 3 and from 3 to 6 is useful but increasing from 6 to 10 is useless since there is no evidence the results are different (i.e. p-values are much larger than 5%).

In Table 3.3-8 we do a similar analysis but maintaining constant the window size and changing the amino acid information to test its influence in the model accuracy.

Change in amino acid information	Amino acid window size			
	0	3	6	10
Baseline to Minimal	0.69%	0.02%	0.13%	0.01%
Minimal to Simple	72.88%	2.06%	0.48%	0.10%
Simple to Complete	24.76%	35.21%	2.07%	48.19%

Table 3.3-8 *P-values* for change in results, because increased amino acid information for SCOP *All*, 25% PEA

The first thing to note is that even with minimal information and window size 0, our C5.0 model is statistically different from the baseline. The *p-value* for the improvement by passing from the Baseline to the *Minimal* classifier (a 0.49% improvement by looking at cell (***Minimal,*** window 0) in Table 3.3-6) is only 0.69%. Although the absolute improvement is very small, the *p-value* shows it is unlikely it happened by chance but rather due to the merit of the classifier.

We also note that there is a difference in passing from *Minimal* to *Simple*-which, looking at the accuracy table, is reflected in better predictions. On the other side, passing from *Simple* to *Complete* proves to be useless since the *p-values* are high.

The patterns shown here for 25% PEA are valid for the other thresholds (2%, 10%, 20% and 30%). Their tables, with the respective significance results, can be found in AII. 3. In 30% PEA, for amino acid window size 0, the C5.0 gain over the baseline is negative. A negative gain is very rare and may only happen when amino acid window size is zero. This is partly because the baseline is already very accurate and because, for those cases, other classifiers like a neural network might

be more suitable (even though the results are just marginally higher than the baseline and not statistically different from C5).

3.3.5 Best Model

From the results above, we can conclude that the best compromise between speed and accuracy is achieved with a window size of six and *Simple* amino acid information. In fact, as we have seen before, a window size of 10 and *Complete* information, besides taking much more time, might even lead to worse results (perhaps because the classifier algorithm gets puzzled with so much data). We have also seen that the SCOP *All* model is indeed the most generic classifier because it performs as well as the specific model for their own classes.

There are, in fact, five best models, one for each of the five PEA values (2, 10, 20, 25 and 30). They are, however, very similar - varying sometimes the values where to split the intervals and rarely the order of attributes- and hence showing one is enough for illustration purposes. Figure 3.3-1 shows the top excerpt of a C5.0 decision tree, as presented by Clementine, for 10% PEA with a window size six, amino acid information *Simple* and for SCOP *All*.

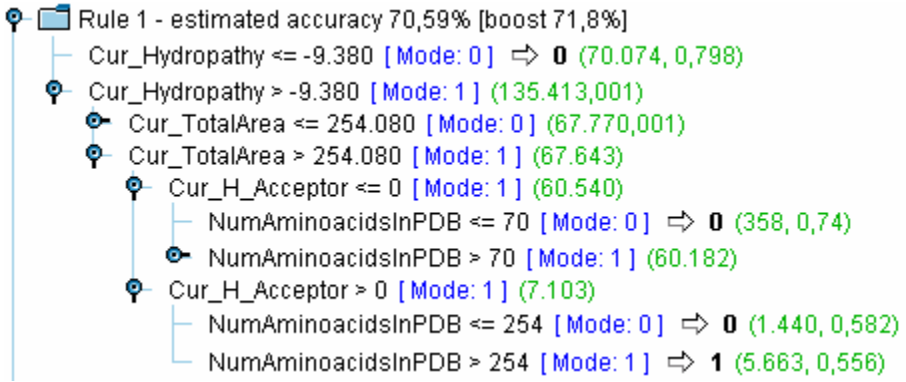


Figure 3.3-1 Top excerpt of C5 model for PEA 10, information *Simple*, window 6 and SCOP *All*.

We have chosen PEA 10 only because it is the where the gain over the baseline is more noticeable. We believe that only happens because this specific cut off value makes the baseline less accurate.

The most important attribute to predict the buried status of the current amino acid is its Hydropathy (*Cur_Hydropathy* column in the data mining table). If it is lower than -9.380 C5.0 will terminate immediately, answering the amino acid will be exposed (0), otherwise the default option will be buried (1) but it might change depending on subsequent attributes.

Hydropathy, being the most significant attribute, is precisely what we expected beforehand. A low value of hydropathy means that the amino acid is hydrophilic, while a high value means the amino acid is hydrophobic, and hence there is a chemical attraction by the hydrophilic amino acids to the solvent and repulsion for the hydrophobic ones. This relationship between hydropathy and burial status was already seen in Table 2.3-4.

The second most important attribute is the total area of the amino acid. If the hydropathy of the current amino acid is higher than -9.380 and the total area is smaller than 254 the default prediction changes from 1 (buried) to 0 (exposed). Other important attribute is the number of amino acids in the current chain (previously discussed in 3.3.3). The analysis continues for other attributes as depicted in Figure 3.3-1.

Another important consideration is that, in the excerpt of the decision tree presented, the only attributes in use are the ones about the current amino acid (the ones prefixed with *Cur*). The attributes regarding the neighbouring amino acids are also used but have a much lower importance, appearing lower in the tree. However, as we have seen in section 3.3.4, they are statistically significant and responsible for the improvement in accuracy of the classifier.

3.4 Validation Set Results

In the previous section, we have shown the results of the data mining algorithms applied to the test data. The main conclusions from those results were to consider an amino acid window size of six, amino acid information *Simple*, the C 5.0 classifier and SCOP *All*, therefore the experiments presented in this section were all performed with those parameters.

We validate the best models by running them with new data, gathered more than a year after the one used for training and testing the models. This validation test is very important to test the robustness of the conclusions taken from the previous experiences.

Subsections 3.4.1 and 3.4.2 briefly explain the update and data exploration phase, which is identical to the ones described extensively in chapter 2. In subsection 3.4.3 the results of applying the models to the validation data are presented and discussed.

3.4.1 Updating the database

The database described in chapter 2 was initially built with PDB files available from the Protein Data Bank as of April 2005 and SCOP 1.67. To validate the model we decided the best approach would be to download the new PDB files (as of June 2006) and the latest SCOP version (1.69 from July 2005, unfortunately as of June 2006 that was still the latest version).

To update our database we created a new column in table *chain_headers* to store the date the chain was added to the DB so that we could distinguish existing chains from new chains. The process to add these PDB files was identical to the one described in 2.1.2. At the end of this process, we 15.890 new chains were added, from 6.540 new pdbs. The new SCOP version was also added to a new table,

SCOP169, identical to the *SCOP* table but representing the latest SCOP information.

3.4.2 Exploring the validation data

The new *SCOP169* table is not exactly a superset of *SCOP* because 150 chains had their SCOP classification changed. There are 3114 distinct families in SCOP 1.69 (comparing to 2886 in SCOP 1.67), from which 1.918 (comparing to 1.742) can be used for data mining, because, as before, we just consider the regular chains.

View *DM_ValidationSet* has the 2.796 possible chains for validation, which are the new regular (i.e.: chains classified in a single SCOP family as explained in 2.2) chains that had not changed classification. These 2.741 chains belong to 557 distinct families, 179 of them are new in SCOP 1.69.

The new families (i.e.: families that did not exist in SCOP 1.67) are particularly important because it is with chains from those families that the validation results are more meaningful. Otherwise, validating with chains that, although new, belong to families that already exist is not very different from validating with any other chains from the same family since there is little difference at the family level in SCOP. We are mainly interested in validating the model with the proteins from new families. Table 3.4-1 shows the distribution of the new families in SCOP 1.69 by their classes.

SCOP Class	D	A	C	B	F	G	E	J	K	H	I
Family Count	59	40	29	16	9	6	6	6	3	3	2

Table 3.4-1 Class distribution of new families in SCOP 1.69

3.4.3 Results

The tables below contains the results of running the best model, presented in 3.3.5, built with the training set, against the several validation sets.

The validation sets were built with Data Mining Table Creator and are generated in the same way as the training and validation sets. The only difference is that all

amino acids in a CSV validation set have the partition flag set to three, meaning validation mode.

The baseline for validation was always calculated with all the amino acids of the current experiment. This could have the effect of slightly increasing the accuracy of the baseline but we have seen that this has no practical implication since it is virtually the same to compute the statistics (i.e.: the average exposed area of each amino acid type) for the baseline with a significant portion of the data or the whole data.

We performed three validation experiments. In the first, all the validation data was (i.e.: all the new chains) used. In the second, only chains from new families were used. Finally, in the third we compare the accuracy of the C50/All model with the C50/D model applied only to the chains which belong to new families of SCOP class D (which is the most populous family of the new families in validation data).

Table 3.4-2 shows the first experiment results, where we have applied the SCOP *All* and *Any* models, built with the training set, to all the validation data.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline	72.59%	67.66%	71.94%	75.19%	79.66%
C50/All	74.61%	70.49%	72.58%	75.94%	80.38%
C50/Any	76.35%	71.74%	73.96%	76.96%	80.72%

Table 3.4-2 Results of applying the best model to all validation data

Not surprisingly, the model disregarding SCOP families (C50/Any) predicts better than the one considering one protein per SCOP family (C50/All). This happens because the validation set, as we have seen, is not family homogeneous.

These results should be compared with those in Table 3.3-3 and Table 3.3-4. The validation set baselines are identical to the baselines for the SCOP *Any* model of the test data (Table 3.3-3). However, the improvement of the SCOP *Any* model against the baseline is significantly smaller here, because this model tends to over fit the training and in new data, although not very different, it performs significantly worst. For the SCOP *All* model the improvements are identical to the

ones seen in the test data, anticipating that it has generalized well enough and does not make distinction between test and validation set.

This experiment only shows that the recent data added to the database is identical to a random part of the already existing database. We should emphasize that this experiment is not very helpful by itself since, in a real world scenario, it is easy to determine if the protein belongs to an existing family, and if that is the case, its structure can be determined by homology.

Table 3.4-3 shows the *p-values* for this first validation experiment.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline-C50/All	0.00%	0.05%	0.27%	0.03%	0.03%
Baseline-C50/Any	0.00%	0.01%	0.00%	0.01%	0.01%
C50/All-C50/Any	0.01%	0.03%	0.01%	0.05%	0.01%

Table 3.4-3 *P-values* for differences in classifier results in first validation experiment

We can easily see that all the *p-values* are below 1%, most of the cases even below 0.1% which show how extremely unlikely it is that the classifier accuracies are values taken from the same population. From this, we can discard the *null* hypothesis and conclude the results are intrinsically different.

The second experiment is the most interesting because it only uses the new families. Only in this scenario one can evaluate how well the generated models for the training and test data have generalized. Table 3.4-4 shows the result of applying the best model to only the new SCOP families in the validation data. Only one chain was chosen to represent each family.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline	76.35%	62.72%	70.63%	72.71%	76.66%
C50/All	77.41%	71.06%	71.43%	73.92%	77.50%
C50/Any	76.44%	69.63%	70.35%	73.21%	77.14%

Table 3.4-4 Results of applying the best model to the new SCOP families in validation data

Here we can see, clearly, that the C5.0/All model always predicts better than the C5.0/Any and that, except for the PEA 10%, the SCOP *Any* model barely surpasses

the baseline. This is a clear indication that the *All* model has generalized very well and that the *Any* model has clearly over fitted not being much more useful than the baseline to predict the burial status of an amino acid.

Table 3.4-5 shows the *p-values* for this second validation experiment.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline-C50/All	0.00%	0.00%	0.00%	0.00%	0.01%
Baseline-C50/Any	0.24%	0.00%	11.19%	19.94%	0.04%
C50/All-C50/Any	0.13%	0.00%	0.00%	0.03%	0.01%

Table 3.4-5 *P-values* for differences in classifier results in second validation experiment

From this *p-value* table, for 20% and 25% PEA, we cannot reject the *null* hypothesis that the Baseline and C5.0/Any classifiers' results have came from the same population (i.e.: that they predict with the same accuracy). For all the other scenarios, the *p-values* are smaller than 1% showing we should reject the *null* hypothesis and consider that the classifiers' results in Table 3.4-4 have come from different populations (i.e.: they predict with different accuracies).

In the third and last validation experiment, we are interested in determining if a model built specifically for a certain class in the training set would outperform the generic SCOP *All* model on its own class. From section 3.3.2 we already suspect the answer is no.

Table 3.4-6 compares the results of applying the generic SCOP *All* and *D* models, built with the training set, and applied to all chains that belong to new *D* families, but ensuring only one chain per family.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline	76.59%	68.02%	70.93%	72.65%	77.24%
C50/All	77.55%	70.85%	71.80%	73.32%	77.36%
C50/D	76.94%	70.18%	71.50%	73.36%	77.60%

Table 3.4-6 Results of applying best model to new D families in validation data

The results show that the C5.0/All model accuracy is always better than the C5.0/D model (except for 30% PEA) and both are better than the Baseline. It might seem a

little surprising that a generic model outperforms a model that has been trained specifically with chains from that class, but it confirms the conclusion reached in section 3.3.2.

Probably this is because the knowledge gained by having seen proteins from different classes, as happens with C5.0/All, contains more information to predict the structure of new families than a model that only knows about the existence of a single class, as is the case of C5.0/D.

Table 3.4-7 shows the *p-values* for this third validation experiment.

Classifier	Percentage of exposed area				
	2%	10%	20%	25%	30%
Baseline-C50/All	0.03%	0.00%	0.00%	0.05%	6.01%
Baseline-C50/D	0.26%	0.00%	0.04%	0.00%	0.17%
C50/All-C50/D	0.81%	0.01%	0.26%	57.26%	1.41%

Table 3.4-7 *P-values* for differences in classifier results in third validation experiment

The interesting point to note from this *p-value* table is the 30% PEA column. The small improvement from C5.0/D over C5.0/All is significant at a level of 5% but not at a level of 1%. For 25% PEA the slight gain of C5.0/D over C5.0/All is clearly not significant. It also seems that if we had higher PEA levels the C5.0/D model would likely outperform, with a statistical significant result, the generic C5.0/All model. However, it makes little sense to try to predict higher PEA values (the outcome is that the huge majority are buried).

In general, these three experiments results confirm that the best model for each PEA level is very robust and performs roughly as accurate in the test set as in the validation set.

Chapter

4

4 Conclusions

In this chapter:

4.1 Related work

There are several papers (e.g.: [4.1-1], [4.1-2], [4.1-3] and [4.1-4]) describing the problem of predicting the relative solvent accessibility (RSA). As explained before in section 2.1.1, the RSA is the ratio between Residue Exposed Area (REA) and Reside Total Area (RTA). The calculation of REA is consensually achieved using the DSSP program but the methods differ for the Residue Total Area values.

It is not possible to directly compare the prediction results between the papers because each paper uses a different scheme to calculate the residue total area (which thereby affects the REA calculations). This is thoroughly discussed in sub section 2.1.1. Besides the different scheme for calculating the residue total area, the fact that each paper uses a different dataset also makes the comparison between literature results harder.

A determining part of the RSA prediction is, obviously, the data set used. As a general note we have observed that, while there was some care in the literature to avoid training and testing with similar proteins, the richness of their datasets (i.e. the number of available proteins) was far smaller than ours. In this section we will try to compare the results but, more importantly, the techniques used.

In [4.1-2], dated from 1999, Richardson and Barlow presented the paper that describes the baseline and which, since then, is used as a benchmark by the more recent papers in the area, specifically the three other and also this thesis. They noted the need for a benchmark and that their simple yet accurate classifier is almost as good as the much more complex methods.

In [4.1-3], Adamczak, Porollo and Meller do an estimation of RSA, using neural networks, rather than imposing cut off points. They point that define cut off points is non-physical and causes difficulties to classifiers so they opt for a pure estimation approach, achieving a correlation between the predicted and real value

of 0.65. They also claim that translating it to a classification problem with a cut-off of 25% in the RSA gives about 77% accuracy while.

In [4.1-4], Gianese, Bossa, and Pascarella also note how the RSA accuracy is highly dependant on the cut off value. Besides creating the conventional two state model (buried/exposed) they also create a three state model (buried/intermediate/exposed). In addition, a number of different PEA levels are used: 0, 5, 10, 25 and 50. Their reported accuracy for the 10 and 25 PEA are 71.2% and 70.3% respectively.

In [4.1-1] Chen et al present a work similar to ours. The aim of this paper is comparing classifier prediction accuracy for the RSA problem with a fixed cut-off of 20%. The classifiers they use are Neural Networks, C 5.0 (the same implementation as ours), Support Vector Machine, Bayesian Statistics and Multi Linear Regression.

Their datasets are built from 2148 unique proteins, taken from FSSP database, with low sequence identity (<25%) and number of amino acids per chain at least 90. Their validation set had just 21 proteins but also with low sequence similarity to the ones in the training set. The FSSP database uses Dali as the structural classifier. We chose SCOP for the reasons explained previously in Methods of Structural Classification of Proteins and built a much more generic approach to create training and validation datasets.

They have tested window sizes from 1 to 19 in increments of 2, reporting that half of the gain of window 19 was achieved using window size of 3.They also note it is useless to use window sizes larger than 9, because there was no improvement in accuracy from 9 to 19. We chose our 0, 3, 6 and 10 window sizes before knowing these results but those window values seem to have been good choices.

They do a more in depth analysis of the protein size effect on RSA prediction. Their dataset is divided accordingly to the protein size, showing that the percentage of exposed area decreases with an increase in chain length. This is natural since,

while the protein becomes bigger, the larger the ratio between its volume (a cubic function) and area (a quadratic function) becomes.

While comparing the classifiers accuracy they also arrive to the same conclusion we did in 3.3.1 by noting that the difference between classifier's accuracy is not statistically relevant.

4.2 Final Summary

Throughout this thesis we have reached several conclusions. Here we will briefly summarize them.

Regarding the prediction of residue solvent accessibility we have seen that, from the four classifiers analyzed (Chaid, Cart, Neural Networks and C5.0), although their results are in general not statistically different, C5.0 is much faster than the others and has the added benefit that its model is easier to understand by a human.

As we have seen in section 3.4, by learning with chains from different SCOP families (the *All* models), rather than by randomly choosing chains among the entire database (the *Any* models), the models generalize better. In addition, the *All* models perform as well in specific family data (e.g.: when all the chains are from a *D* class) as the specific models in the same data.

The protein size has a moderate importance in determining the burial status of an amino acid. The most important attribute is the amino acid hydropathy. This was already anticipated.

One of the main results is that our models are a statistically significant improvement over the very simple and fast to compute Baseline classifier, although the Baseline classifier achieves levels of accuracy only slightly lower than the more complex machine learning approaches.

With the description of the work we executed, we believe we have achieved the three objectives to which we aimed at the beginning, the building of a relational protein database, insights in Protein Structure Data and a set of models to predict the buried status of amino acids.

Along with this thesis, as explained in Appendix C, a DVD is distributed which allows all the work performed in this thesis to be reproduced.

4.3 Future work

To continue this work a natural step would be to consider the confidences of the classifiers and apply some kind of voting procedure. However, that was already done in [4.1-1] and the improvement is not significant. It seems there is a consensus in the area that for predicting Residue Solvent Accessibility there is not much more that can be done with the methods explained in this thesis and in the papers of Related work. More information or different techniques are needed to improve the achieved accuracies.

In respect to Marco Correia's work [1.3-1], this thesis results might help in improving his heuristic to predict what atom domain should be excluded. To test whether the ideas presented on this thesis improve his heuristic, the first approach would be to implement a variant of the baseline model presented for percentages of exposed area between 2 and 10%. Just by itself that should increase the accuracy visibly. A little more accuracy can be achieved by implementing the best model for the chosen accuracy but that would be a more complex task. Since his heuristic works at the atom level the idea would be to check the amino acid where the atom lies and consider that all the atoms of the same amino acid have the same domains excluded.

A further improvement would be to develop a method to predict the buried status of a single atom inside a protein. That would require information at the atom level and would require an atom level version of DSSP that would be able to calculate the exposable area of an amino acid rather than an entire amino acid.

In a more personal note, this thesis also fomented an interest in the more broad area of BioInformatics and hence I decided to pursue my Phd in the area. I received a scholarship both from the Fundação para a Ciência e Tecnologia to start the Phd at FCT-UNL in Lisbon, and from the Wellcome Trust to start the Phd at the Imperial College in London. I opted for the latter where, after a mandatory first year of formal training in biology, I will pursue research in the area of BioInformatics.

References

Data Mining overview

[1.1-1] Inteligência Artificial – "Fundamentos e Aplicações" - Costa, Ernesto; Simões, Anabela Simões

[1.1-2] The CRISP-DM Process Model

[1.1-3] "Algoritmos para a predição da estrutura terciária de proteínas"- Dissertação de mestrado de Alexandre Paulo Lourenço Francisco, Dezembro de 2004

[1.1-4] "Comparison of target selection methods in Direct Marketing" – Dissertação de mestrado de Sara Alexandra Cordeiro Madeira, Julho de 2002

[1.1-5] "Rough sets theory and data reduction in information systems and data mining"- Mofreh Hogo, Miroslav Šnorek

[1.1-6] Statsoft DataMining techniques: http://www.statsoft.com/textbook/stdatmin.html

[1.1-7] Clementine 9.0 Algorithms Guide

[1.1-8] Clementine 9.0 Node Reference

[1.1-9] Enterprise Miner SEMMA
http://www.sas.com/technologies/analytics/datamining/miner/semma.html

[1.1-10] Wikipedia Cross-Validation http://en.wikipedia.org/wiki/Cross-validation

[1.1-11] Wikipedia Decision Tree http://en.wikipedia.org/wiki/Decision_tree

[1.1-12] Data Mining methodology usage http://www.kdnuggets.com/polls/2002/methodology.htm

[1.1-13] Clementine 9.0 User's Guide

[1.1-14] Wikipedia Artificial Neural Network http://en.wikipedia.org/wiki/Artificial_neural_network

[1.1-15] Neural Network Delta Rule: http://uhavax.hartford.edu/compsci/neural-networks-delta-rule.html

Protein Background

[1.2-1] Definition of Secondary Structure of Proteins (DSSP) http://swift.cmbi.ru.nl/gv/dssp/

[1.2-2] Description of the DSSP output: http://swift.cmbi.ru.nl/gv/dssp/descrip.html

[1.2-3] Structural Classification of Proteins (SCOP) http://scop.mrc-lmb.cam.ac.uk/scop/

[1.2-4] Protein Data Bank http://www.rcsb.org

[1.2-5] PDB Format description:
http://www.rcsb.org/pdb/file_formats/pdb/pdbguide2.2/guide2.2_frame.html

[1.2-6] A consensus view of fold space: Combining SCOP, CATH and the Dali Domain Dictionary: http://www.proteinscience.org/cgi/content/full/12/10/2150

[1.2-7] A musical introduction to Protein Structure:
http://www.whozoo.org/mac/Music/Primer/Primer_index.htm

[1.2-8] Introduction to Structural Classification of Proteins: http://scop.berkeley.edu/intro.html

[1.2-9] Wikipedia Protein: http://en.wikipedia.org/wiki/Protein

[1.2-10] Wikipedia SCOP: http://en.wikipedia.org/wiki/Structural_Classification_of_Proteins

[1.2-11] Introduction to Biological Units and the PDB Archive: http://ww.rcsb.org/pdb/static.do?p=education_discussion/educational_resources/biounit_tutorial.html

Problem History

[1.3-1] Marco Correia "Heuristic search for Protein Structure Determination", 2004, Master thesis

[1.3-2] L. Krippahl and P. Barahona, PSICO: Solving Protein Structures with Constraint Programming and Optimisation, Constraints, Constraints, Vol. 7, No. 3/4, Kluwer Academic Press, pp. 317-331, July/October 2002

Loading data

[2.1-1] Wang Y, Zhang H, Scott RA 1995 . A new computational model for protein folding based on atomic solvation. Protein Sci 4(7):1402-11.

[2.1-2] Aminoacid Complete Information: http://wwwtest.bionet.nsc.ru/mgs/programs/crasp/texts/AA_Properties.htm

[2.1-3] L. Shrake A, Rupley JA. Environment and exposure to solvent of protein atoms: lysozyme and insulin. J Mol Biol 79: 351-371 (1973).

[2.1-4] Chothia, C.(1976).The nature of the accessible and buried surfaces in proteins. *J. Mol. Biol.* 105, 1-14.

[2.1-5] Rose, G. D., Geselowitz, A. R., Lesser, G. J., Lee, R. H., and Zehfus, M.H. (1985).Hydrophobicity of amino acid residues in globular proteins. *Science*, 229(4716):834-838.

Statistical Tests

[3.2-1] Wikipedia T-Test http://en.wikipedia.org/wiki/T-test

[3.2-2] Wikipedia A-Nova http://en.wikipedia.org/wiki/Analysis_of_variance

Related work

[4.1-1] Chen, H., Hu, X., Yoo, I. and Zhou, H.-X. (2004). Classification Comparison of Prediction of Solvent Accessibility From Protein Sequences. In Proc. Second Asia-Pacific Bioinformatics Conference (APBC2004), Dunedin, New Zealand. CRPIT, 29. Chen, Y.-P. P., Ed. ACS. 333-338.

[4.1-2] Richardson C. J., Barlow D. J. The bottom line for prediction of residue solvent accessibility. Protein Eng 12: 1051--1054 (1999).

[4.1-3] R. Adamczak, A. Porollo and J. Meller; Accurate Prediction of Solvent Accessibility Using Neural Networks Based Regression, Proteins: Structure, Function and Bioinformatics, 56(4):753-67 (2004).

[4.1-4] G. Gianese,. F. Bossa, and S. Pascarella. Improvement in prediction of solvent accessibility by probability profiles, Pmrein Engineering. Vol 1612, pp 987-992, 2003.

Chapter

5

5 Appendices

In this chapter:

Appendix A Database Description

A.1 List of tables, views and functions

Tables:

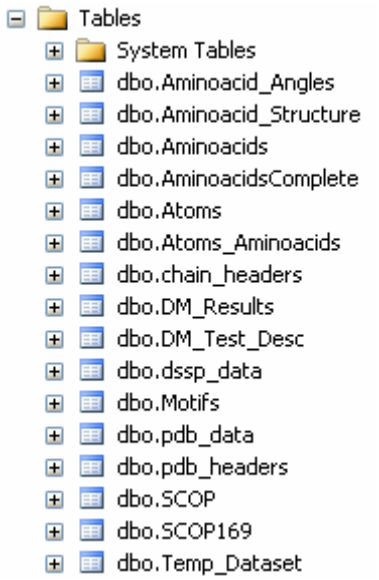


Figure A-1 List of tables of ProteinsDB

Functions:

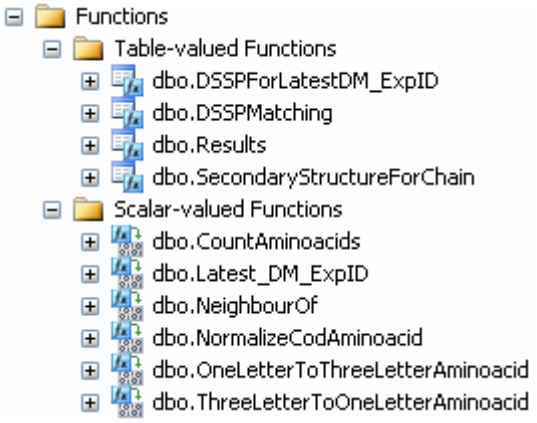


Figure A-3 List of functions of ProteinsDB

Views:

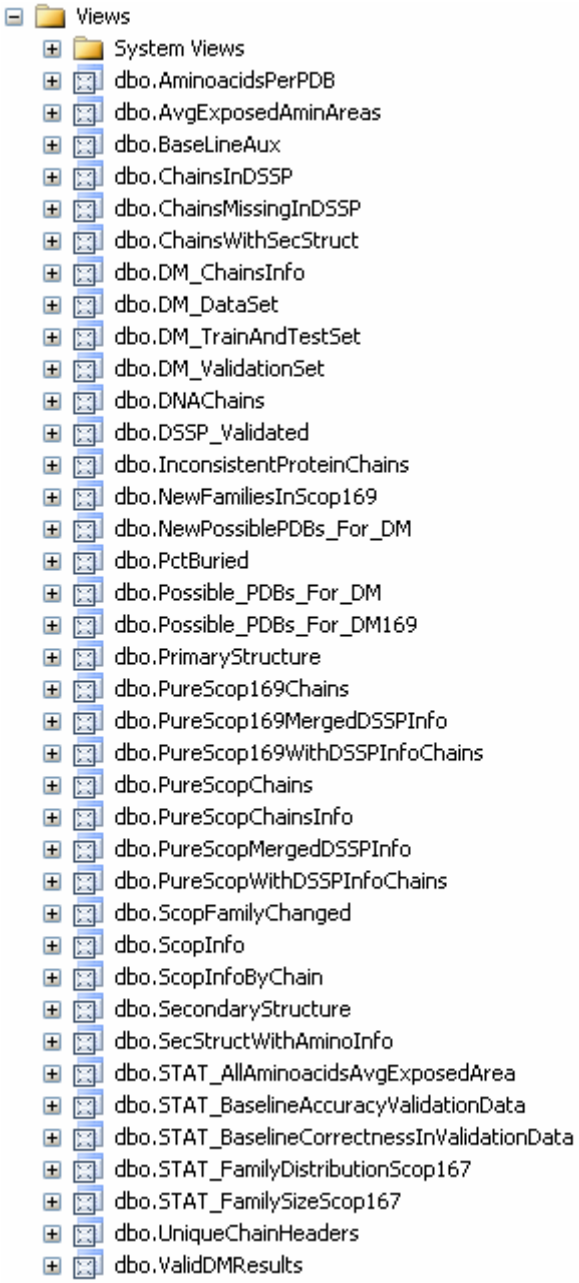


Figure A-2 List of views of ProteinsDB

A.2 Database schema

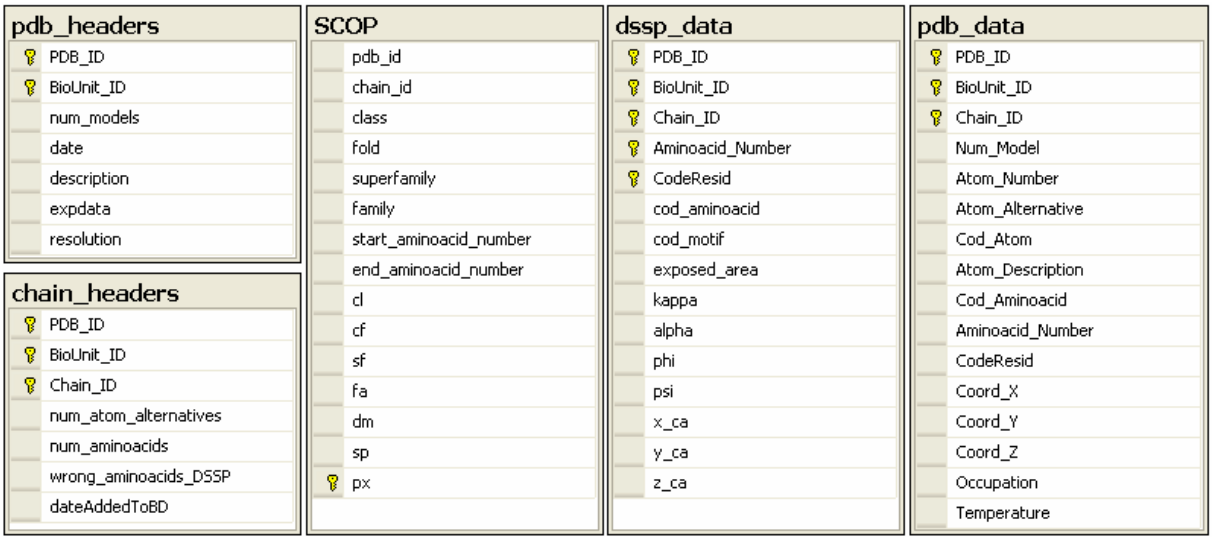


Figure A-4 Tables pdb_headers, chain_headers, SCOP, dssp_data and pdb_data

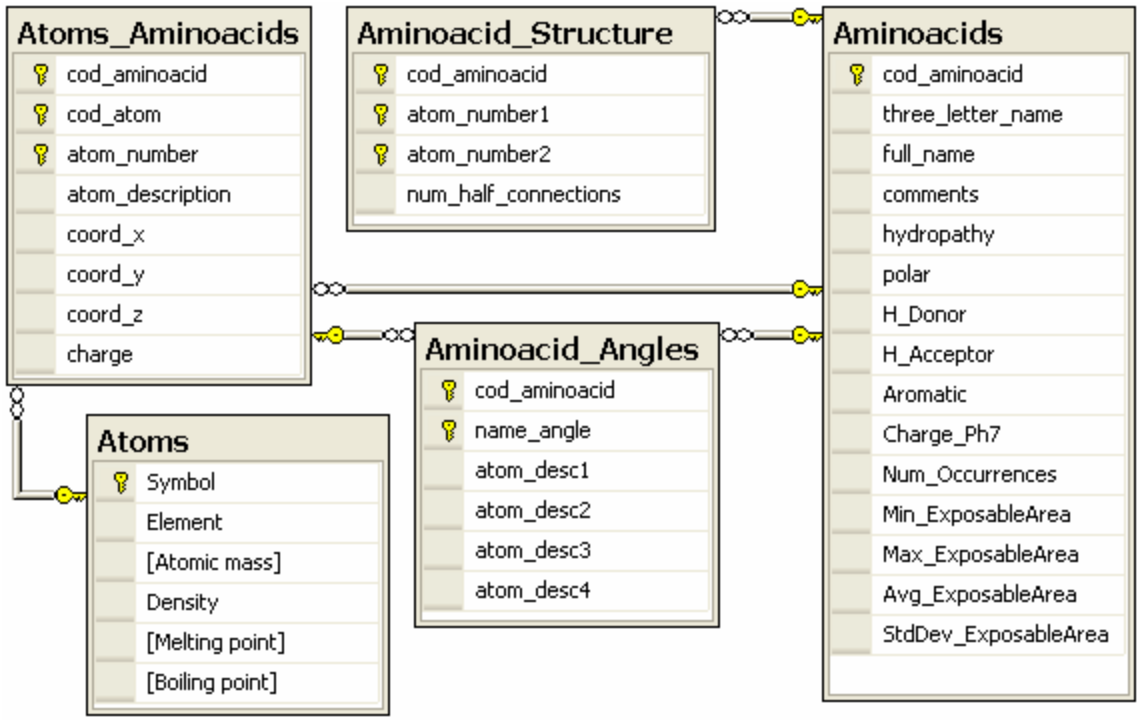


Figure A-5 Tables related to Amino acids

Tables *Atoms*, *Atoms_Aminoacids*, *Aminoacid_Structure* and *Aminoacid_Angles* are constant (i.e.: have immutable date), but are not used in this project. However, as with the *pdb_data* table, in a further project that requires atom level information these columns will likely be very useful.

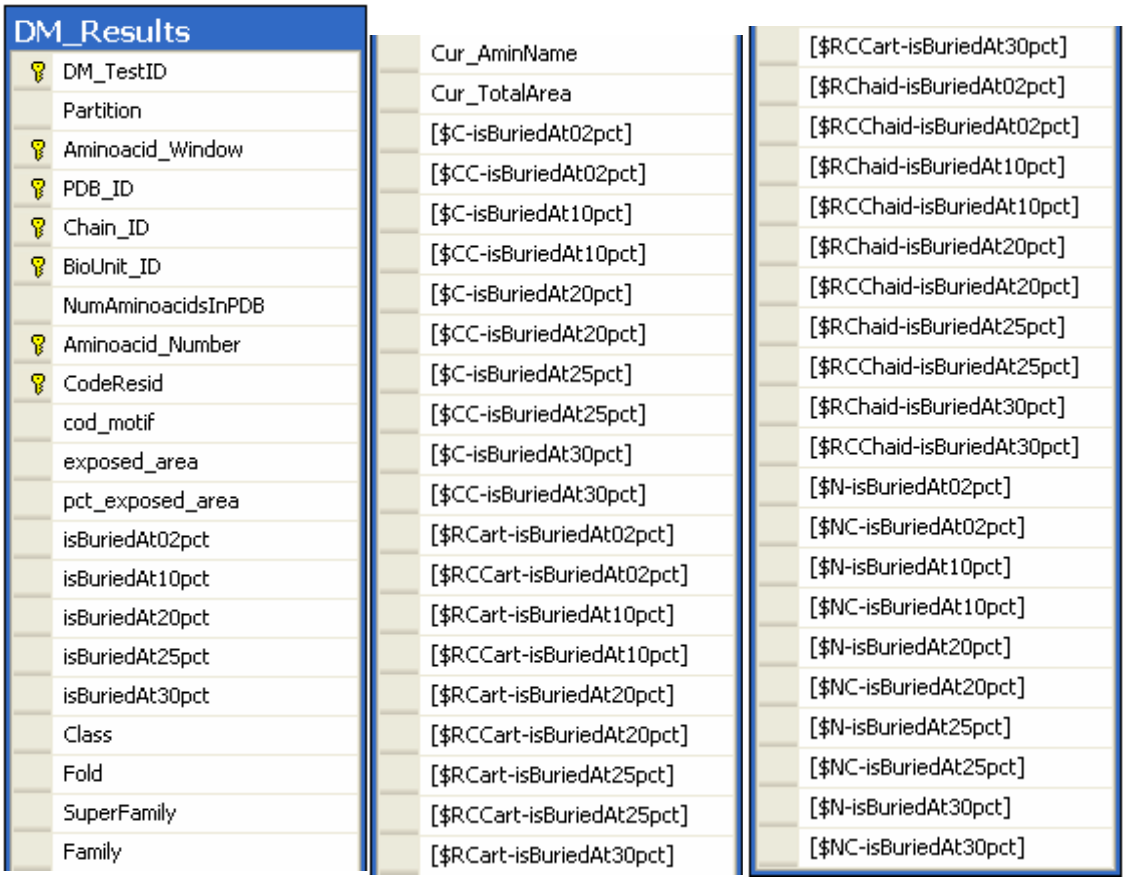
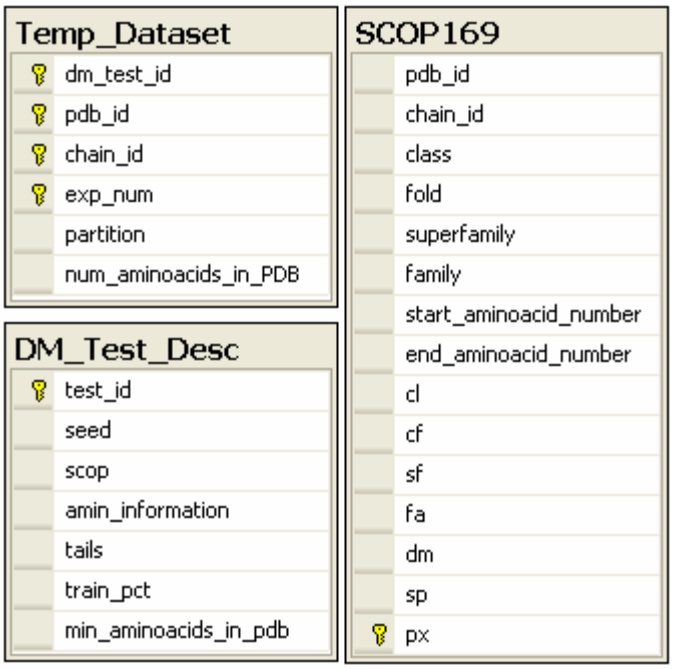


Figure A-6 DM_Results table



SCOP169 table schema is equal to the SCOP table. The difference is that former holds information from scop version 1.69 while the latter holds information from version 1.67.

Temp_Dataset table is a temporary table Data Mining Table Creator uses to write the chains it selected for the mining.

DM_Test_Desc table is where Data Mining Table Creator stores the parameters of the generated datasets.

Figure A-7 Temp_Dataset, DM_Test_Desc and SCOP169 tables

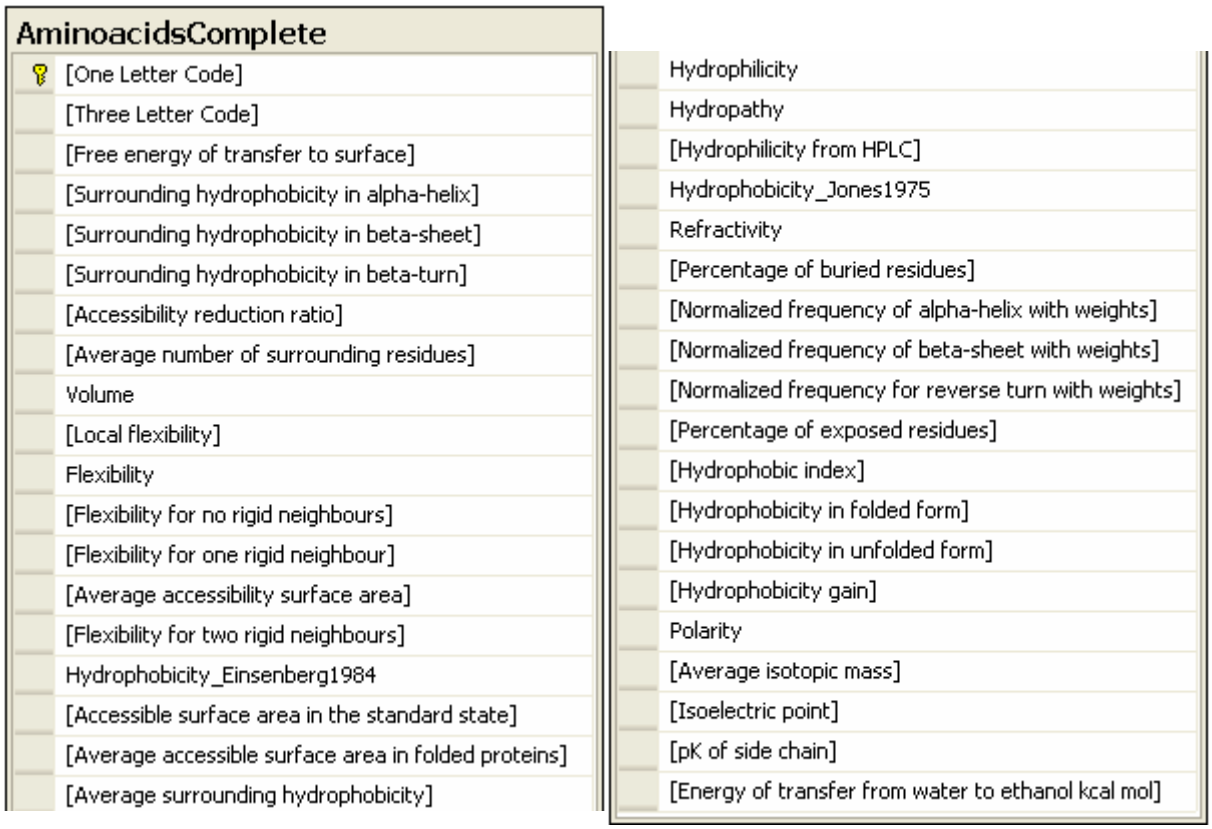


Figure A-8 AminoacidsComplete Table

The *AminoacidsComplete* table is built from 37 amino acid features taken from [2.1-2].

Appendix B Additional Information

AII. 1. List of the 20 amino acids. First column is the one letter code, Second column is the three letter code and third column is the full amino acid name.

cod_aminoacid	three_letter_n...	full_name
A	Ala	Alanine
C	Cys	Cysteine
D	Asp	Aspartic
E	Glu	Glutamic acid
F	Phe	Phenylalanine
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
K	Lys	Lysine
L	Leu	Leucine
M	Met	Methionine
N	Asn	Asparagine
P	Pro	Proline
Q	Gln	Glutamine
R	Arg	Arginine
S	Ser	Serine
T	Thr	Threonine
V	Val	Valine
W	Trp	Tryptophan
Y	Tyr	Tyrosine

AII. 2. Times, rounded to the nearest minute, to build a model for the different levels of amino acid information

Window size	Percentage of exposed area																			
	2%				10%				20%				25%				30%			
	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN
0	0	0	0	2	0	0	0	2	0	0	0	4	0	0	0	3	0	0	0	2
3	0	1	2	2	1	1	2	2	1	1	2	3	1	1	2	3	0	1	2	3
6	0	1	4	3	1	1	3	4	1	1	4	4	1	1	4	4	0	1	4	4
10	1	2	6	6	1	2	6	5	1	2	6	5	1	2	7	6	1	2	7	5

Table B-1 Minutes taken to build a model using amino acid information *Minimal*

Window size	Percentage of exposed area																			
	2%				10%				20%				25%				30%			
	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN
0	0	0	1	2	1	0	1	4	1	0	1	2	1	0	1	3	0	0	1	1
3	2	2	3	3	2	2	3	3	2	2	4	3	2	2	3	4	2	2	3	4
6	3	4	7	6	3	4	6	6	3	4	7	7	3	4	7	7	3	4	7	7
10	5	6	11	10	5	6	10	9	4	7	11	9	4	11	11	10	4	7	11	9

Table B-2 Minutes taken to build a model using amino acid information *Simple*

Window size	Percentage of exposed area																			
	2%				10%				20%				25%				30%			
	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN	C5	CH	CR	NN
0	0	1	15	13	1	1	1	8	1	1	1	5	1	1	1	15	1	1	1	16
3	5	12	24	39	5	14	27	50	5	15	27	40	5	16	26	40	6	2	26	40
6	11	25	59	70	10	28	64	71	9	31	65	68	10	32	61	71	12	33	65	69
10	18	42	128	121	16	46	133	130	16	51	122	126	21	58	135	112	19	42	136	143

Table B-3 Minutes taken to build a model using amino acid information *Complete*

AII. 3. Tables of C 5.0 results for SCOP *All* models

Tables for percentage of exposed area 2%:

Percentage of exposed area: 02% SCOP All (Baseline: 75.59%)				
Amino acid Information	Amino acid window size			
	0	3	6	10
Minimal	0.00%	0.00%	0.00%	0.00%
Simple	0.00%	0.70%	0.96%	0.93%
Complete	0.00%	0.42%	0.94%	0.92%

Table B-4 C5.0 model improvement over SCOP *All*, PEA 2% baseline for the various levels of amino acid information and window sizes

Amino acid Information	Change in amino acid window size		
	From 0 to 3	From 3 to 6	From 6 to 10
Minimal	100%	100%	100%
Simple	0.00%	0.22%	27.00%
Complete	1.30%	2.00%	79.00%

Table B-5 *P-values* for change in results, because increased window size, for SCOP *All*, PEA 2%

Change in amino acid information	Amino acid window size			
	0	3	6	10
Baseline to Minimal	100%	100%	100%	100%
Minimal to Simple	100%	0.02%	0.07%	0.29%
Simple to Complete	39.10%	9.40%	80.60%	92.03%

Table B-6 *P-values* for change in results, because increased amino acid information, for SCOP *All*, PEA 2%

Tables for percentage of exposed area 10%:

Percentage of exposed area: 10% SCOP All (Baseline: 65.99%)				
Amino acid Information	Amino acid window size			
	0	3	6	10
Minimal	2.81%	3.13%	3.04%	3.08%
Simple	2.59%	3.70%	3.75%	3.69%
Complete	2.75%	3.40%	3.47%	3.76%

Table B-7 C5.0 model improvement over SCOP *All*, PEA 10% baseline for the various levels of amino acid information and window sizes

Amino acid Information	Change in amino acid window size		
	From 0 to 3	From 3 to 6	From 6 to 10
Minimal	1.16%	15.06%	22.62%
Simple	0.08%	16.92%	41.70%
Complete	1.20%	66.60%	9.62%

Table B-8 *P-values* for change in results, because increased window size, for SCOP *All*, PEA 10%

Change in amino acid information	Amino acid window size			
	0	3	6	10
Baseline to Minimal	0.00%	0.00%	0.00%	0.00%
Minimal to Simple	18.09%	0.05%	0.15%	0.32%
Simple to Complete	60.07%	5.42%	9.29%	64.17%

Table B-9 *P-values* for change in results, because increased amino acid information, for SCOP *All*, PEA 10%

Tables for percentage of exposed area 20%:

Percentage of exposed area: 20% SCOP All (Baseline: 70.31%)				
Amino acid Information	Amino acid window size			
	0	3	6	10
Minimal	0.36%	0.52%	0.52%	0.47%
Simple	0.37%	0.65%	0.96%	1.05%
Complete	0.36%	0.53%	0.75%	0.63%

Table B-10 C5.0 model improvement over SCOP *All*, PEA 20% baseline for the various levels of amino acid information and window sizes

Amino acid Information	Change in amino acid window size		
	From 0 to 3	From 3 to 6	From 6 to 10
Minimal	3.30%	100%	23.33%
Simple	0.00%	0.96%	33.42%
Complete	0.10%	0.80%	37.06%

Table B-11 *P-values* for change in results, because increased window size, for SCOP *All*, PEA 20%

Change in amino acid information	Amino acid window size			
	0	3	6	10
Baseline to Minimal	29.40%	0.91%	1.21%	2.20%
Minimal to Simple	88.29%	2.63%	0.53%	0.04%
Simple to Complete	66.94%	22.19%	14.53%	5.46%

Table B-12 *P-values* for change in results, because increased amino acid information, for SCOP *All*, PEA 20%

Tables for percentage of exposed area 30%:

Percentage of exposed area: 30% SCOP All (Baseline: 77.60%)				
Amino acid Information	Amino acid window size			
	0	3	6	10
Minimal	-0.41%	0.53%	0.46%	0.45%
Simple	-0.24%	0.65%	0.92%	0.84%
Complete	-0.05%	0.83%	0.85%	0.84%

Table B-13 C5.0 model improvement over SCOP *All*, PEA 30% baseline for the various levels of amino acid information and window sizes

Amino acid Information	Change in amino acid window size		
	From 0 to 3	From 3 to 6	From 6 to 10
Minimal	0.16%	33.97%	39.43%
Simple	0.16%	2.68%	22.52%
Complete	0.54%	89.28%	91.94%

Table B-14 *P-values* for change in results, because increased window size, for SCOP *All*, PEA 30%

Change in amino acid information	Amino acid window size			
	0	3	6	10
Baseline to Minimal	6.72%	0.10%	0.19%	0.18%
Minimal to Simple	40.28%	1.13%	0.07%	0.14%
Simple to Complete	23.73%	7.96%	37.14%	79.72%

Table B-15 *P-values* for change in results, because increased amino acid information, for SCOP *All*, PEA 30%

Appendix C Reconstructing the work

In this appendix we briefly describe the software that is distributed with this thesis and that allows the reconstruction of whole the work described. The minimum recommended hardware requirements for a computer to rebuild this work are 1 gigabyte of RAM, 1 Ghz CPU and 160 gigabytes of hard disk. The operating system where the work was developed was Windows XP but it should also work without problems on Windows 2000 and Windows Vista.

In the DVD distributed with this thesis, we provide:

- This thesis in Word and PDF format (requires Word 2003 or PDF reader)
- One gigabyte of compressed pdb files (all new pdbs between February 2005 and June 2006, requires RAR) (in order the files could fit into a single DVD only this small portion of our PDB mirror is distributed)
- Batch scripts to load PDB files to the *ProteinsDB* database
- DSSP program
- SCOP 1.67 and 1.69 text files
- Data Mining Table Creator binary and sources (to compile the source code, Visual Studio 2005 is required, to execute .Net framework 2.0 required)
- Batch scripts used for data mining automatization
- Data mining stream (requires Clementine 9)
- Literature papers and other thesis where available (requires PDF reader)
- Analysis of Mining results in Excel (requires Excel 2003)
- MDF database file and log (requires SQL Server 2005) (we removed the *pdb_data* table so it could fit in a single DVD)

More information available on the author's website: http://ctp.di.fct.unl.pt/~jcas

Glossary

Asymetric Unit: The smallest portion of a crystal structure to which crystallographic symmetry operations (e.g. : rotations, translations) can be applied to generate a unit cell.

Baseline: A simple amino acid residue burial status classifier (see page 50).

Biological Unit: The macromolecule that has been shown to be or is believed to be functional.

Clementine: Commercial Data Mining Software from SPSS.

CRISP-DM: Cross Industry Standard Process for Data Mining.

CSV: Comma separated file.

DSSP: Definition of Secondary Structure of Proteins.

PDB File: Protein Data Bank file.

Regular Chain: A chain which belongs to a unique scop domain.

SCOP: Structural Classification of Proteins.

Unit cell: The component that is stacked multiple times to generate the entire crystal.

Index