Strategy Shaper

Susmita Chatterjee
Project Report CS 470
April 26, 2004

- # *Table of Contents*

Preface

This project could not have been accomplished without the generous support and guidance from the quantitative team of McKinley Capital Management, Inc. I'd also like to thank current and past UAA faculty for their support. In particular, I would like to single out Professor Mock's CS 405[1], Professor Scott's CS 360[2] and former Professor Gordon's CS 202[3] as being particularly valuable.

0.1 Overview:

The purpose of this report is to provide an overview of the technology behind this factor selection application for stock selection modeling and discuss the various stages of its development cycle. This report is divided into five Chapters:

- Chapter 1: This chapter discusses the background and motivation behind my CS470 topic.
- Chapter 2: Describes the algorithms and design
- Chapter 3: Describes the Implementation of the application
- Chapter 4: Covers the prototyping and testing
- Chapter 5: Briefly outlines how the tool can be used and the further enhancements of the tool that are under consideration.

[1] Artificial Intelligence
[2] Database Systems
[3] Data Structure & Algorithms

Chapter 1: Background

This chapter discusses the background and motivation behind my CS470 topic. It also explains the challenges faced by quantitative investment managers. Their difficulties are caused due to the limitations of tool sets available in money management industry. I will briefly explain why it is difficult to have a perfect tool that solves the problems in particular with this industry. Finally, I presented the technology behind the application that I developed for my project. The advantages this application could offer when applied to investment management are also explained and last but most important the limitations of this tool.

1.1 The Topic:

One of the most common approaches to quantitative investment management is to reduce the universe of stocks to a manageable set of stocks that confirm to a group of desired characteristics. The industry experts are constantly developing and testing tools that can give them an advantage in selecting stocks with characteristics that can produce favorable results.

My CS470 topic is another effort to solve this problem by applying the development in artificial intelligence to the same old problem of stock selection. This report documents how this task was completed.

1.2 Background:

Although, most investment managers stop short of a pure quantitative process invoking optimization and higher mathematics, they always use some form of quantitative methodology to select stocks. The most common methods uses for this purpose are described in the next section.

1.2.1 Goal:

The goal of my project was to selection of best factors for superior stock selection models.

1.2.2 Solution:

The solution I came up with is a web-based application where you can upload data on factors. It will analyzed the data based on ID3 algorithm and will provide the best factors for better stock selection.

1.3 Current Methods:

1.3.1 Screening:

The screening methodology selects stocks based on constrains on preferred characteristics of the investment manager. The investor generally also decides the ranges, based on the investor's experience and investment style. Screening is helpful. It is, however, by no means a complete or strictly scientific process.

1.3.2 Multi-Factor Approach:

It is the next most popular method where it weight factors and arrive at a summary score for each stock, which is the used to select a group of stocks for investment.

In both systems of stock selection, some of the stocks get excluded from consideration based on one criterion while meeting many other criteria because of the factor constrains or unusual weight on that factor. The other important aspect of the process is that the characteristics are not linearly distributed, so not all the ranges of the factor are affective. It is certain pockets of the factor ranges that are effective.

1.4 Build a Decision Tree:

Decision tree learning algorithm has been successfully used in expert systems in capturing knowledge. The main task performed in these systems is using inductive methods to the given values of attributes of an unknown object. It determines appropriate classification according to decision tree rules. I examine the decision tree-learning algorithm ID3 and implement this algorithm using the .Net framework. I first implement basic ID3 in which I dealt with the target function that has discrete output values. I also extend the domain of ID3 to real-valued output, such as numeric data and discrete outcome rather than simply Boolean value. The application provided at last section offers a simulation of decision-tree learning algorithm in various situations.

1.4.1 What is Decision Tree?

What is decision tree: A decision tree is a tree in which each branch node represents a choice between a number of alternatives, and each leaf node represents a decision.

Decision tree is commonly used for gaining information for the purpose of decision-making. Decision tree starts with a root node on which it is for users to take actions. From this node, users split each node recursively according to decision tree learning algorithm. The final result is a decision tree in which each branch represents a possible scenario of decision and its outcome.

1.4.2 What is decision tree learning algorithm?

Decision tree learning is a method for approximating discrete-valued target functions, in which a decision tree represents the learned function. Decision tree learning is one of the most widely used and practical methods for inductive inference.
Decision trees classify instances by traverse from root node to leaf node. I start from root node of decision tree, testing the attribute specified by this node, and then moving down the tree branch according to the attribute value in the given set. This process is the repeated at the sub-tree level.

The target function has discrete output values. It can easily deal with instance which is assigned to a Boolean decision, such as 'true' and 'false', ‘p (positive)' and ‘n (negative)'. Although it is possible to extend target to real valued outputs, I will cover the issue in the later part of this report.

1.4.3 Why is Decision Tree learning an attractive Inductive learning method?

Purely inductive learning methods formulate general hypotheses by finding empirical regularities over the training examples.
For inductive learning, decision tree learning is attractive for three main reasons:
1. Decision tree is a good generalization for unobserved instance, only if the instances are described in terms of features that are correlated with the target concept.
2. The methods are efficient in computation that is proportional to the number of observed training instances.
3. The resulting decision tree provides a representation of the concept that appeal to human because it renders the classification process self-evident.

1.5 Decision Tree Learning Algorithm:

1.5.1 ID3 Basic:

ID3 is a simple decision tree-learning algorithm developed by Dr. J. Ross Quinlan (1983). The basic idea of ID3 algorithm is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node. In order to select the attribute that is most useful for classifying a given sets, I introduce a metric – information gain.
To find an optimal way to classify a learning set, what I need to do is to minimize the questions asked (i.e. minimizing the depth of the tree). Thus, I need some function, which can measure which questions provide the most balanced splitting. The information gain metric is such a function.

1.5.2 Entropy – measuring homogeneity of learning set:

In order to define information gain precisely, I will like to discuss entropy first.

First, lets assume, without loss of generality, that the resulting decision tree classifies instances into two categories, I'll call them P (positive) and N (negative).
Given a set S, containing these positive and negative targets, the entropy of S related to this Boolean classification is:
Entropy (S) = - P (positive) log2P (positive) - P (negative) log2P (negative)
P (positive): proportion of positive examples in S
P (negative): proportion of negative examples in S

For example, if S is (0.5+, 0.5-) then Entropy (S) is 1, if S is (0.67+, 0.33-) then Entropy (S) is 0.92, if P is (1+, 0 -) then Entropy (S) is 0. Note that the more uniform is the probability distribution, the greater is its information.
You may notice that entropy is a measure of the impurity in a collection of training sets. But how it is related to the optimization of my decision making in classifying the instances? What you will see at the following will answer this question.

1.5.3 Information Gain – measuring the expected reduction in Entropy:

As I mentioned before, to minimize the decision tree depth, when one traverse the tree path, he/she need to select the optimal attribute for splitting the tree node, which can be easily implemented with the attribute with the most entropy reduction.
I have defined information gain as the expected reduction of entropy related to specified attribute when splitting a decision tree node.
The information gain, Gain (S, A) of an attribute A,
Gain (S, A) = Entropy (S) - Sum for v from 1 to n of (|Sv|/|S|) * Entropy (Sv)
I can use this notion of gain to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root. In my project, I however modified this concept and ranked the factor groups based on entropy values, the lower the entropy value the better the factor. The depth of the decision tree is a function of number of cumulative stocks that are available in those groups sorted from lower to higher entropy values.

1.5.4 How I implement the ID3 Algorithm here:

ID3 (Learning Sets S, Attributes Sets A, Attributesvalues V)
Return Decision Tree
Begin:
Load learning sets first, create decision tree root node 'rootNode', add learning set S into root node as its subset.
For rootNode,
then

compute Entropy(rootNode.subset)
first If Entropy(rootNode.subset)==0,
then
rootNode.subset consists of records all with the same value for the categorical attribute,
return a leaf node with **decision attribute:attribute value;**
If Entropy(rootNode.subset)!=0,
then compute information gain for each attribute left(have not been used in splitting), find attribute A with Maximum(Gain(S,A)).
Create child nodes of this rootNode and add to rootNode in the decision tree.
For each child of the rootNode, apply ID3 (S,A,V) recursively until reach node that has entropy=0 or reach leaf node.
End ID3.

An advantage of this method is that this form of representation of rules is intuitive and easily understood by the human. However, the issue of determination of significance of a found rule becomes a very serious problem for the decision trees approach. The problem originates from the fact that there are a smaller number of records left at each next level of the classification tree that is being built. The tree is splitting data into a large number of small sets of specific cases. The larger is the number of different cases, the smaller is each next separate set of training examples, and there is less confidence that correct further classification can be performed. If the built tree is rather "bushy": if it contains a large number of small branches, then such a tree does not provide a meaningful and statistically justified solution. Application to complex real world problems shows that in majority of systems utilizing the decision trees method this problem does not find a satisfactory solution so far.

Chapter 2: Design

During the design phase, initial goals were identified, based on the goals; I developed the requirements for the product. And based on the requirements, I developed the system design. This requirements and design was formally reviewed in a meeting, changes were made, and the design document was accepted. The project was appointed with relatively few restrictions. The freedom granted allotted independent research on new technologies. As development environment, I choose to use ASP for my web form development and the .NET framework for the development environment. I also choose SQL2000 database as my backend database, and ADO for data transfer.

2.1 Reason for choosing the Technologies:

1) Fast Access of Data - Since ADO uses COM interfaces it provides a direct access to a table or a record-set without any hierarchy. If I would have used DAO or ODBC then I would have to go thorough a hierarchy to access a record-set or table.
2) The reason I choose ASP is because in ASP.net the script is run on the server side, thus no browser support for a scripting language is required, moreover the Server-side scripts cannot be readily copied because only the result of the script is returned to the browser.

2.2 Requirements for the product:

Although there was no precedence or specific design requirements for this application, I came with these requirements based on my conversations with the clients.

1. Menu with features to upload the dataset
2. A small window to present the view of the dataset
3. A command button to request generation of a decision tree
4. A window to view the resulting decision tree
5. Some mechanism to scroll through the decision tree
6. Some mechanism to print or save the results
7. A command button to quit or cancel the operation.
8. Finish the operation in a reasonable amount of in less than a minute.
9. Secured Login

2.3 Initial Design Goals:

Based on the above requirements and discussions with the client the following initial design goals were developed:

2.3.1 Compatibility and Modularity:

Modularity:

I organized all files under the windows 200 IIS directory. I made a directory called "Dtree" under 'wwwroot'. Beneath the "DTree" directory, I have subdirectories called 'ASP_files', 'Images', 'COM_DLLs'. The database will be maintained on a separate SQL 2000 Server.

2.3.2 Server Concurrency:

Both the Servers that I am using in our project IIS5.0 and SQL 2000 are concurrent servers.

2.3.3 Deadlock:

The issue of deadlock will be handled by the database.

2.3.4 Data Representation:

Data should be provided in CSV format, first row would be treated as column heading. And one of them should be a positive or negative performance in last one year. The other column of the data should be provided in the quintile value of factors.

The data for the generation of the decision tree is stored in the SQL server, so is the tables with the entire person details of the users, which is used to authenticate the users.

2.3.5 Performance and Speed:

Performance and speed is another important design goal, as the data and the applications are running from a web server, a quick production of the decision tree was a concern. Since production of the decision tree is data dependent so as an agreed upon check is established. For an example as if the time required to produce the tree is more than 1 minute for a data set with five factors and 150000 rows, then I have to optimize the code for more efficiency.

2.3.6 Adaptability:

Since the system is expected to be modified later, during and after field-testing, a flexible implementation was considered as an important issue.

2.3.7 System Architecture:

The system is being composed of pages. These pages can be thought of as objects with different properties. Each page serves specific purposes. Figure 2.3.1 shows a High-level abstraction of the System Architecture.

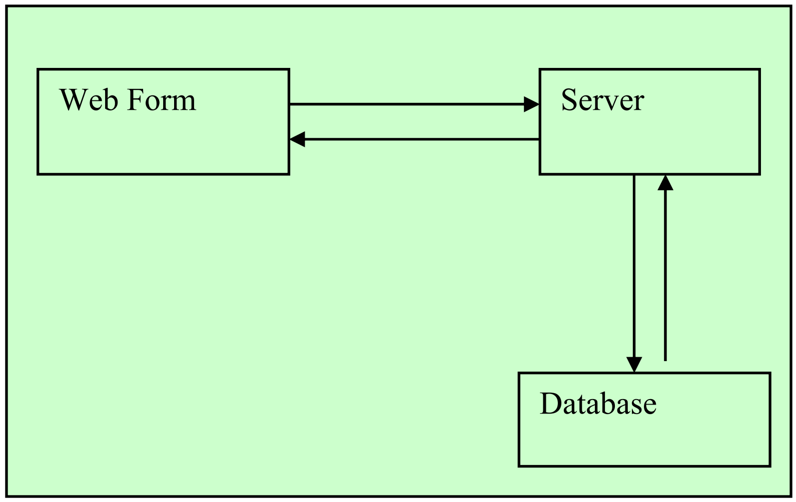

Figure 2.3.1 High-level abstraction of the System Architecture.

Upon authentication of the login; the user's personal information will be stored systematically on the database.
Figure 2.3.2 shows a High-level overview of the System Architecture. It shows that after browsing the main page user have to sigh in to use the application and then after loading their own data they can see the result. The result will be displayed as a binary tree.

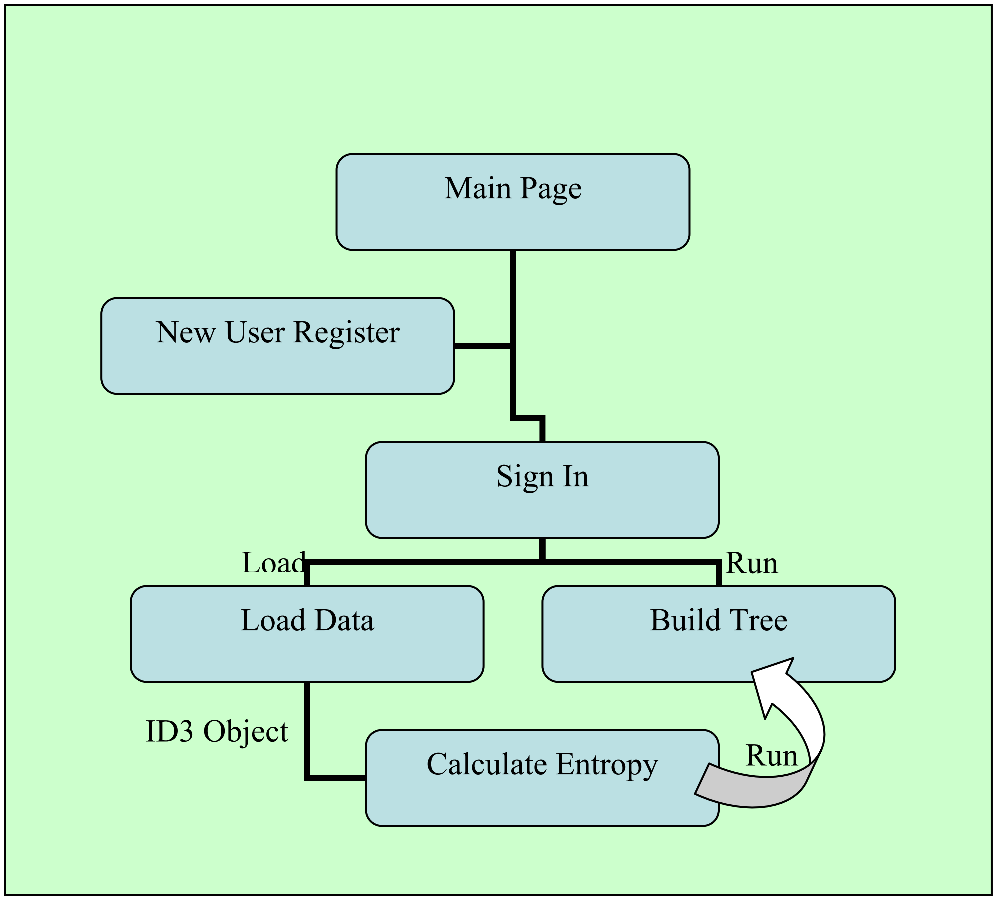

Figure 2.3.2 System Architecture.

Figure 2.3.3 shows the loin page in a browser. User will go to the "Sign In" page where the new users have to register and the existing user can login.

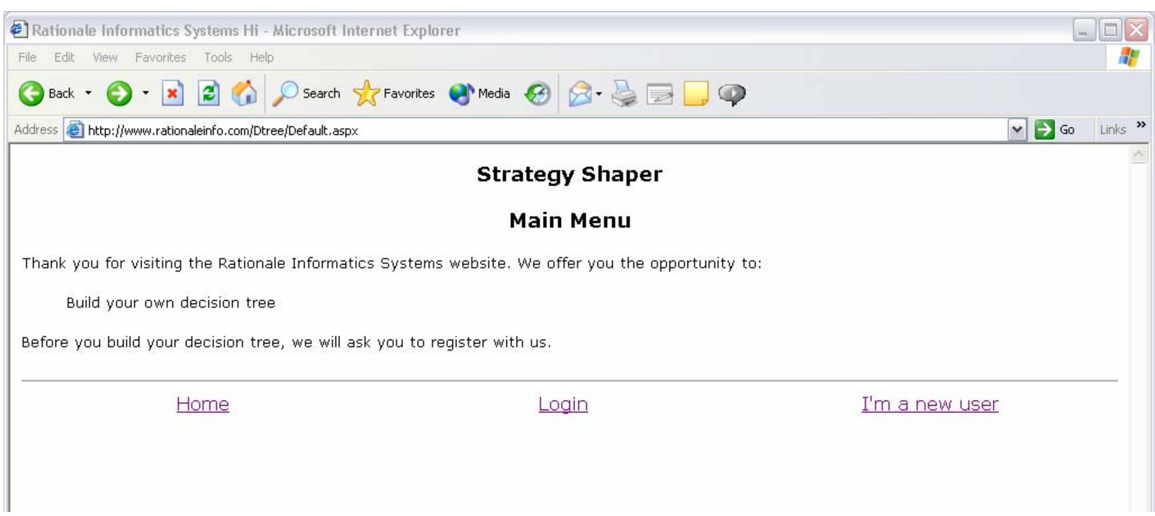

Figure 2.3.3 Login Page

In the application the connection between the web browser and the server is set up as shown in Figure 2.3.4:

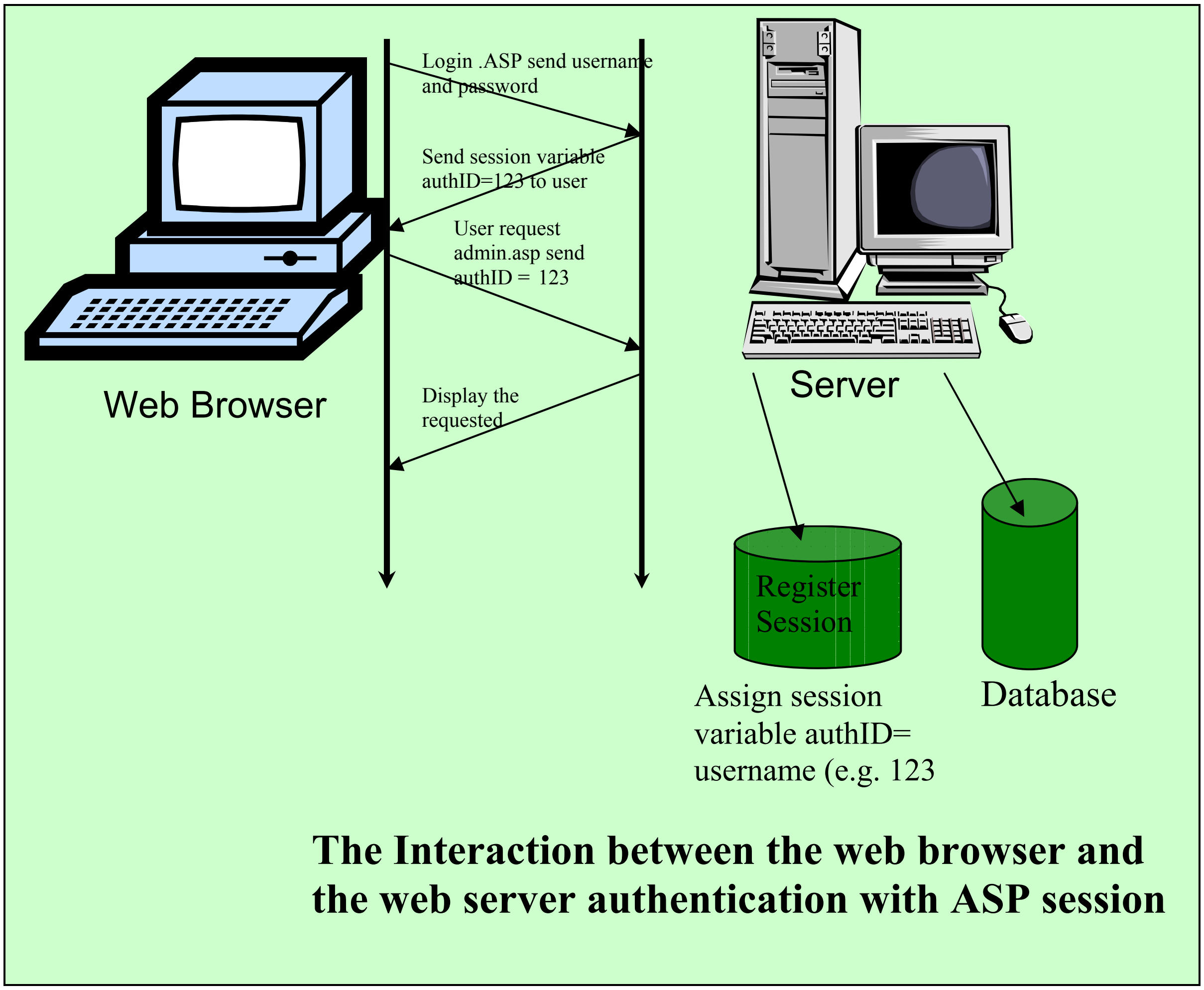

Figure 2.3.4 The connection between the Server and the Web Browser.

2.4 Algorithms:

The difficulty of this project did not reside in algorithms but instead with the design and implementation. Though, the algorithms had to solve several major issues. One of which was the, the algorithms needed to be designed to calculate the entropy value, should also use the benefits of the SQL server. The most important parts of the algorithm to this system were that of the server/database interaction and to set up the data files for calculating the entropy value of the factors.

Most of my algorithms involved SQL statements. All the dynamic ability of the web forms is based on the information stored in the database. Due to the nature of web programming, there were plenty of places for code re-use. I found myself repeatedly recycling source code on my web pages. Due to this action, the number of lines of source code on many web pages grew tremendously. Because my design was structural, I did not have any need for abstract data types and used only primitive data types native to .NET's programming environment. There were attempts late in the development cycle where code was dramatically condensed with the creation of generic functions that executed tasks that were repeatedly performed in many places. To generate and use these generic functions was time consuming but the source code became more reliable and reusable.

2.5 Data Files*:*

The data file for input is a single CSV file with six columns, the first column is the stock symbol; between the other five of four columns are the quintile values of the factors. The remaining one represents the relative performance of the stock with respect to a market index (in this case S&P500) for the period of one year. The positive and the negative values are represented as 0's (negative) and 1's (positive). Below is a screenshot as an example of data providing to the application.

```
Symbol,PSLTM,PCLTM,PE,EPSGR,ORS,PCHG
MMM,1,2,1,5,4,1
ABT,1,2,1,5,4,0
ACE,1,2,1,5,4,1
ADCT,1,2,1,5,4,1
ADBE,1,2,1,5,4,1
AMD,1,2,1,5,4,1
AES,1,2,1,5,4,1
AET,1,2,1,5,4,1
AFL,1,2,1,5,4,0
A,1,2,1,5,4,1
APD,1,2,1,5,4,0
ACV,1,2,1,5,4,0
ABS,1,2,1,5,4,0
AA,2,2,1,5,4,1
AYE,2,2,1,5,4,1
ATI,2,2,1,5,4,1
AGN,2,2,1,5,4,1
AW,2,2,1,5,4,1
ALL,2,2,1,5,4,0
AT,2,2,1,5,4,0
ALTR,2,2,1,5,4,1
MO,2,2,1,5,4,1
ABK,2,2,1,5,4,1
AHC,2,2,1,5,4,0
AEE,2,2,1,5,4,0
AEP,2,2,1,5,4,0
AXP,2,2,1,5,4,1
AM,2,2,1,5,4,1
AIG,2,2,1,5,4,0
APCC,2,2,1,5,4,1
```

The columns headings are described below:
-Symbol: Company represented stock name.

-PSLTM: Ratio of the month end Price of the company stock and the average reported monthly sales of the company represented by the stock.

PCLTM: Ratio of the month end Price of the company stock and the average reported monthly Cash Flow of the company represented by the stock.

PE: Ratio of the month end Price of the company stock and the average reported earnings of the company represented by the stock.

ORS: Ratio of the opportunity and risk associated with company represented by the stock.

PCHG: Relative Price Change of the stock and the benchmark during the last 12 months.

2.6 Preparing the Data files:

2.6.1 Data Upload

Upload the file to a SQL Server with the help of a parser that understand the file format.

2.6.2 SQL Advantage

By uploading the data to a SQL server I get the advantage of sorting and quicker calculations.
Figure 2.6.1 shows a .CSV file is uploading in the Server.

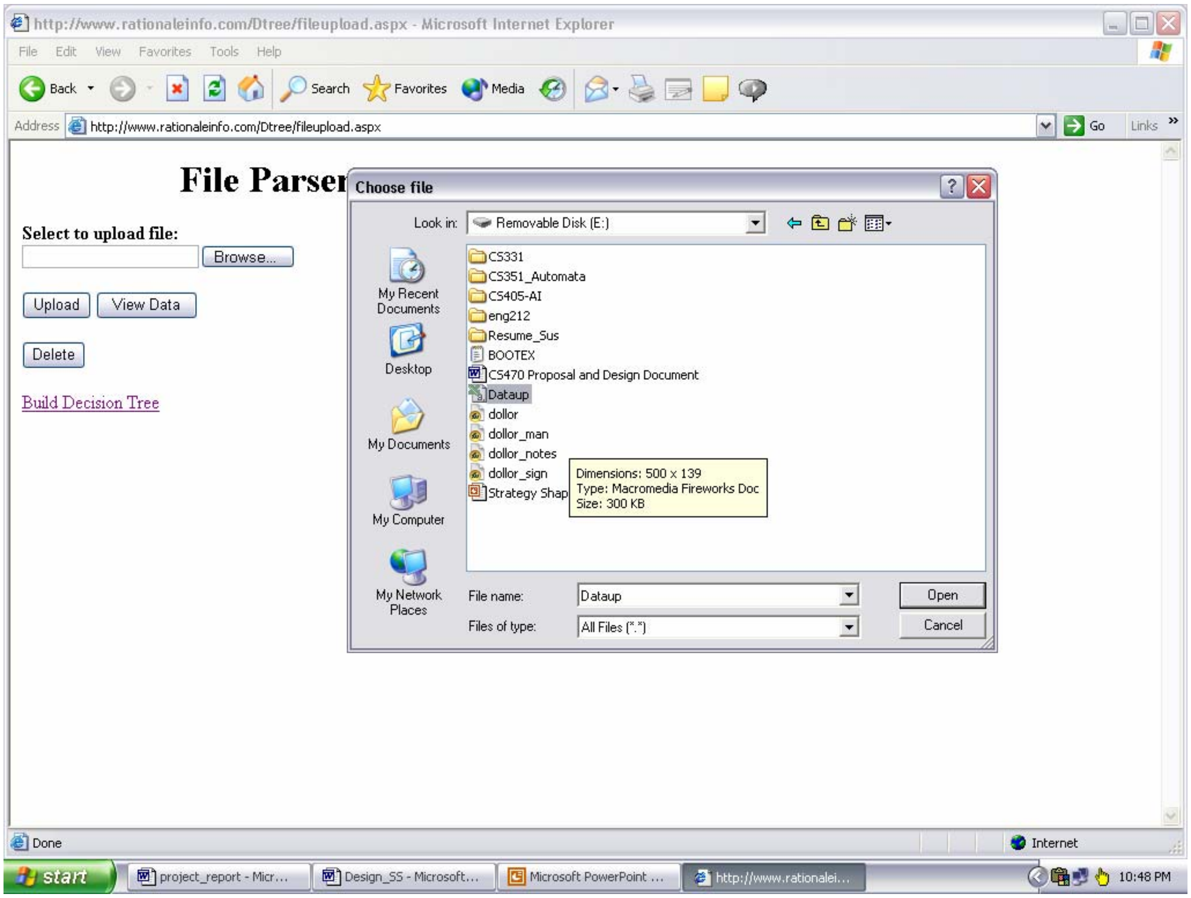

Figure 2.6.1 Uploading data on Server.

After uploading the data on the SQL Server user can also view the data if they want to see the data they have uploaded. The users also have the option to "Delete" the data they have uploaded to the server.
Figure 2.62 shows that the data user uploaded in the browser are deleted.

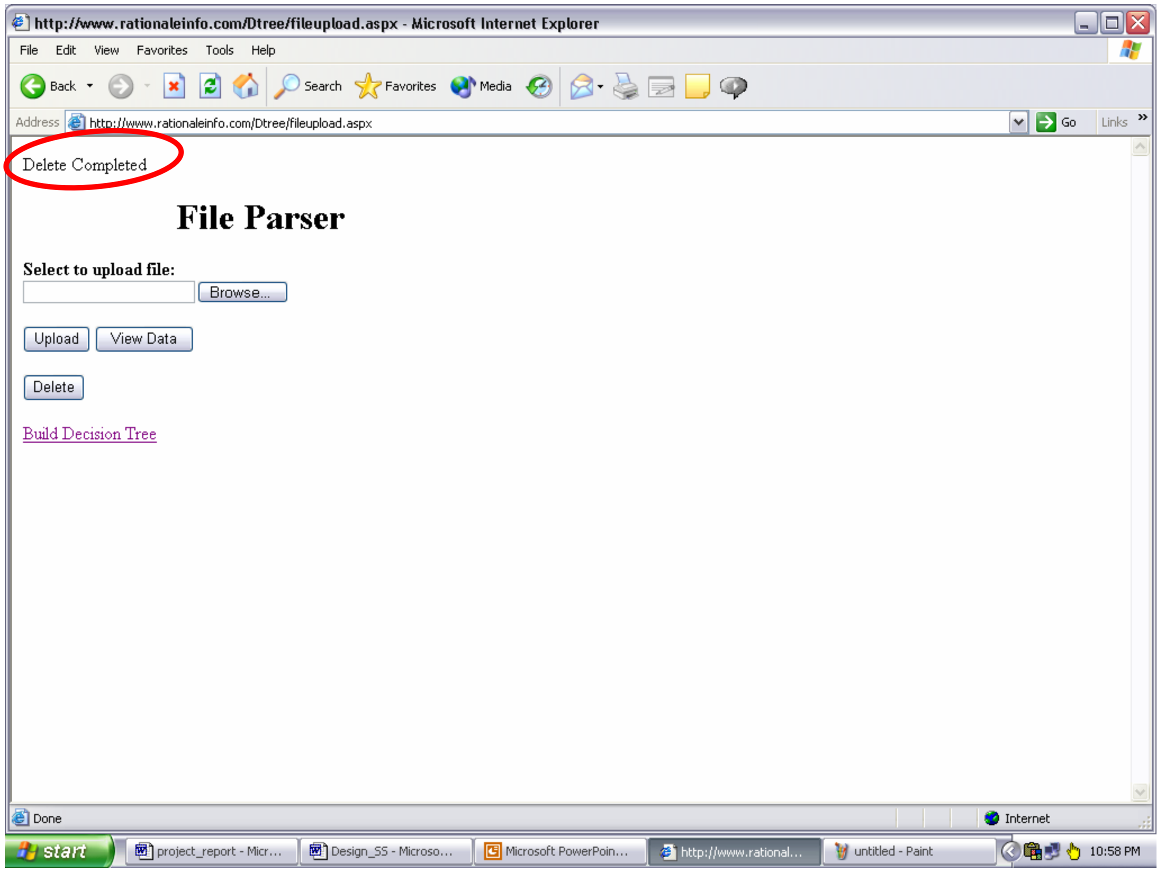

Figure 2.6.2 Data deleted from Server.

By clicking the hyperlink given in the bottom of the page user will go to the page where the decision tree can be generated.
Figure 2.6.3 shows a resulting tree-view of the application.

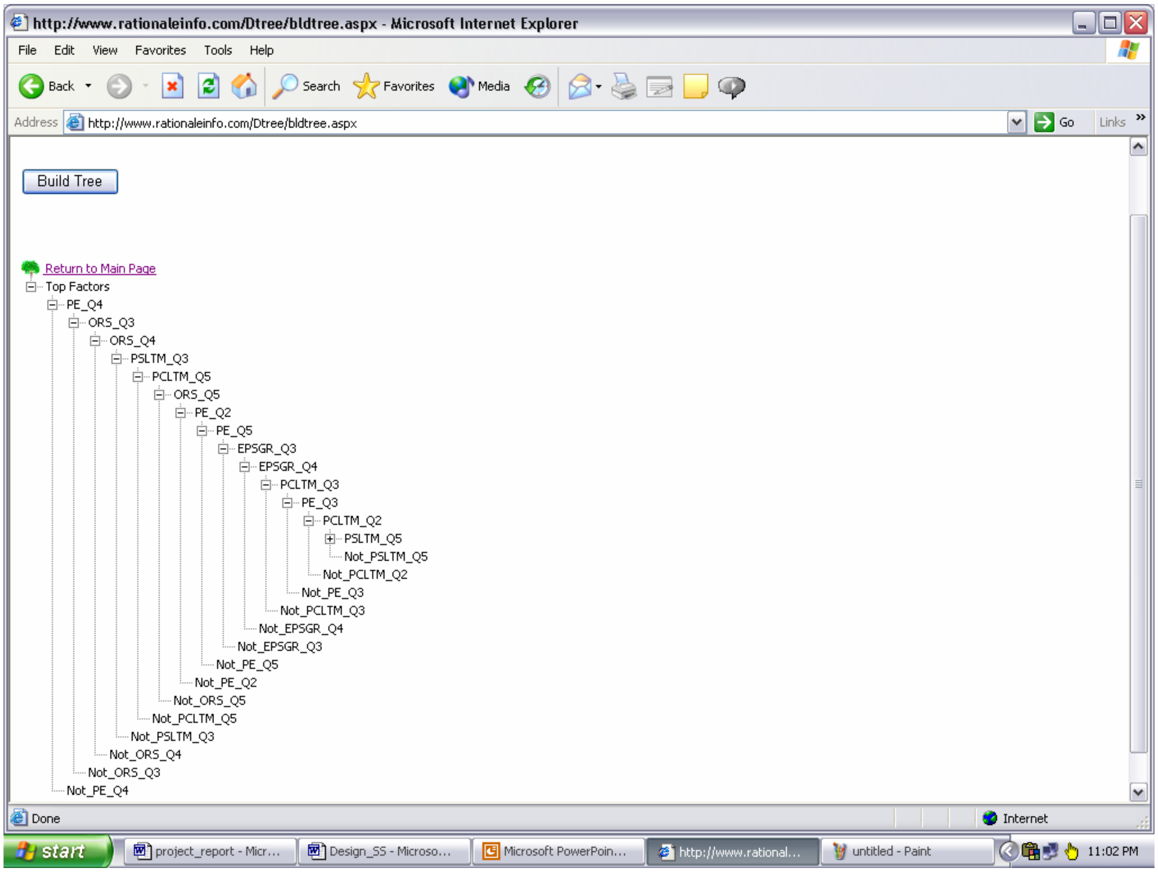

Figure 2.6.3 Resulting Tree-view.

Chapter 3: Implementation

Implementation followed the plan developed during the design phase. Feature interdependencies dictated that the feature implementation order roughly followed the task breakdown given in design section of this report:

- The first task was development of database and forms, the parser that run at the web server for uploading data from the client machine to SQL server on the web.
- The second task was to develop the decision tree algorithm and a sorting algorithm similar to quick sort, which can work on a 2D array.
- The third task was to collect the results from the previous task and build the decision tree from those results and display it on the browser.

With few clear requirements for the application tool by my client, I based on my project implementation on prototyping and consulted with my clients on all aspects during the development phase.

3.1 Programming:

Early in the development cycle, I opted to use Microsoft's .NET Framework and had the tough task to learn three major computer development components: Microsoft's .NET Framework, ASP coding and third party vendor tools. The project was started from scratch and a hefty amount of hours of research were spent on trying to learn the workings of both these languages. I decided to use VB.net 2003 because of its popularity and my previous experience with Visual Basic 6.0. In contrast, ASP.NET was a completely new language for me. I had no prior experience in of web programming either; learning how to use some of ASP.NET's built-in features was, at times, a tough task but was very exciting.
The production of the decision tree from the data set involved few steps. After the data is loaded to the SQL server, I designed a query to collect the sorted quintile values of each field and their associated attribute of the classifier field which can have only two values -"yes" or "no" - which correspond to the decision, the total number of positive and negative associations are determined, and entropy values are calculated and stored in an array, this whole process is repeated for all the fields of the data set.
After the quintile values of the function and their associated entropy values are determined I sort the arrays where entropy values and their associated factor quintiles were stored, the sorting parameter was the entropy value.

I also decided to introduce one special field ("stock symbol") into table, although it was not used in any of the functionality of this application, it does help keeps track of which data set the given tuple of the relation belongs at any given time. Displaying the Binary Tree was another challenge for me, as the directory control that comes with the Visual Studio.net is designed primarily for window applications only. So I decided to test the third party vendors' components, I tested the components available from the following vendors, and my views for each of them are as below.

1. Component Studio: I tested the Component studio's utility called LineGraph-5M. It is a Web Custom Control that allows you to add Line and Scatter Plot graphs to your ASP.NET pages. It can create graphs with an absolute minimum of effort, yet contains enough options to allow you to customize its appearance and function to suit varied needs. Although it is a good utility but did not serve my purpose because it could not produce a tree or a multi series donut graphs which I could have used to represent a circular decision tree
2. Visual ASP: At first the Visual ASP's Tree View control looked like the perfect tool for building the tree view for my project, but after spending few good hours, I found out it is unusable because it has very poor documentation and have lots of bugs, my efforts to contact customer support also went in vein so I decided not to use this product.
3. Component One: My experience with Component One's Chart for .Net Control was similar to Component Studio; I did not have the right features that I can use in my project.
4. ASP TreeView: I found ASP TreeView is much faster and much smaller (40 KB) than all other 3rd party TreeViews. Yet it is easiest to use. It has many styles and the ability to load dynamically. It allows me to put any HTML tags and script in the nodes. I decided to use this in the project because it was able to do most of the things I needed and also because of it was available for free of cost.

Chapter 4: Testing

4.1 Testing and Debugging:

A large amount of time was spent on testing and debugging in my project. Some web pages I designed took a large amount of time to debug due to the use of third party vendor components on the page. There was no ideal method to perform this input validation so the large amount of time spent on testing and debugging those web pages was inevitable.

In the early stages of development, there were problems encountered with data upload to the database in an efficient manner. The time spend in preparing the parser took a significant portion of my project. The time spent on debugging these problems was, overall, very consuming also.

4.2 Requirements Challenges:

The lack of a formal requirements and use of prototyping presented numerous challenges. Though, I came to realize that the clients did not approve of this format and wanted everything to be done on the same page. I made some adjustments to the upload form, once again, for allowing the user to reside on the same page after each workload component submission. The page was shown to the clients and once again clients came up with more suggestions.

4.3 Work Breakdown:

The Gantt chart in figure 4.3.1 shows the anticipated and actual work schedule. The values in green represent the projected amount of time per task determined during the planning stage of the project, and the values in blue represent the text. The project is finished in time, so the total amount of actual time is same as the projected schedule.

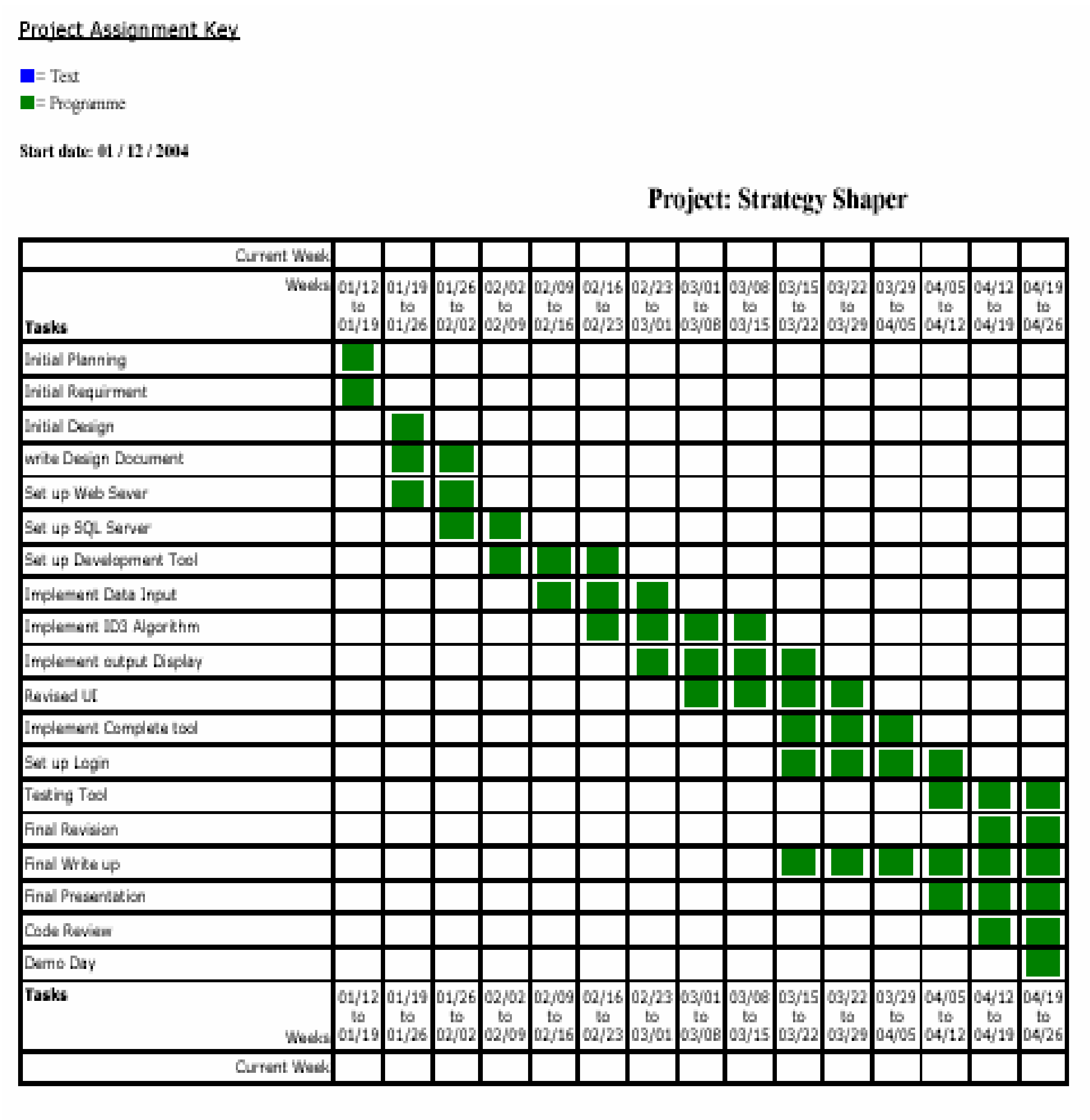

Figure 4.3.1 Gantt chart

Chapter 5: Conclusion

5.1 Future Steps:

The Strategy Shaper application will need more development and testing before it is widely used. The first version will certainly prove to some extend how much advantage the process has over existing ways of stocks selection. To get actual feel, my clients have agreed to maintain a paper portfolio of stocks selected by this procedure.
Once I have evidence that this can add significant advantage over the other processes, then I will be certainly interested to implement a version that is more user friendly and with some more advanced features on display side of the application, like just by clicking at a node of the tree, you get a list of the stocks that are included in that group or may be by right clicking you get to see the summary statistics of the stocks for the factors used in the analysis.

5.2 Summary and Conclusions:

Implementing a concept learned in my AI class in the real application was very helpful. Although the method used in this project can be applied for solution of classification tasks only, it was a perfect solution for my client's application where the most common problem is the task of predicting values of some numerical variable.

Forecast of dynamics of a market - the system, which corrects its behavior depending on its own history - is probably one of the most difficult existing tasks. Thus it is considered as a challenge to the might of mathematics and modern computer technology. On the other hand, even a small improvement in portfolio management efficiency against the market average can result in very substantial perks.

In this class, I learned what it takes to put a complete application in front of a client. It also helped me learn some new tools and last but not least it gave me the understanding of the software development process.

I am pleased to know that the results of my efforts will be put in use by my clients. Before, I finish my report I want to thank my clients for their support and understanding. I also want to thank my advisor professor Kenrick Mock for his support and suggestions throughout this project.

References

1. Rockford Lhotka. Visual Basic .Net Business Objects. Wrox Press, 2003.
2. Scott Mitchell, Steve Walther, Doug Seven, Donny Mack,Chris Payne, Billy Anders, Adam Nathan, Dan Wahlin. ASP.NET: Tips, Tutorials, and Code. Sams Publishing, 2003
3. Louis Davidson. Professional SQL Server 2000 Database Design. Wrox Press, 2003.
4. Mock, Kenrick. "Artificial Intelligence". CS 405. 2003. http://www.math.uaa.alaska.edu/~afkjm/cs405/
5. Luger, George F. Artificial Intelligence, Structures and Strategies for Complex Problem Solving, 4th edition. Addison-Wesley, 2000.