Resource management on Cloud systems with Machine Learning

Master thesis

Zhenyu Fang

fangzhenyufzy@gmail.com

Master in Information Technology

Barcelona School of Informatics

Technical University of Catalonia

Advisor: Ricard Gavaldà Mestre

Co-Advisor: Jordi Torres Viñals

July, 2010

Abstract

Machine Learning techniques based on Weka are adopted to build a middleware platform called "SysWeka", which extends Weka capabilities and provide a software interface for usage by higher application for managing resources on cloud systems. This work is based on Javier Alonso's and Josep Lluís Berral's doctoral theses works. In this work, three different machine learning methodologies are employed to make classifications and predictions from source datasets; these predictive results can be used in distributed decision systems. Particularly, the confidence prediction and the study about Importance-Aware Linear Regression involve innovative application usage and a promising research. The experimental evaluation platform offers here contains detailed performance estimation and evaluation of referred methods. It is expected that this framework provides a fast and easy approach to build applications based on Machine Learning.

Contents

7 Appendix

Chapter 1

Introduction

1.1 Project background

With progressive spotlight on cloud computing as a possible solution for a flexible, on-demand computing infrastructure for lots of applications, many companies and unions have joined the tendency. Obviously, cloud computing have been recognized as a model in support of services. Within that cloud system, massive distributed data center infrastructure, virtualized physical resources, virtualized middleware platform as well as applications are all being provided and consumed as services.

Since large numbers of data processed and the energy resources cost generated have become a major economical and environmental factor, Green IT [1] has been put forward as a solution to lessen IT departments' cost. Therefore, it is crucial to obtain a rational prediction from this complicated system in order to achieve better management.

To make the data center more economical and reduce the environmental impact, framework that can highly optimize energy efficiency has been proposed by Josep Llu ís Berral [2], from Technical University of Catalonia, where a framework was proposed that provides an intelligent consolidation methodology using different techniques including Machine Learning.

On the other hand, the growing complex of modern computer systems has lead to increasingly software faults. Just like the operation systems, application platforms have become more functional, extensible, complicated and even interact with each other, which greatly increase the software-level errors. Therefore, plenty of techniques have been researched and developed to avoid software failures. General bug-fixed mode cannot catch the pace of modern on-demand system, Machine Learning, model construction and prediction strategy are now proposed to address this tough problem.

As Javier Alonso's works described in [3] [4], adaptive software aging prediction based on Machine Learning is proposed in [3]. A series system metrics are used to predict software aging time, which is an important foundation for this project.

1.2 Project motivation

Machine Learning contains massive advantageous methods to make classification and prediction. Weka is a data mining and Machine Learning tools written in Java that involves API interface and easy extensibility. This tool is appropriate for common experiments and testing manually. However, our goal is to do prediction automatically and more general for the energy efficiency and software failures like scenarios.

A platform should be constructed to meet these requirements and be extensible as well. Therefore, further applications using machine learning can just interact with this platform and call the functions directly without operating from raw data. This can provide more general view from application layer and hide specific machine learning algorithms, which improves application efficiency and make it easy to start.

1.3 Project objectives

This project is focusing on building a software platform for making developments in cloud computer systems to achieve decision-making prediction, which utilizes diverse Machine Learning techniques and provides a software interface for prediction operation.

This work is based on the idea proposed by Josep Lluís Berral [2] and Javier Alonso [3], for this project, Machine Learning is the key point to enhance the prediction accuracy and construct the components for flexible and general usage. Also data mining and mathematical methodologies are applied to gather information, implement and promote the core functionality.

The objective of this work is to offer an extensible middleware that uses different Machine Learning techniques to provide functionality of building models, prediction, evaluation and performance analysis based on proposed framework. Those Cloud applications can work with these interfaces without even realizing certain low level infrastructure details, which causes high transparence and convenience for massive Cloud application development.

The middleware is designed and built between the lower Machine Learning infrastructures and higher cloud applications, where prediction accuracy, extensible functionality, confidence prediction and evaluation measurements are developed to construct the extension of the existing standard. Data gathered from emulator [3] is regarded as input knowledge and will be modeled with different Machine Learning algorithms including Linear Regression, M5P and Bayesian networks. Predicting instances to obtain predictive results and confidence after modeling data can acquire prediction of future states which helps to make wise decision for the higher cloud applications.

Within this middleware, Weka is adopted as a communication interface to the original dataset and offers well-defined API to manipulate building models and classification.

1.4 Project Environment

This project is developed in the framework of a multidisciplinary effort started approximately three years ago by researchers at the Computer Architecture Department (DAC) of UPC, the Software department (LSI) of UPC, and the Barcelona Supercomputing Center (BSC). One of the advisors of this thesis, Professor Jordi Torres, belongs to the High Performance Computing Group of DAC, and manager for Autonomic Systems and eBusiness Platforms research line in BSC. The other advisor of this thesis, Professor Ricard Gavaldà, belongs and currently coordinates the LARCA research group of UPC, whose main line of research is the theory and applications of machine learning and data mining. Recently, both teams have investigated the role of machine learning and data mining to build self-managing systems, with emphasis on achieving efficiency without compromising performance.

As part of this research effort, there are two ongoing Ph.D. theses co-advised by professors Torres and Gavaldà. The Ph.D. thesis of Javier Alonso, to be defended in a few months, deals with ways of achieving high-availability in cluster systems, and in particular of mitigating the effects of software aging in web servers; to this end, it is essential to be able to predict the effect of software aging and oncoming machine crashes, for which machine learning techniques are particularly applicable. The Ph.D. thesis of Josep Lluís Berral, currently taking shape, will deal with the efficient scheduling of workloads and resource allocation in virtualized cloud environments; there, it is crucial to be able to predict the variation and resource consumption of incoming workloads, as well as the effect of workload variation and resource allocation on the performance of both virtual and physical machines.

The goals of this Master thesis can be viewed as providing some bridge between existing machine learning frameworks (specifically, Weka) and the specific, not totally standard, machine learning requirements of these Ph.D. Thesis and the ongoing research project.

1.5 Document Organization

The document is structured as follows: In chapter 2, the state of the art in Resource management on Cloud systems with machine learning is shown. In chapter 3, three main Machine Learning methodologies are explained. Chapter 4 describes the work in this project, SysWeka Platform. Chapter 5 illustrates experimental evaluation on this platform in detail. In chapter 6, some conclusions and expectation on future work are presented. Finally, chapter 7 as Appendix involves numerous experimental results, model structures and datasets representation.

Chapter 2

State of the Art

This chapter explains the state of the art of Cloud systems, data mining and resource management with Machine Learning, from the whole complex systems to key technologies in resource management.

2.1 Cloud systems

Cloud Computing has become one of the popular buzzwords in the IT area after Web2.0. This is not a new technology, but the concept that binds different existed technologies altogether including Grid Computing, Utility Computing, distributed system, virtualization and other mature technique. As a key service delivery platform, Cloud computing systems provide environments to enable resource sharing in terms of scalable infrastructures, middleware, application development platforms and value-added business applications. Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) are three basic service layer [5].

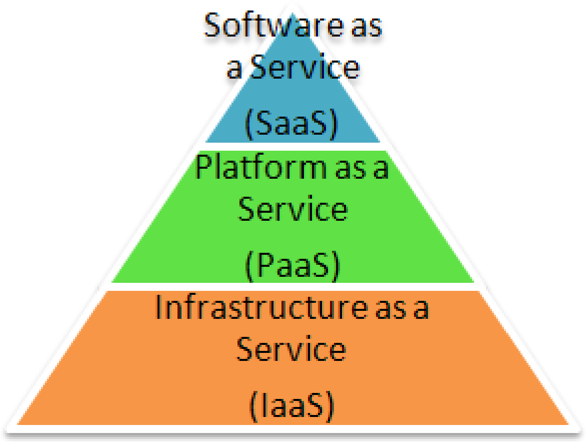

Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)

Figure 2.1: Cloud system architecture

- SaaS: This layer is very familiar to the web users that hosts applications and provides on-demand services to users. Applications delivered via the SaaS model benefit consumers by relieving them from installing and maintaining the software and they can be paid by resource usage or license models [5].

- PaaS: This is the layer in which we see application infrastructure emerge as a set of services and support applications. In order to achieve the scalability required within a cloud, the different services offered here are often virtualized. Examples of offerings in this part of the cloud include IBM WebSphere Application Server virtual images, Amazon Web Services, Boomi, Cast Iron, and Google App Engine. Platform services enable consumers to be sure that their applications are equipped to meet the needs of users by providing application infrastructure based on demand [5].

- IaaS: The bottom layer of the cloud is the infrastructure services layer. Here a set of physical assets such as servers, network devices and storage disks offered as provisioned services to consumers. Examples of infrastructure services include IBM BlueHouse, VMWare, Amazon EC2, Microsoft Azure Platform, Sun ParaScale Cloud Storage, and more. Infrastructure services address the problem of properly equipping data centers by assuring computing power when needed. In addition, due to the fact that virtualization techniques are commonly employed in this layer, cost savings brought about by more efficient resource utilization can be realized [5].

According to [6], typically there are four types of resources that can be provisioned and consumed over the Internet. They can be shared among users by leveraging economy of scale. Provisioning is a way of sharing resources with requesters over the network. One of the major objectives of Cloud Computing is to leverage Internet or Intranet to provision resources to users.

- Infrastructure resources contain computing power, storage and physical machine and networks provision. For instance, Amazon EC2 provides web service interface to easily request and configure capacity online [7].
- Software resources include middleware and development resources. The middleware consists of cloud-centric operating systems, application servers, databases and others. The development resources comprehend design platforms, development, testing, and deployment tools.
- Application resources mean that various applications have been moved into cloud environment and delivered as a service known as SaaS as explained above. For example, Google has adopted the Cloud Computing platform to offer many Web-based applications for business and personal usage [8].
- Business Process is a set of coordinated tasks and activities, which represents certain business service shown as a workflow. Business Process Management tools integrated in cloud systems can reuse, compose and communicate with these processes.

2.2 Data mining

Data mining is a practical technology to analyze and extract patterns from raw data, which can transform the original data into knowledge and beneficial information. The idea is to build computer programs that sift through databases automatically, seeking regularities or patterns. Strong patterns, if found, will likely generalize to make accurate predictions on future data.

In data mining, the data is stored electronically and the search is automated or at least augmented by computer. It has been estimated that the amount of data stored in the world’s databases doubles every 20 months. As the flood of data swells and machines that can undertake the searching become commonplace, the opportunities for data mining increase. As the world grows in complexity, overwhelming us with the data it generates, data mining becomes our only hope for elucidating the patterns that underlie it. Intelligently analyzed data is a valuable resource. It can lead to new insights and, in commercial settings, to competitive advantages [11].

There have been some efforts to define standards for data mining, for example the 1999 European Cross Industry Standard Process for Data Mining (CRISP-DM 1.0) and the 2004 Java Data Mining standard (JDM 1.0). These are evolving standards; later versions of these standards are under development. Independent of these standardization efforts, freely available open-source software systems like the R Project, Weka, KNIME, RapidMiner and others have become an informal standard for defining data-mining processes. The first three of these systems are able to import and export models in PMML (Predictive Model Markup Language) which provides a standard way to represent data mining models so that these can be shared between different data mining applications [9].

In this project, Weka (Waikato Environment for Knowledge Analysis) is used to perform data mining and Machine Learning functions, which is a popular suite of Machine Learning software written in Java, developed at the University of Waikato, New Zealand [10]. The Weka workbench contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality. Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes.

2.3 Resource management with machine learning

Machine Learning is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A major focus of Machine Learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too complex to describe generally in programming languages, so that in effect programs must automatically describe programs.

There are two main types of Machine Learning algorithms. In this work, supervised learning is adopted here to build models from raw data and perform regression and classification.

- Supervised learning: It deduces a function from training data that maps inputs to the expected outcomes. The output of the function can be a predicted continuous value (called regression), or a predicted class label from a discrete set for the input object (called classification). The goal of the supervised learner is to predict the value of the function for any valid input object from a number of training examples. The most widely used classifiers are the Neural Network (Multilayer perceptron), Support Vector Machines, k-nearest neighbor algorithm, Regression Analysis, Bayesian statistics and Decision tree.
- Unsupervised learning: It determines how the inputs are formed like clustering where learner is given unlabeled examples. Unsupervised learning is closely related to the problem of density estimation in statistics. However unsupervised learning also encompasses many other techniques that seek to summarize and explain key features of the data. Some forms of unsupervised learning is clustering, self-organizing map.

Chapter 3

Machine Learning techniques

In this chapter we describe some machine learning techniques, the ones that will be most relevant to our work. Many, many more techniques exist, and the ones we chose here are neither the most sophisticated, nor necessarily the ones that provide better accuracy in general. We chose two of them (linear regression and decision trees) because they were the ones mostly used in our reference works [2,3,4]; additionally, one of the goals of the thesis was to investigate the use of a third kind of techniques (Bayesian networks) in the context of computer resource management and prediction.

3.1 Linear Regression

In statistics, linear regression is a staple technique that works with numeric attributes. It is one method of linear models which model the relationship between a scalar variable *y* and one or more variables denoted *X* and these models depend linearly on the unknown parameters to be estimated from the data. Generally, linear regression could refer to a model in which the median or some other quantile of the conditional distribution of *y* given *X* is expressed as a linear function of *X*.

Linear regression has been used extensively in practical applications, because models which depend linearly on their unknown parameters are much easier to build than non-linear ones. Specifically, when the outcome or class and all attributes are numeric, linear regression is a natural method to consider.

The idea is to express the class as a linear combination of the attributes with predetermined weights:

$$y = w_0 + w_1a_1 + w_2a_2 + \dots + w_ka_k$$

where y is the class; a_1, $a_2 \dots a_k$ are the attribute values and $w_0, w_1 \dots w_k$ are weights.

The weights are calculated from the training data. Here the notation gets a little heavy, because this is a clear way of expression the attribute values for each training instance. The first instance will have a class, say $y^{(1)}$ and attribute values $a_1^{(1)}$, $a_2^{(2)}$,…, $a_k^{(1)}$, where the superscript denotes that it is the first example. Moreover, it is notationally convenient to assume an extra attribute a_0 whose value is always 1.

So the predicted value for the first instance's class can be written as:

$$w_0a_0^{(1)} + w_1a_1^{(1)} + w_2a_2^{(1)} + \ldots + w_ka_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(i)}$$

This is the predicted, not the actual, value for the first instance's class. The difference between the predicted and the actual values the interest one. The method of linear regression is to calculate the coefficients w_j, there are $k + 1$ of them, to minimize the sum of the square of these differences over all the training instances. Suppose there are n training instances; denote the ith one with a superscript (i). Then the sum of the squares of the differences is

$$\sum_{i=1}^{n} (y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2$$

where the expression inside the parentheses is the difference between the ith instance's actual class and its predicted class. This sum of squares is what we have to minimize by choosing the coefficients appropriately.

Numerous procedures have been developed for parameter estimation and inference in linear regression. These methods differ in computational simplicity of algorithms, presence of a closed-form solution, robustness with respect to heavy-tailed distributions, and theoretical assumptions needed to validate desirable statistical properties such as consistency and asymptotic efficiency.

Ordinary least squares (OLS) is the simplest and thus very common estimator. It is conceptually simple and computationally straightforward. OLS estimates are commonly used to analyze both experimental and observational data. This method minimizes the sum of squared residuals, and leads to a closed-form expression for the estimated value of the unknown parameter w.

Often those n equations are stacked together and written in vector form as:

$$Y = AW + \varepsilon \ ,$$

where

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ .. \\ y_n \end{bmatrix}, A = \begin{bmatrix} A'_1 \\ A'_2 \\ .. \\ A'_n \end{bmatrix} = \begin{bmatrix} 1 & a_1^1 & \cdots & a_k^1 \\ 1 & a_1^2 & \cdots & a_k^2 \\ 1 & \vdots & \cdots & \vdots \\ 1 & a_1^n & \cdots & a_k^n \end{bmatrix}, W = \begin{bmatrix} w_0 \\ w_1 \\ .. \\ w_k \end{bmatrix}, \ \varepsilon = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ .. \\ \varepsilon_n \end{bmatrix}$$

- y_i is called dependent variable which represent the real class value with instance i.
- The matrix A is called independent variables which shows each attribute value with each instance.
- W is a k +1 dimensional parameter vector. Its elements are called effects, or regression coefficients.

- ε_i is called error term or noise.

According to the OLS algorithm, the unknown parameter W can be calculated as:

$$W = (A'A)^{-1}A'Y$$

where ' denotes matrix transpose and $^{-1}$ is matrix inversion.

Linear regression is an excellent, simple method for numeric prediction, and it has been widely used in statistical applications for decades. Of course, linear models suffer from the disadvantage of, well, linearity. If the data exhibits a non-linear dependency, the best-fitting straight line will be found, where "best" is interpreted as the least mean-squared difference. This line may not fit very well. However, linear model serve well as building blocked for more complex learning methods.

3.2 Decision Tree and Model Tree

A "divide-and-conquer" approach to the problem of learning from a collection of independent instances leads naturally to a style of representation called a decision tree. Decision tree is one of the most popular classification algorithms in Data mining and Machine Learning, which is a tree-structured model of a set of attributes to test in order to predict the output. Decision tree learning is a methodology that uses inductive inference to approximate a target function, which will produce discrete values. It is widely used, robust to noisy data and considered a practical method for learning disjunctive expressions [11].

Nodes in a decision tree involve testing a particular attribute. Usually, the test at a node compares an attribute value with a constant. However, some trees compare two attributes with each other, or use some function of one or more attributes. Leaf nodes give a classification that applies to all instances that reach the leaf or a set of classifications, or a probability distribution over all possible classifications. To classify an unknown instance, it is routed down the tree according to the values of the attributes tested in successive nodes and when a leaf is reached, the instance is classified according to the class assigned to the leaf [11].

The structure of decision tree is shown below, which is a simple tree generated with Weka. This example predicts whether the weather is good enough to play outside. There are five nominal attributes in all (outlook, temperature, humidity, windy, play) and play is the class to be predicted. And the decision tree learning algorithm just selects four attributes including class to construct the tree with five leaves and eight nodes.

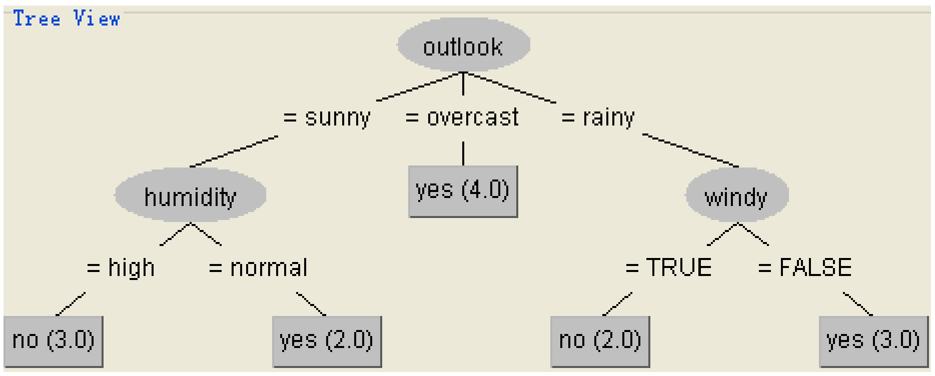
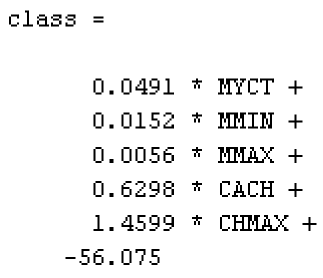


Figure 3.1: Example decision tree used to predict playing outside or not according to weather

If the attribute that is tested at a node is a nominal one, the number of children is usually the number of possible values of the attribute. If the attributes is numeric, the test at a certain node usually determines whether its value is greater or less than a predetermined constant, giving a two-way split.

This kind of decision trees are designed for predicting categories rather than numeric quantities. When it comes to predict numeric quantities, the same kind of tree can be used, but the leaf nodes of the tree should contain a numeric value that is the average of all the training set values to which the leaf applies. Since statisticians use term regression for the process of computing an expression that predicts a numeric quantity, decision trees with averaged numeric values at leaves are called *Regression Tree* [11].

Figure 3.2 shows a linear regression equation for class and Figure 3.3 shows a regression tree. The leaves of the tree are numbers that represent the average outcome for instances that reach the leaf. The tree is much larger and more complex than the regression equation. And regression tree is more accurate because a simple linear model poorly represents the data in this problem. Figure 3.4 is a tree whose leaves contain linear expressions, that is, regression equations, rather than single predicted value. This is called *Model Tree*. Figure 3.4 involves five linear models that belong to the five leaves, labeled from LM1 to LM5. The model tree approximates continuous functions by linear models.


```
class =

      0.0491 * MYCT +
      0.0152 * MMIN +
      0.0056 * MMAX +
      0.6298 * CACH +
      1.4599 * CHMAX +
    -56.075
```
Figure 3.2: Linear regression

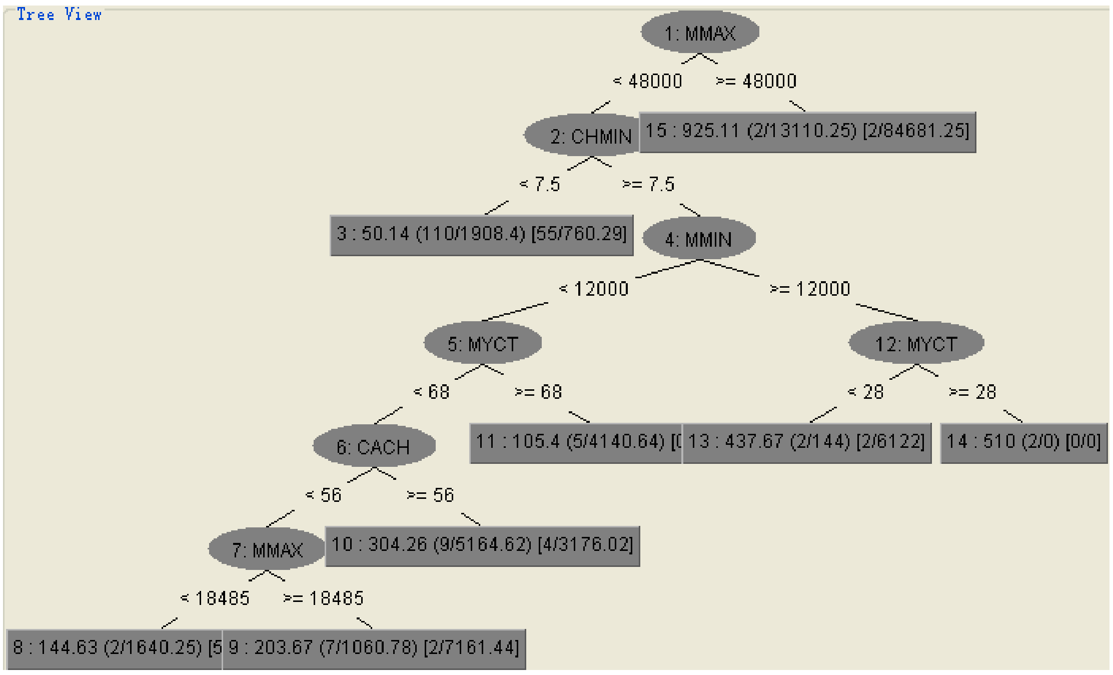
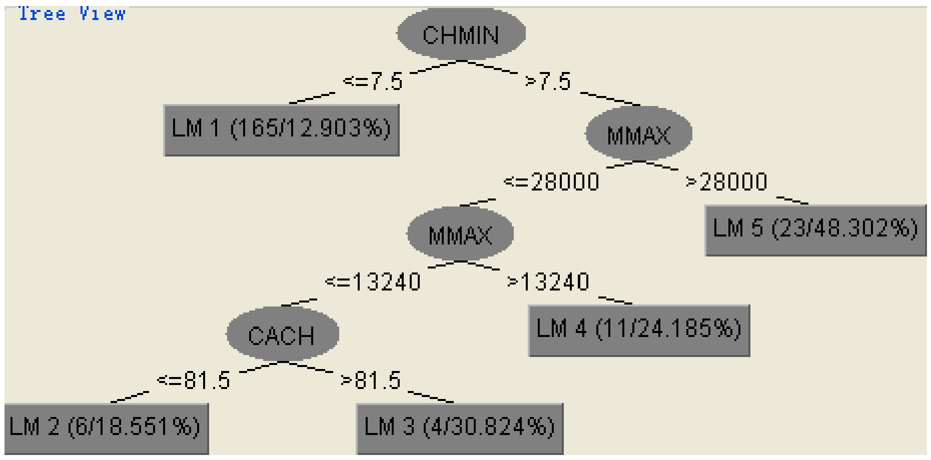


Figure 3.3: Regression tree



LM num: 1

class = -0.0055 * MYCT + 0.0013 * MMIN + 0.0029 * MMAX
+ 0.8007 * CACH + 0.4015 * CHMAX + 11.0971

LM num: 2

class = -1.0307 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX
+ 0.7866 * CACH - 2.4503 * CHMIN + 1.1597 * CHMAX + 70.8672

LM num: 3

class = -1.1057 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX
+ 0.7995 * CACH - 2.4503 * CHMIN + 1.1597 * CHMAX + 83.0016

LM num: 4

class = -0.8813 * MYCT + 0.0086 * MMIN + 0.0031 * MMAX
+ 0.6547 * CACH - 2.3561 * CHMIN + 1.1597 * CHMAX + 82.5725

LM num: 5

class = -0.4882 * MYCT + 0.0218 * MMIN + 0.003 * MMAX
+ 0.3865 * CACH - 1.3252 * CHMIN + 3.3671 * CHMAX - 51.8474

Figure 3.4: Model tree

The problem of constructing a decision tree can be expressed recursively. First, select an attribute to place at the root node and make one branch for each possible value. This splits up the example set into subsets, one for every value of the attribute. Now the process can be repeated recursively for each branch, using only those instances that actually reach the branch. If at any time all instances at a node have the same classification, stop developing that part of the tree. One practical algorithm is call C4.5 that is a series of improvements to ID3 which was developed and refined over many years by J.Ross Quinlan of the University of Sydney, Australia [12]. In Weka, J48 classifier implements the C4.5 algorithm and M5P implements the model tree method.

3.3 Bayesian Networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). Bayesian networks are drawn as a network of nodes, one for each attribute, connected by directed edges in such a way that there are no cycles – a directed acyclic graph [11].

Formally, Bayesian networks are directed acyclic graphs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes which are not connected represent variables which are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's parent variables and gives the probability of the variable represented by the node [11].

From [11], we can acquire definition and explanation of Bayesian network. Some contents are as follows.

Probability estimates are often more useful than plain predictions. They allow predictions to be ranked, and their expected cost to be minimized. In fact, there is a strong argument for treating classification learning as the task of learning class probability estimates from data. What is being estimated is the conditional probability distribution of the values of the class attribute given the values of the other attributes. The classification model represents this conditional distribution in a concise and easily comprehensible form.

Given values for each of a node’s parents, knowing the values for any other ancestors does not change the probability associated with each of its possible values, which means ancestors do not provide any information about the likelihood of the node’s values over and above the information provided by parents. This can be expressed:

$$\Pr[\text{node} \mid \text{ancestors}] = \Pr[\text{node} \mid \text{parents}]$$

which must hold for all values of the nodes and attributes involved. In statistics this property is called conditional independence. Multiplication is valid provided that each node is conditionally independent of its grandparents, great-grandparents and so on, given its parents. The multiplication step results directly from the chain rule in probability theory, which states that the joint probability of n attributes a_i can be decomposed into this product:

$$\Pr[a_1, a_2, \ldots, a_n] = \prod_{i=1}^{n} \Pr[a_i \mid a_{i-1}, \ldots, a_1]$$

The decomposition holds for any order of the attributes. Because Bayesian network is an acyclic graph, its nodes can be ordered to give all ancestors of node a_i indices smaller than i. Then, because of the conditional independence assumption,

$$\Pr[a_1, a_2, \ldots, a_n] = \prod_{i=1}^{n} \Pr[a_i \mid a_{i-1}, \ldots, a_1] = \prod_{i=1}^{n} \Pr[a_i \mid a_i'\text{s parents}]$$

which is exactly the multiplication rule that we applied previously.
Therefore,

$$P(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} P(X_v = x_v \mid X_{v+1} = x_{v+1}, \ldots, X_n = x_n)$$

$$= \prod_{v=1}^{n} P(X_v = x_v \mid X_j = x_j \text{ for each } X_j \text{ which is a parent of } X_v)$$

The way to construct a learning algorithm for Bayesian networks is to define two components: a function for evaluation a given network based on the data and a method for searching through the space of possible networks. The quality of a given network is measured by the probability of the data given the network.

Figure 3.5 shows the Bayesian network Graph with the weather sample generated by Weka. Figure 3.6 illustrates the Probability Distribution Table for node “temperature” that contains three nominal values: hot, mild and cool. These probabilities are calculated given the value of the parents of “temperature” – “play” and “outlook”.

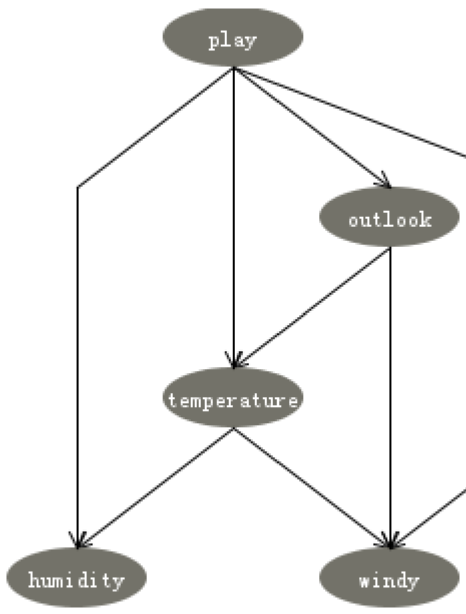
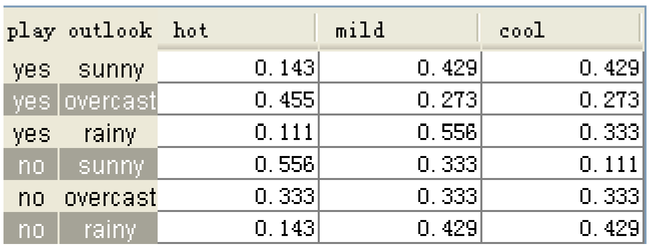


Figure 3.5: Bayesian network Graph with the weather sample.



play	outlook	hot	mild	cool
yes	sunny	0.143	0.429	0.429
yes	overcast	0.455	0.273	0.273
yes	rainy	0.111	0.556	0.333
no	sunny	0.556	0.333	0.111
no	overcast	0.333	0.333	0.333
no	rainy	0.143	0.429	0.429

Figure 3.6: Probability Distribution Table for node "temperature"

One simple and fast learning algorithm is call *K2* [13], starting with a given ordering of the attributes. Then it processes each node in turn and greedily considers adding edges from previously processed nodes to the current one. In each step it adds the edge that maximizes the network's score. When there is no further improvement, attention turns to the next node. One potentially instructive trick is to ensure that every attribute in the data is in the *Markov blanket* [14] of the node that represents the class attribute. A node's Markov blanket includes all its parents, children and children's parents. It can be shown that a node is conditionally independent of all other nodes given values for the nodes in its Markov blanket [11].

Another good learning method for Bayesian network classifiers is called *tree augmented Na ïve Bayes* (TAN) [15]. As the name implies, it takes the Na ïve Bayes classifier and adds edges to it. The class attribute is the single parent of each node of a Naive Bayes network: TAN considers adding a second parent to each node. If the class node and all corresponding edges are excluded from consideration, and assuming that there is exactly one node to which a second parent is not added, the resulting classifier has a tree structure rooted at the parentless node – this is where the name comes from. For this restricted type of network there is an efficient algorithm for finding the set of edges that maximizes the network's likelihood based on computing the network’s maximum weighted spanning tree. This algorithm is linear in the number of instances and quadratic in the number of attributes [11].

Bayesian networks are a special case of a wider class of statistical models called *graphical models*, which include networks with undirected edges (called *Markov networks*). Graphical modes are becoming increasingly popular in the Machine Learning community today.

Chapter 4

The SysWeka Platform

4.1 Framework description

The SysWeka platform for Systems-oriented Weka is designed and built between the lower Machine Learning infrastructures and higher cloud applications, which provides interface for higher software development. Figure 4.1 is the architecture of this platform. Command Interpreter collects the input and interpret the commands. General Evaluation utilizes the Prediction and Confidence models to provide general functionality for evaluation. Prediction and Confidence calculate the predictive values and confidence values with different classifiers specified in the command. Moreover, data interface is used to load raw data from source files and categories them into two types, numeric one and nominal one, which is constructive for prediction and confidence components. There is another new Importance-Aware Linear Regression method that is updated from normal linear regression with Importance-Aware feature.

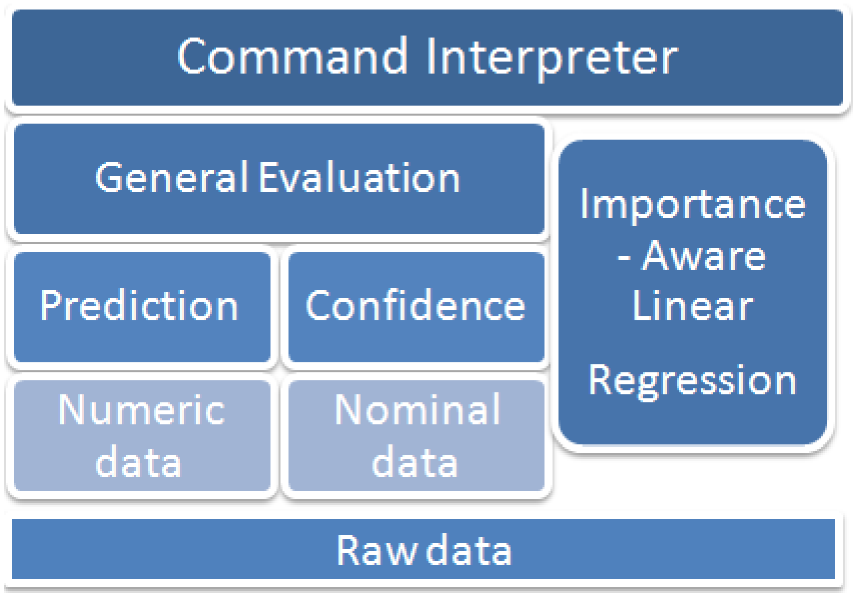


Figure 4.1: SysWeka platform architecture

The following graph is the main class diagram. The Predictor and Command Interpreter support the user interface of the middleware. Prediction and Confidence are computed according to the class data type. Also General Evaluation model can be considered from both nominal and numeric views internally. Actually, in the whole project environment, data type should be specified. The source data file is generated by the emulator [4] and is processed in the Prediction or Confidence model to predict the class values and confidence. General Evaluation is designed for general confidence prediction, which means that Machine Learning data sample gathered from all kinds of resources can be tested and evaluated here to obtain the prediction result directly. This is a common platform to do prediction and confidence calculations.

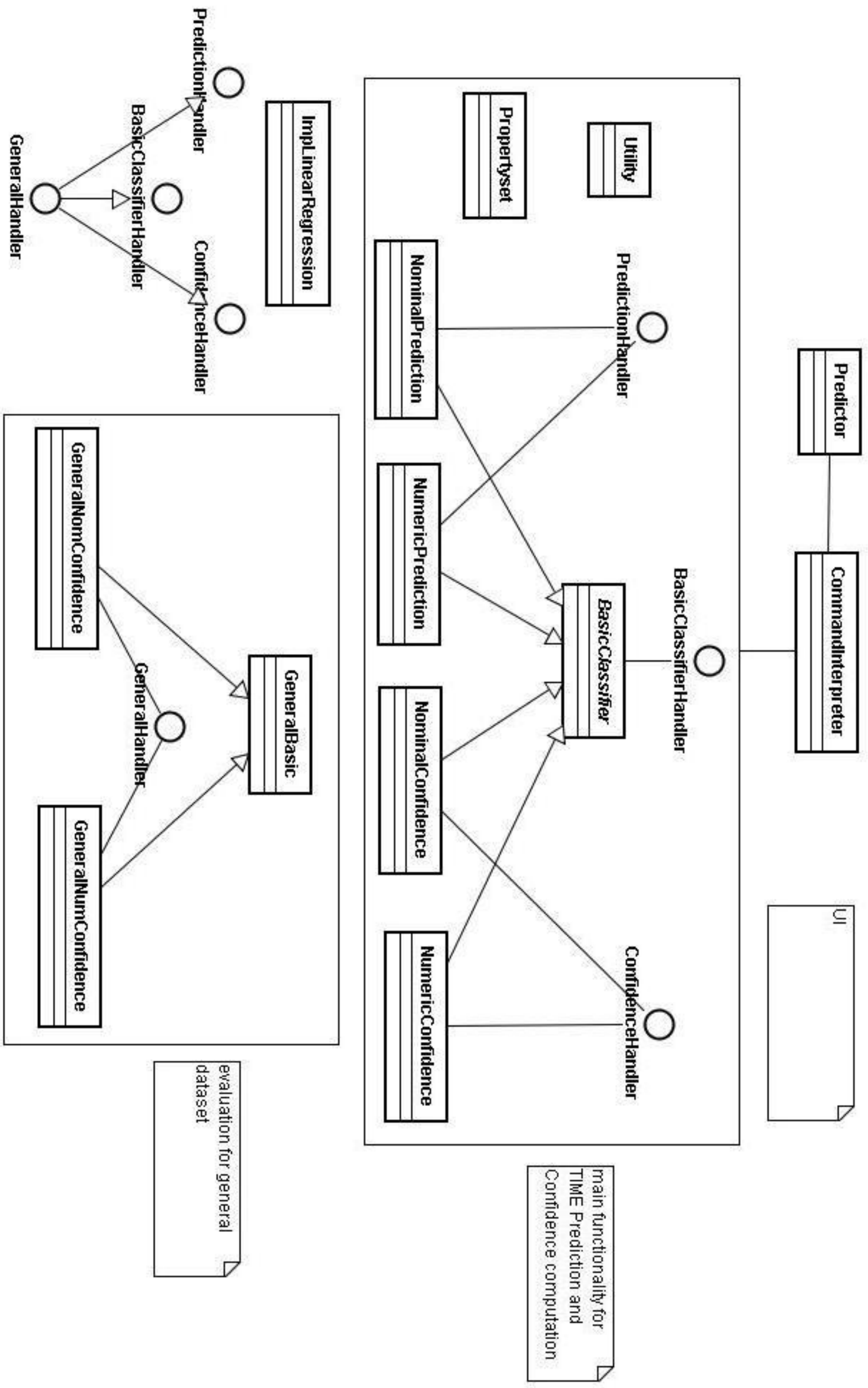
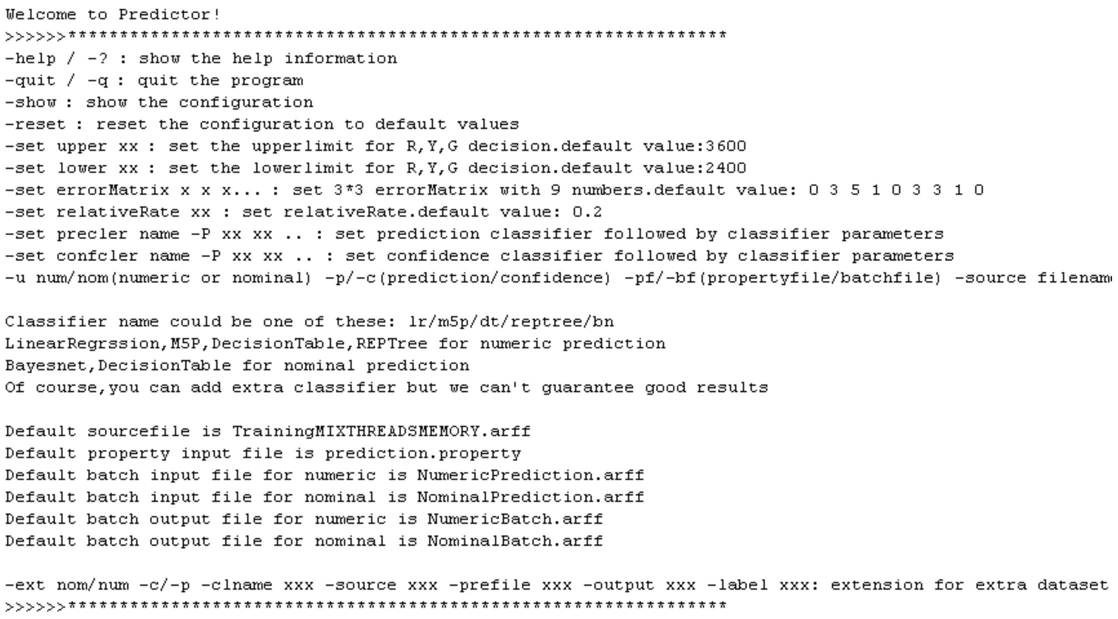


Figure 4.2: SysWeka main class diagram

The Figure 4.3 is the start-up user interface including command explanation and configuration information.



```
Welcome to Predictor!
>>>>>>***************************************************************
-help / -? : show the help information
-quit / -q : quit the program
-show : show the configuration
-reset : reset the configuration to default values
-set upper xx : set the upperlimit for R,Y,G decision.default value:3600
-set lower xx : set the lowerlimit for R,Y,G decision.default value:2400
-set errorMatrix x x x... : set 3*3 errorMatrix with 9 numbers.default value: 0 3 5 1 0 3 3 1 0
-set relativeRate xx : set relativeRate.default value: 0.2
-set precler name -P xx xx .. : set prediction classifier followed by classifier parameters
-set confcler name -P xx xx .. : set confidence classifier followed by classifier parameters
-u num/nom(numeric or nominal) -p/-c(prediction/confidence) -pf/-bf(propertyfile/batchfile) -source filenam

Classifier name could be one of these: lr/m5p/dt/reptree/bn
LinearRegrssion,M5P,DecisionTable,REPTree for numeric prediction
Bayesnet,DecisionTable for nominal prediction
Of course,you can add extra classifier but we can't guarantee good results

Default sourcefile is TrainingMIXTHREADSMEMORY.arff
Default property input file is prediction.property
Default batch input file for numeric is NumericPrediction.arff
Default batch input file for nominal is NominalPrediction.arff
Default batch output file for numeric is NumericBatch.arff
Default batch output file for nominal is NominalBatch.arff

-ext nom/num -c/-p -clname xxx -source xxx -prefile xxx -output xxx -label xxx: extension for extra dataset
>>>>>>***************************************************************
```

Figure 4.3: Start-up user interface

4.2 Prediction components

Prediction components provide prediction interface to predict nominal or numeric class and also evaluate the prediction results from original datasets. These source datasets are the outcome of Javier Alonso's works [3] [4]. Those system metrics are selected to represent the specific field of systems and models are built based on these attributes to predict the TIME_UNTIL_FAULT class. The work in this paper will extend and apply to that kind of scenarios.

During this process, we make further development from results of Javier Alonso's works. Mainly, TIME_UNTIL_FAULT class is divided into three categories (RED, YELLOW and GREEN), which can indicate different urgency of application actions. These categories predicted from TIME_UNTIL_FAULT have different definitions according to varied applications. Furthermore, these three metrics can donate the necessity of migrating some VM to other physical machine [2] or necessity of recovery operation, jobs-rescheduling and booting.

In this paper, RED means this machine will crash within very limited time. YELLOW means this machine will crash in a certain time, which gives an ordinary warning to the scheduling system, and GREEN, of course, shows the satisfying performance of certain machine.

After building and labeling this new class, classifiers should be adopted to construct new prediction models. We use three main methods here: Linear Regression, M5P and Bayesian networks, all described in previous section. As is known, linear regression and M5P are used to predict numeric values, while Bayesian networks focus on nominal prediction. In order to employ these methodologies, different strategies are made. Since TIME_UNTIL_FAULT is numeric class, Linear Regression and M5P can be used directly to predict TIME_UNTIL_FAULT first, and then TIME_WARNING could be calculated from the TIME_UNTIL_FAULT values. With Bayesian networks, first, TIME_WARNING is calculated, and then it will be predicted without TIME_UNTIL_FAULT class.

Internally, Bayesian networks, Decision Table and other nominal prediction method can be used for Nominal Prediction. And Linear Regression, M5P, Decision Table, REPTree and other numeric prediction method can be used for Numeric Prediction for this TIME_WARNING class. Furthermore, after building prediction models, predictions can be acted through batch file or property file, while batch file mode is similar as off-line and property file mode is like on-line process.

When it comes to evaluation, relative coefficient is specified to estimate the relative error rate that is used in numeric prediction to express the relative error. Because in some cases, errors should be considered with relative rate not exact values. The relative rate is more suitable for especially large numbers, for instance, given 900 seconds to crash, the predictive time-to-crash is 1500 seconds, this difference, 600 seconds, is more crucial than given 9000 seconds to crash with predictive value 8400 seconds. Consequently, relative error rate is introduced in numeric prediction to specify the relative error status.

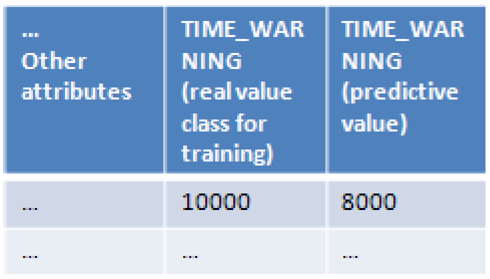
In addition, error metrics are involved when evaluating the TIME_WARNING predictive results. The different predictive results can have a diverse influence based on different real values. For example, if the real value is RED and the predictive value is GREEN, which means the decision system may continue to assign jobs to this machine and this will of course make machine crash and reduce the availability. So, this is an important error. On the other hand, if the real value GREEN is predicted as RED, the decision system may no longer send any jobs to this machine and do recovery procedures on that machine. This will not badly damage the availability and performance and it seems to have no influence on the whole system. Considering the different importance of distinct results could generate, error metrics are required to adjust the error accuracy.

4.3 Confidence Prediction

Confidence has been defined as “a state of being certain either that a hypothesis or prediction is correct or that a chosen course of action is the best or most effective” in science, which is an essential metrics to represent the correctly rate.

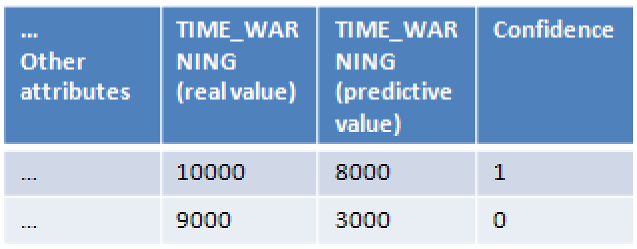
Confidence is calculated from the difference of real value and predictive value assigned from 0 to 1 to indicate the correctly rate of prediction. After acquiring the confidence values, confidence is selected as the new class to be predicted and models will be built using original attributes, predictive value and confidence without previous class—the real value. In this case, there are two models constructed, one to predict the real value and the other to predict the confidence based on the previous predictive value. Therefore, we can not only do time-warning prediction but also measure the predictive result with confidence to reveal its accuracy.

The confidence process can be described as follows:



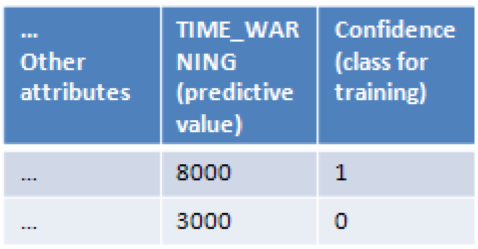
… Other attributes	TIME_WARNING (real value class for training)	TIME_WARNING (predictive value)
…	10000	8000
…	…	…

(a)



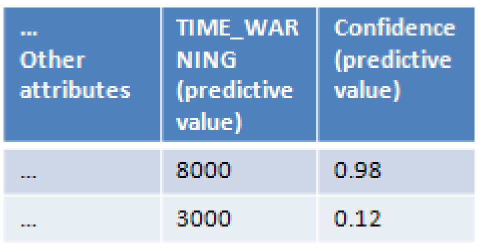
… Other attributes	TIME_WARNING (real value)	TIME_WARNING (predictive value)	Confidence
…	10000	8000	1
…	9000	3000	0

(b)



… Other attributes	TIME_WARNING (predictive value)	Confidence (class for training)
…	8000	1
…	3000	0

(c)



… Other attributes	TIME_WARNING (predictive value)	Confidence (predictive value)
…	8000	0.98
…	3000	0.12

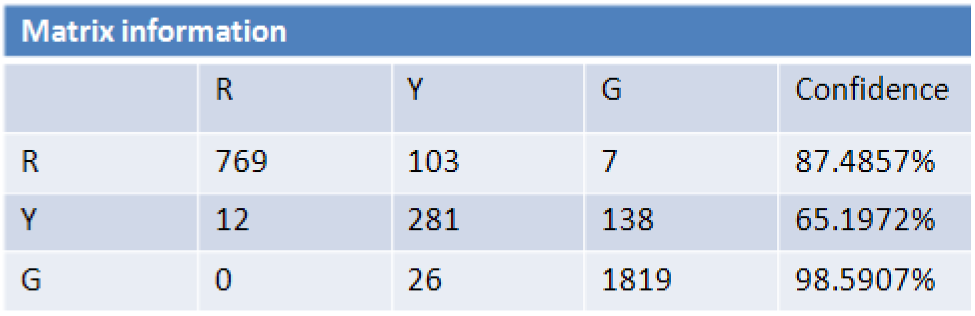
(d)

Figure 4.4:
(a): set TIME_WARNING as class for training and predict the class value.
(b): add a new numeric attribute Confidence and calculate its value.
(c): remove the real value attribute and set Confidence as class for training.
(d): utilize first model to predict the TIME_WARNING then adopt second model to predict Confidence.

During Confidence prediction, two different models will be built to predict the TIME_WARNING and Confidence. Afterwards, we can obtain both predict values and its confidence according to different attribute values, which improves the prediction accuracy extraordinarily.

Each instance's confidence is predicted respectively and whole prediction confidence of each warning type (RED, YELLOW, and GREEN) is measured as well.

The following chart shows the distribution of prediction results using M5P algorithm and error metrics with 2751 numbers of instances where the confidence demonstrates the correctly rate of each category (R-R, Y-Y and G-G).



Matrix information

	R	Y	G	Confidence
R	769	103	7	87.4857%
Y	12	281	138	65.1972%
G	0	26	1819	98.5907%

Figure 4.5: Distribution of prediction results using M5P and error metrics

4.4 Importance-Aware Linear Regression

In Section 3.1, we discuss Linear Regression and Ordinary least squares (OLS) estimator. Here, Importance-Aware features are added to linear regression to support prediction with different importance of different instances. Since diverse instance may have varied importance in practical environments, prediction with Importance-Aware can surely enhance the experience and usage of prediction. Here, we still use Ordinary least squares (OLS) to estimate the Importance-Aware Linear Regression.

4.4.1 Derivation of the formulas

From Section 3.1, the sum of the squares of the differences is written as:

$$Q = \sum_{i=1}^{n} (y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2$$

where the expression inside the parentheses is the difference between the *i*th instance's actual class and its predicted class. This sum of squares is what we have to minimize by choosing the coefficients appropriately.

Here importance metric *imp*$_{(i)}$ is added into this formula:

$$Q = \sum_{i=1}^{n} imp_{(i)} (y^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)})^2 \qquad (4.4.1.1)$$

This sum of the product of given importance and squares is what we should minimize by choosing the coefficients appropriately. Using the same mathematic idea, we let the first order derivative equal to zero and compute the coefficients w_j. That is

$$\frac{\partial Q}{\partial w_j} = 0, (j = 0,1,2 \ldots . k)$$

$$\sum_{i=1}^{n} imp_{(i)}\left(y^{(i)}-\sum_{j=0}^{k} w_j a_j^{(i)}\right)^2=\sum_{i=1}^{n}\left(\sqrt{imp_{(i)}}\,y^{(i)}-\sqrt{imp_{(i)}}\sum_{j=0}^{k} w_j a_j^{(i)}\right)^2$$

$$\begin{cases}\dfrac{\partial Q}{\partial w_0}=2\displaystyle\sum_{i=1}^{n}\left(\sqrt{imp_{(i)}}y^{(i)}-\sqrt{imp_{(i)}}w_0\ \ldots-\sqrt{imp_{(i)}}\,w_k a_k^{(i)}\right)\left(-\sqrt{imp_{(i)}}\right)=0\\ \qquad\qquad\qquad\ldots\ldots\ldots\ldots\\ \dfrac{\partial Q}{\partial w_k}=2\displaystyle\sum_{i=1}^{n}\left(\sqrt{imp_{(i)}}y^{(i)}-\sqrt{imp_{(i)}}w_0\ \ldots-\sqrt{imp_{(i)}}\,w_k a_k^{(i)}\right)\left(-\sqrt{imp_{(i)}}a_k^{(i)}\right)=0\end{cases}$$

$$\begin{cases}w_0\displaystyle\sum_{i=1}^{n} imp_{(i)}+w_1\sum_{i=1}^{n} imp_{(i)}a_1^{(i)}+\ldots+w_k\sum_{i=1}^{n} imp_{(i)}a_k^{(i)}=\sum_{i=1}^{n} imp_{(i)}y^{(i)}\\ \qquad\qquad\qquad\ldots\ldots\ldots\ldots\\ w_0\displaystyle\sum_{i=1}^{n} imp_{(i)}a_k^{(i)}+w_1\sum_{i=1}^{n} imp_{(i)}a_1^{(i)}a_k^{(i)}+\ldots+w_k\sum_{i=1}^{n} imp_{(i)}(a_k^{(i)})^2=\sum_{i=1}^{n} imp_{(i)}y^{(i)}a_k^{(i)}\end{cases}$$

These formulas can be written with matrix form:

$$\begin{bmatrix}\displaystyle\sum_{i=1}^{n} imp_{(i)} & \displaystyle\sum_{i=1}^{n} imp_{(i)}a_1^{(i)} & \ldots & \displaystyle\sum_{i=1}^{n} imp_{(i)}a_k^{(i)}\\ \displaystyle\sum_{i=1}^{n} imp_{(i)}a_1^{(i)} & \displaystyle\sum_{i=1}^{n} imp_{(i)}(a_1^{(i)})^2 & \ldots & \displaystyle\sum_{i=1}^{n} imp_{(i)}a_k^{(i)}a_1^{(i)}\\ \ldots & \ldots & \ldots & \ldots\\ \displaystyle\sum_{i=1}^{n} imp_{(i)}a_k^{(i)} & \displaystyle\sum_{i=1}^{n} imp_{(i)}a_1^{(i)}a_k^{(i)} & \ldots & \displaystyle\sum_{i=1}^{n} imp_{(i)}(a_k^{(i)})^2\end{bmatrix}\begin{bmatrix}w_0\\ w_1\\ w_2\\ \ldots\\ w_k\end{bmatrix}=\begin{bmatrix}\displaystyle\sum_{i=1}^{n} imp_{(i)}y^{(i)}\\ \displaystyle\sum_{i=1}^{n} imp_{(i)}y^{(i)}a_1^{(i)}\\ \ldots\\ \displaystyle\sum_{i=1}^{n} imp_{(i)}y^{(i)}a_k^{(i)}\end{bmatrix}$$

We define $A=\begin{bmatrix}1 & a_1^1 & \ldots & a_k^1\\ 1 & a_1^2 & \cdots & a_k^2\\ \ldots & \ldots & \ldots & \ldots\\ 1 & a_1^n & \cdots & a_k^n\end{bmatrix}$, then $A'=\begin{bmatrix}1 & 1 & \ldots & 1\\ a_1^1 & a_1^2 & \ldots & a_1^n\\ \ldots & \ldots & \ldots & \ldots\\ a_k^1 & a_k^2 & \ldots & a_k^n\end{bmatrix}$.

Define diagonal matrix $T=\begin{bmatrix}\sqrt{imp_1} & 0 & 0 & 0\\ 0 & \sqrt{imp_2} & 0 & 0\\ 0 & 0 & \ldots & 0\\ 0 & 0 & 0 & \sqrt{imp_n}\end{bmatrix}$,

Then

$$A'T = \begin{bmatrix} \sqrt{imp_1} & \sqrt{imp_2} & \dots & \sqrt{imp_n} \\ \sqrt{imp_1}\,a_1^1 & \sqrt{imp_2}\,a_1^2 & \dots & \sqrt{imp_n}\,a_1^n \\ \dots & \dots & \dots & \dots \\ \sqrt{imp_1}\,a_k^1 & \sqrt{imp_2}\,a_k^2 & \dots & \sqrt{imp_n}\,a_k^n \end{bmatrix}$$

Define $X = (A'T)' = T'(A')' = T'A = TA = \begin{bmatrix} \sqrt{imp_1} & \sqrt{imp_1}\,a_1^1 & \dots & \sqrt{imp_1}\,a_k^1 \\ \sqrt{imp_2} & \sqrt{imp_2}\,a_1^2 & \dots & \sqrt{imp_2}\,a_k^2 \\ \dots & \dots & \dots & \dots \\ \sqrt{imp_n} & \sqrt{imp_n}\,a_1^n & \dots & \sqrt{imp_n}\,a_k^n \end{bmatrix}$

$X' = A'T$

So the matrix form can be expressed as follows:

$X'X\,W = X'TY$, where $W = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_k \end{bmatrix}$ and $Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \dots \\ Y_k \end{bmatrix}$

According to matrix operation, because Rank(X) = k+1, $X'X$ is a square matrix of rank k+1. So $X'X$ is a square matrix of full rank. Therefore, the inverse matrix of $X'X$ exists. Then

$$W = (X'X)^{-1} X'TY$$

After matrix inversing and transposing, the coefficient matrix W is acquired, which is the goal for our Importance-Aware linear regression.

On the other hand, the weights of linear regression without Importance-Aware can be expressed as follows using the definition above.

$$W = (A'A)^{-1} A'TY$$

The two formulas are very similar and the Importance-Aware weighs involve one additional diagonal matrix T, which means we can build our own linear regression new version or can modify the Weka source code conveniently.

Chapter 5

Experimental Evaluation

5.1 Experimental prediction with diverse Machine Learning methods

During the experiments, a source dataset file is used to make further prediction and analysis. This dataset file is one of Javier Alonso's works for predicting software time until crash [3] [4]. There are many system metrics in that dataset file including two different resources: Threads and Memory, individually or merged, where the numeric class TIME_UNTIL_FAULT is predicted by those system metrics in Javier Alonso's work using M5P methodology. However, these series of experiment focus on predicting the nominal class TIME_WARNING (R-RED, Y-YELLOW and G-GREEN) with confidence, performance comparison and accuracy improvements.

Starting from the source dataset file, there are 49 kinds of numeric system metrics in the model and some of them are listed as follows. Detailed descriptions on these metrics can be found in [3].

throughput
Reponse_time
Workload
SYSTEM_LOAD
DISC_USAGE
SWAP
PROCESSES
MEMORY_SYSTEM
TOMCAT_MEMORY
THREADS
HTTP_CONNECT
MSYSQL_CONNECT
THREADS_VARIATION_EWMA
EDEN_PERCENTAGE_USED

DIVISION_ATTRIBUTE_EDEN_MEMORY_VARIATION
NORMALIZED_INVERSE_EDEN_MEMORY_VARIATION
MEMORY_SYSTEM_EWMA
TOMCAT_MEMORY_EWMA
TOMCAT_MEMORY_VARIATION_EWMA
NORMALIZED_SYSTEM_MEMORY_VARIATION_EWMA
OLD_MEMORY_USED
OLD_PERCENTAGE_USED
……

Table 5.1: Key system metrics in the dataset source file

In respect of TIME_UNTIL_FAULT class, minimum is 0, maximum is 20362, mean is 7393.124 and standard deviation is 5550.533. Based on upper and lower thresholds, TIME_UNTIL_FAULT value is transformed into TIME_WARNING value. In our experiments, the default value of upper limit is 3600 seconds and lower limit is 2400 seconds. Within 2751 instances, there are 610 instances labeled R, 319 instances labeled Y and 1822 labeled G.

First, linear regression, M5P, and Bayesian network are adopted to show the different accuracy for prediction. Nominal and numeric prediction can be utilized in these experiments using different strategies described in Chapter 4.

The details of the Linear Regression Model, M5P Model, and Bayesian network Model with default setting values, relative rate and error Matrix presented can be found in Appendix 7.1.

From this experiment, we can get the comparison table as follows:

	Linear Regression	M5P	Bayesian network
Time to build model	0.594 s	1.438 s	0.313 s
Correlation coefficient	-0.0108	0.9958	NAN
Correctly Classified Instances	NAN	NAN	92.8753%
Relative absolute error	54617311691.7164%	4.1581%	14.4073%
Relative error	47.8735%	7.5972%	NAN
Correctly Prediction Rate for R	41.3887%	87.4857%	57.8723%
Correctly Prediction Rate for Y	10.6886%	65.1972%	74.1772%
Correctly Prediction Rate for G	94.2192%	98.5908%	91.2998%

Table 5.2: Comparison with three kinds of methods

This table indicates the different key measurements generated by Linear Regression, M5P and Bayesian network using *K2* as search algorithm and SimpleEstimator as estimator. The Correctly Classified Instances, Correlation coefficient and other metrics are generated by 10-fold cross-validation while the matrix information shown is calculated by predicting training data with relative error matrix to test models.

As is illustrated above, linear regression is not appropriate for this time-warning prediction, whose Correlation coefficient is negative, Relative absolute error is even out of imagination and higher Correctly Prediction Rate for G. This means linear regression is suitable to predict large numbers that is, if system runs smoothly without unexpected errors, it is linear and will take a long time until crash. And that is why it is widely used and generally successful.

M5P is more successful in this experiment than linear regression and Bayesian network only with lower Correctly Prediction Rate for Y compared with Bayesian network. The Correctly Prediction Rate for R and G are more satisfactory and the relative error shows a large improvement. Just like statements in Javier Alonso's work [4], M5P proved to be more efficient to predict non-linear numeric values, because it involves model tree and partly linear.

Bayesian network, even though it is a more sophisticated technique, does not have a promising performance in this experiment. According to its definition discussed in Section 3.3, Bayesian networks are drawn as a network of nodes, one for each attribute, connected by directed edges – a directed acyclic graph.

In this experiment, the Bayesian network prediction method adopts *K2* as search algorithm with max one parent per node, which definitely degrades Bayesian network's function at this point, because there are 49 attributes in this dataset and some of these attributes are relative to each other and these connections have a large influence on the class. With max one parent per node *K2* search algorithm, every node only has one parent – the class. The Probability Distribution Table can be only calculated based on class, which eliminates the impact with other possible attributes. So the result is not as good as it should be. The more parents a node has, the higher impact the node will obtain. As more parents per node are specified using *K2* search algorithm, the performance improves tremendously.

Here we do not change the estimator and still use the simple and fast search algorithm *K2* only with more parents per node, actually three parents and five parents for comparison. To be surprised Bayesian network prediction with max three or more parents per node using training set results performs perfect classification to predict training set.

The details of Bayesian networks with max three or more parents per node using training set results obtained by these algorithms can be found in Appendix 7.2.

Though Bayesian networks with max three parents per node obtain 100% classification using training set test, they still achieve 99.3457% correctly rate for cross-validation. Another experiment using 10-fold cross-validation shows the little difference between Bayesian network with max three parents and five parents per node.

The Correctly Classified Rate is very similar but it costs 6.19 seconds with max five parents per node to build models, 4 times longer than time spent with max three parents per node. So we only consider max three parents per node in this project.

In the last experiment, Bayesian network using *K2* as search algorithm and SimpleEstimator as estimator has perfect classification with max three parents per node using training set test. According to the 10-fold cross-validation above, it also proved to be very satisfactory, almost 99% correctly rate. The following experiment is using 60% percentage split for Bayesian network with max three parents per node. The details of these experiments can be also found in Appendix 7.2.

Bayesian network using *K2* as search algorithm and SimpleEstimator as estimator with max three parents per node performs advantageous classification with three different kinds of testing methods: use training set, cross- validation and percentage split. Though the graphical model is more complex than other models, this method does provide high performance within reasonable time.

5.2 Confidence Prediction experiments

Here confidence prediction experiments of both numeric class and nominal class will be presented. Since confidence is a number valued from 0 to 1 measuring the accuracy of the predictive values, there are two other classifiers that can help to improve the confidence accuracy. Decision Table is used for both numeric and nominal prediction and REPTree is used for numeric prediction.

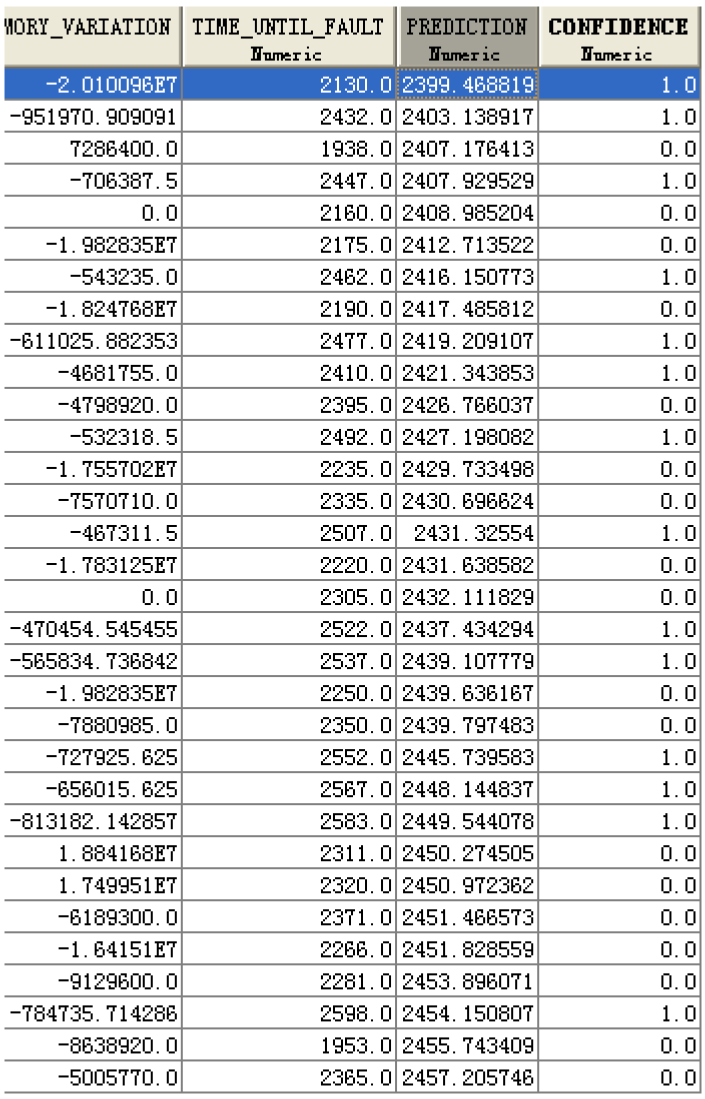
M5P is adopted to train and predict the numeric class value as a start and then use other methods to make a confidence training and prediction.

Decision Table is the simplest and basic way to represent the dataset, which involves selecting attributes to build plenty of rules that try to conclude the dataset. REPTree builds a decision or regression tree using information gain/variance reduction and prunes it using reduced-error pruning [11].

The detailed experimental results are shown in Appendix 7.3.

These charts in Appendix 7.3 show that confidence prediction and class value prediction are quite two different thing. M5P works well in prediction class values while is inappropriate for confidence prediction here. So experiments should be performed to test the advantageous algorithms. Furthermore, when we look at the dataset, we can find that because the class prediction classifier is more accurate and GREEN data (large TIME_UNTIL_FAULT value) is more than other YELLOW and RED data, the confidence valued 1 is much more than that valued 0, the predictive confidence (average value is almost 0.9) is more close to 1 than 0.5 or 0.

The following image is training middle-dataset using M5P to predict class value and REPTree to predict confidence. In this experiment, entire training set is used for testing confidence prediction. Weka ArffViewer is used here to represent these datasets.



MORY_VARIATION	TIME_UNTIL_FAULT Numeric	PREDICTION Numeric	CONFIDENCE Numeric
-2.010096E7	2130.0	2399.468819	1.0
-951970.909091	2432.0	2403.138917	1.0
7286400.0	1938.0	2407.176413	0.0
-706387.5	2447.0	2407.929529	1.0
0.0	2160.0	2408.985204	0.0
-1.982835E7	2175.0	2412.713522	0.0
-543235.0	2462.0	2416.150773	1.0
-1.824768E7	2190.0	2417.485812	0.0
-611025.882353	2477.0	2419.209107	1.0
-4681755.0	2410.0	2421.343853	1.0
-4798920.0	2395.0	2426.766037	0.0
-532318.5	2492.0	2427.198082	1.0
-1.755702E7	2235.0	2429.733498	0.0
-7570710.0	2335.0	2430.696624	0.0
-467311.5	2507.0	2431.32554	1.0
-1.783125E7	2220.0	2431.638582	0.0
0.0	2305.0	2432.111829	0.0
-470454.545455	2522.0	2437.434294	1.0
-565834.736842	2537.0	2439.107779	1.0
-1.982835E7	2250.0	2439.636167	0.0
-7880985.0	2350.0	2439.797483	0.0
-727925.625	2552.0	2445.739583	1.0
-656015.625	2567.0	2448.144837	1.0
-813182.142857	2583.0	2449.544078	1.0
1.884168E7	2311.0	2450.274505	0.0
1.749951E7	2320.0	2450.972362	0.0
-6189300.0	2371.0	2451.466573	0.0
-1.64151E7	2266.0	2451.828559	0.0
-9129600.0	2281.0	2453.896071	0.0
-784735.714286	2598.0	2454.150807	1.0
-8638920.0	1953.0	2455.743409	0.0
-5005770.0	2365.0	2457.205746	0.0

Figure 5.1: Training confidence mid dataset

This real-confidence is generated by comparing the difference between the real value and the predictive value for TIME_UNTIL_FAULT using thresholds. Then this model is trained to predict confidence value.

The following dataset can be found in Appendix 7.4.

The dataset (Figure 7.18) is the result of the whole confidence prediction work. Predictive value for TIME_UNTIL_FAULT and the confidence about this prediction is also estimated. So we can make a further decision whether this prediction can be trusted or not.

Since Bayesian network with max three parents per node performs perfect using training set test, in order to illustrate the status of confidence, Bayesian network with max one parent per node is practiced in the following experiment to reduce the accuracy to make sure that there will be enough confidence valued 0.

Nominal class value is predicted and also the numeric confidence is estimated. Here some prediction is not correct so the confidence is estimated as 0.

5.3 Evaluation on Importance-Aware Linear Regression

In this section, experiments are taken to clarify the functionality of Importance-Aware Linear Regression. In order to represent the result explicitly, we test the algorithm using training dataset without relative error matrix to verify the accuracy and compare the difference. When using this algorithm, importance-per instance should be specified. Generally in the future, this method will be further updated and developed to benefit the importance of current instances. The principle of this algorithm is described and derived in Section 4.4.

The comparison of general Linear Regression model and Importance-Aware Linear Regression model can be found in Appendix 7.5.

Here *imp(R)* is assigned 1, *imp(Y)* as 4 and *imp(G)* as 2. The Importance-Aware Linear Regression prediction results are more satisfactory than linear regression as shown in table 5.3. The Importance-Aware correctly prediction rate for R and Y improve 10% from 70% and 29% to 80% and 39%, which is very important for R prediction, because if R is predicted as Y or G, it is rather detrimental for decision system.

Real class value	Linear Regression (Predictive Result)			Importance-Aware Linear Regression (Predictive Result)		
	R	Y	G	R	Y	G
R	426	102	82	494	37	79
Y	115	95	109	69	126	124
G	13	28	1781	7	19	1796
Correctly Prediction Rate	69.836%	29.781%	97.750%	80.984%	39.498%	98.573%

Table 5.3: Prediction results of Linear Regression and Importance-Aware Linear Regression

In next experiment, we set the importance according to the training set class classification. We assign GREEN data 4, YELLOW data 4 and 2 to RED data. These coefficients are obtained by minimizing the formula (4.4.1.1) in Section 4.4.

According to the formula (4.4.1.1) and experiment, the larger value we assign to one classification, the less chance for the class close to the thresholds be predicted into this classification. This can be explained from the expression above: if the $imp_{(i)}$ is becoming larger, the sum value will be correspondingly bigger with original classification while the difference between the real value and predictive value does not change. At the same time, if another classification can reduce the difference and involves a smaller $imp_{(i)}$ value, then this new classification is more accepted for the model. However, since formula (4.4.1.1) is calculated with sum operation, every $imp_{(i)}$

value has an impact on each other. Practically, it is more complicated to manipulate them accurately.

Therefore, we assign a smaller value to the RED data in order to try to classify more data to RED category, because RED data predicted as YELLOW will make more detrimental influence than YELLOW data classified as RED. Exactly, RED prediction accuracy is higher valued 89.18% compared with 69.83% but the YELLOW prediction accuracy is relatively lower only 15%. If all the importance is same, just as there is not importance specified.

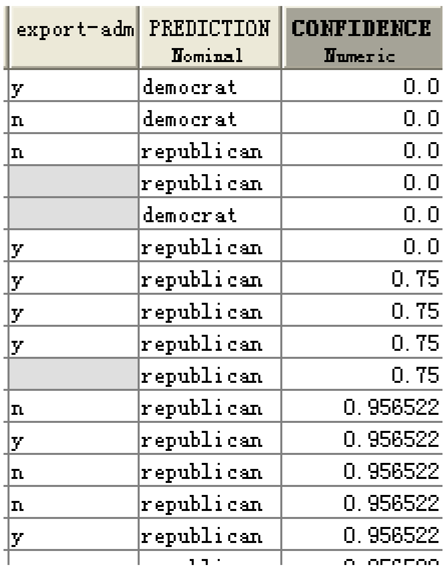
Imp(R) value while Imp(Y)=4 Imp(G) = 4	R-Prediction Rate	Y-Prediction Rate	G-Prediction Rate
2	89.180 %	15.047 %	98.683 %
3	81.639 %	26.019%	98.024 %
4	69.836 %	29.781%	97.750%
5	63.934 %	42.006 %	95.664 %
6	47.869%	32.602%	95.664 %
7	33.279%	26.019%	95.280%
8	19.016%	19.749%	94.786%
9	10.984%	15.987%	93.578%
10	7.049%	14.734%	92.261%

Table 5.4: Different prediction rate with different Imp(R)

From this table, given Imp(Y) and Imp (G) are the same valued 4, the larger Imp(R) value is, the worse performance R-Prediction Rate will have. Also Y and G prediction rate will be affected gradually. Of course, these series of experiments set the importance based on classification. On the other hand, different importance making strategies can be developed to meet more complex requirements in practice.

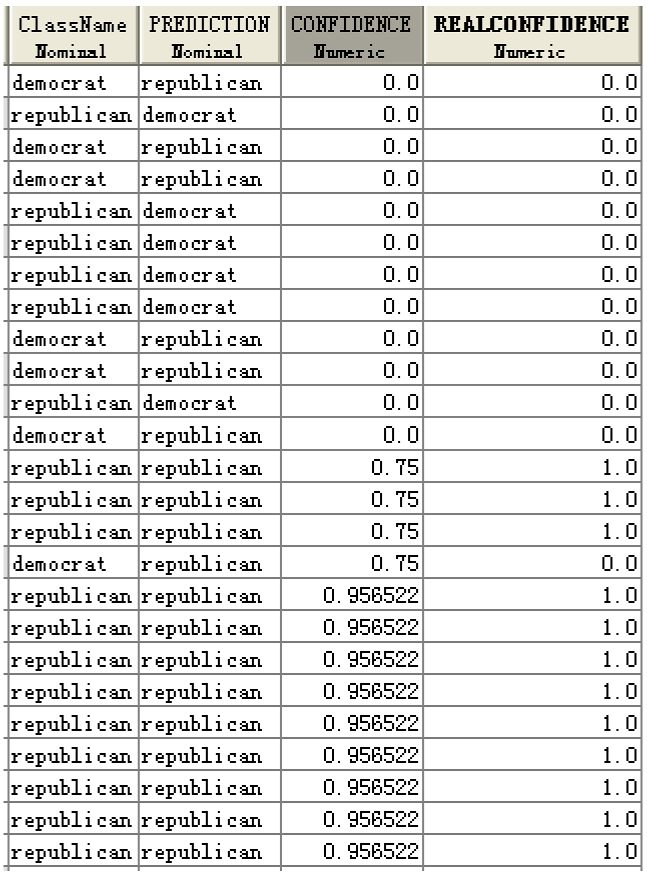
5.4 General Practice

Here we utilize this framework to predict other general dataset. A famous dataset call "Congressional Voting Records Data Set" [16], which includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. This dataset contains nine difference attributes that represent distinct types of votes. The objective is to predict whether a person is a democrat or republican from his varied kinds of votes. Predictive class and confidence are all estimated.



export-adm	PREDICTION Nominal	CONFIDENCE Numeric
y	democrat	0.0
n	democrat	0.0
n	republican	0.0
	republican	0.0
	democrat	0.0
y	republican	0.0
y	republican	0.75
y	republican	0.75
y	republican	0.75
	republican	0.75
n	republican	0.956522
y	republican	0.956522
n	republican	0.956522
n	republican	0.956522
y	republican	0.956522

Figure 5.2: Predictive class value and confidence



ClassName Nominal	PREDICTION Nominal	CONFIDENCE Numeric	REALCONFIDENCE Numeric
democrat	republican	0.0	0.0
republican	democrat	0.0	0.0
democrat	republican	0.0	0.0
democrat	republican	0.0	0.0
republican	democrat	0.0	0.0
republican	democrat	0.0	0.0
republican	democrat	0.0	0.0
republican	democrat	0.0	0.0
democrat	republican	0.0	0.0
democrat	republican	0.0	0.0
republican	democrat	0.0	0.0
democrat	republican	0.0	0.0
republican	republican	0.75	1.0
republican	republican	0.75	1.0
republican	republican	0.75	1.0
democrat	republican	0.75	0.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0
republican	republican	0.956522	1.0

Figure 5.3: Comparison of real values and prediction values

In this example, Bayesian network using *K2* as search algorithm with max three parents per node is used to predict nominal class value and DecisionTable method is adopted to predict the confidence.

These grey areas mean the missing values. Figure 5.2 is the standard output for general prediction involving predictive class value and confidence, while dataset in

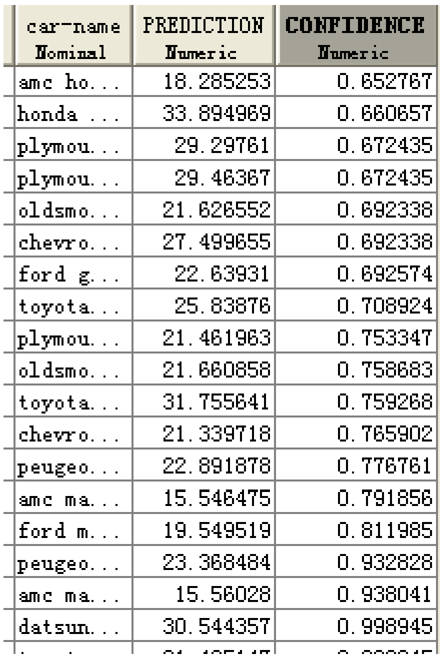
Figure 5.3 is utilized for testing to show the difference explicitly. Confidence 0 means this prediction for class value is incorrect, which can be explained as the diverse values of ClassName (class) and Prediction result. From this dataset, the following table can be achieved.

Confidence Threshold	Correctly Prediction Rate
Confidence >= 75%	99.29%
Confidence < 75%	0%

Table 5.5: Confidence accuracy

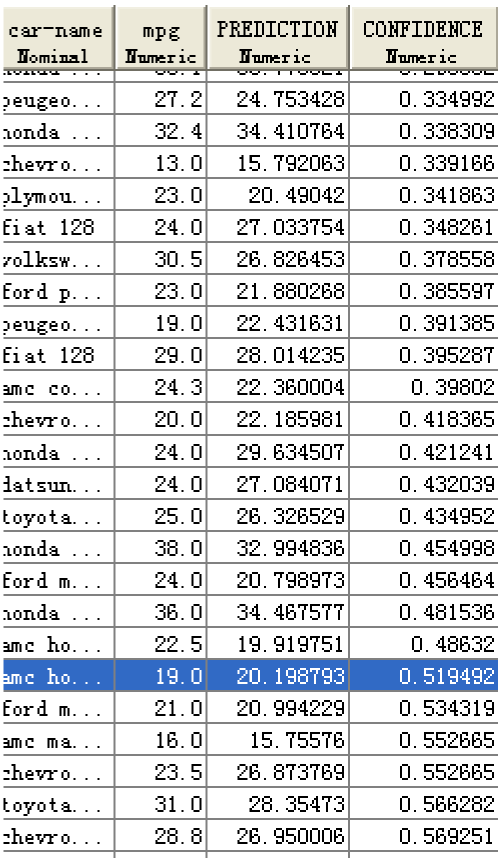
The following experiment is designed for general numeric confidence prediction using “Auto MPG” dataset that was taken from the StatLib library which is maintained at Carnegie Mellon University. The dataset was used in the 1983 American Statistical Association Exposition [17].

There are eight attributes and one numeric class “mpg”. Figure 5.11 is the output dataset of the confidence prediction. M5P method is used for class and confidence prediction here.



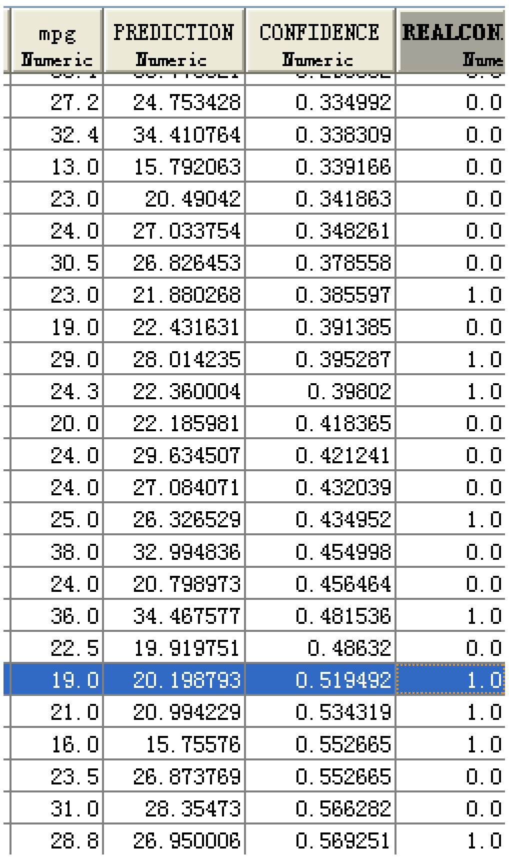
car-name Nominal	PREDICTION Numeric	CONFIDENCE Numeric
amc ho...	18.285253	0.652767
honda ...	33.894969	0.660657
plymou...	29.29761	0.672435
plymou...	29.46367	0.672435
oldsmo...	21.626552	0.692338
chevro...	27.499655	0.692338
ford g...	22.63931	0.692574
toyota...	25.83876	0.708924
plymou...	21.461963	0.753347
oldsmo...	21.660858	0.758683
toyota...	31.755641	0.759268
chevro...	21.339718	0.765902
peugeo...	22.891878	0.776761
amc ma...	15.546475	0.791856
ford m...	19.549519	0.811985
peugeo...	23.368484	0.932828
amc ma...	15.56028	0.938041
datsun...	30.544357	0.998945

Figure 5.4 Predictive class value and confidence



car-name Nominal	mpg Numeric	PREDICTION Numeric	CONFIDENCE Numeric
peugeo...	27.2	24.753428	0.334992
honda ...	32.4	34.410764	0.338309
chevro...	13.0	15.792063	0.339166
plymou...	23.0	20.49042	0.341863
fiat 128	24.0	27.033754	0.348261
volksw...	30.5	26.826453	0.378558
ford p...	23.0	21.880268	0.385597
peugeo...	19.0	22.431631	0.391385
fiat 128	29.0	28.014235	0.395287
amc co...	24.3	22.360004	0.39802
chevro...	20.0	22.185981	0.418365
honda ...	24.0	29.634507	0.421241
datsun...	24.0	27.084071	0.432039
toyota...	25.0	26.326529	0.434952
honda ...	38.0	32.994836	0.454998
ford m...	24.0	20.798973	0.456464
honda ...	36.0	34.467577	0.481536
amc ho...	22.5	19.919751	0.48632
amc ho...	19.0	20.198793	0.519492
ford m...	21.0	20.994229	0.534319
amc ma...	16.0	15.75576	0.552665
chevro...	23.5	26.873769	0.552665
toyota...	31.0	28.35473	0.566282
chevro...	28.8	26.950006	0.569251

(a)



mpg Numeric	PREDICTION Numeric	CONFIDENCE Numeric	REALCON Nume
27.2	24.753428	0.334992	0.0
32.4	34.410764	0.338309	0.0
13.0	15.792063	0.339166	0.0
23.0	20.49042	0.341863	0.0
24.0	27.033754	0.348261	0.0
30.5	26.826453	0.378558	0.0
23.0	21.880268	0.385597	1.0
19.0	22.431631	0.391385	0.0
29.0	28.014235	0.395287	1.0
24.3	22.360004	0.39802	1.0
20.0	22.185981	0.418365	0.0
24.0	29.634507	0.421241	0.0
24.0	27.084071	0.432039	0.0
25.0	26.326529	0.434952	1.0
38.0	32.994836	0.454998	0.0
24.0	20.798973	0.456464	0.0
36.0	34.467577	0.481536	1.0
22.5	19.919751	0.48632	0.0
19.0	20.198793	0.519492	1.0
21.0	20.994229	0.534319	1.0
16.0	15.75576	0.552665	1.0
23.5	26.873769	0.552665	0.0
31.0	28.35473	0.566282	0.0
28.8	26.950006	0.569251	1.0

(b)

Figure 5.5: Different view of numeric confidence prediction

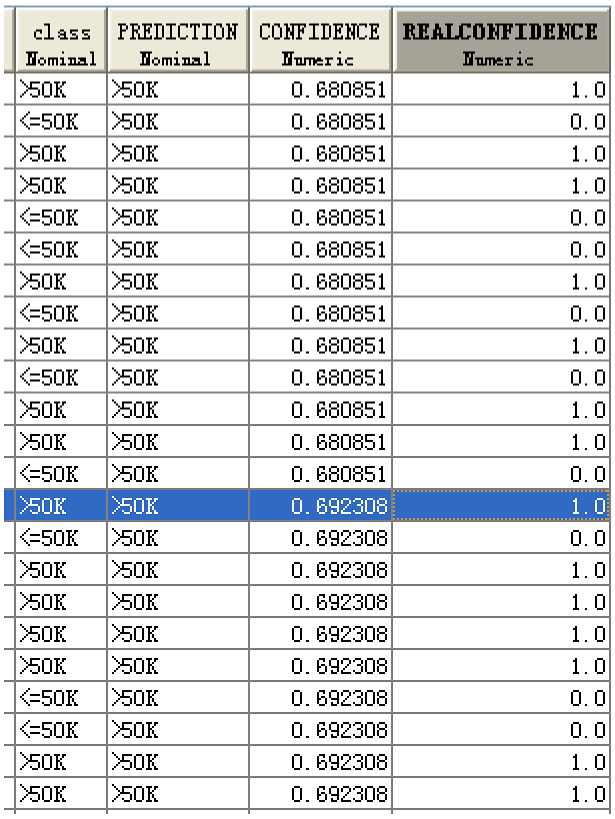
Figure 5.5 (a) shows the class value (mpg), predictive value for class and predictive confidence, while Figure 5.5 (b) indicates all data attributes above and also the real confidence assigned by program to show the difference directly. Confidence 0 means this prediction for class value cannot be trust, otherwise confidence 1 implies the correct prediction. From these middle datasets, we can make the following conclusion.

Confidence Threshold	Correctly Prediction Rate
Confidence >= 51.9%	98.03%
Confidence < 51.9%	14.29%

Table 5.6: Confidence accuracy

Another example uses the famous “adult” dataset to determine whether a person makes over 50K a year [18]. There are 14 nominal and numeric attributes, one nominal class and 17000 instances in this experiment. Bayesian network using *K2* as search algorithm with max three parents per node is used to predict nominal class value and DecisionTable method is adopted to predict the confidence.

Figure 5.6 illustrates the comparison of predictive results and real values including confidence.



class	PREDICTION	CONFIDENCE	REALCONFIDENCE
Nominal	Nominal	Numeric	Numeric
>50K	>50K	0.680851	1.0
<=50K	>50K	0.680851	0.0
>50K	>50K	0.680851	1.0
>50K	>50K	0.680851	1.0
<=50K	>50K	0.680851	0.0
<=50K	>50K	0.680851	0.0
>50K	>50K	0.680851	1.0
<=50K	>50K	0.680851	0.0
>50K	>50K	0.680851	1.0
<=50K	>50K	0.680851	0.0
>50K	>50K	0.680851	1.0
>50K	>50K	0.680851	1.0
<=50K	>50K	0.680851	0.0
>50K	>50K	0.692308	1.0
<=50K	>50K	0.692308	0.0
>50K	>50K	0.692308	1.0
>50K	>50K	0.692308	1.0
>50K	>50K	0.692308	1.0
>50K	>50K	0.692308	1.0
<=50K	>50K	0.692308	0.0
<=50K	>50K	0.692308	0.0
>50K	>50K	0.692308	1.0
>50K	>50K	0.692308	1.0

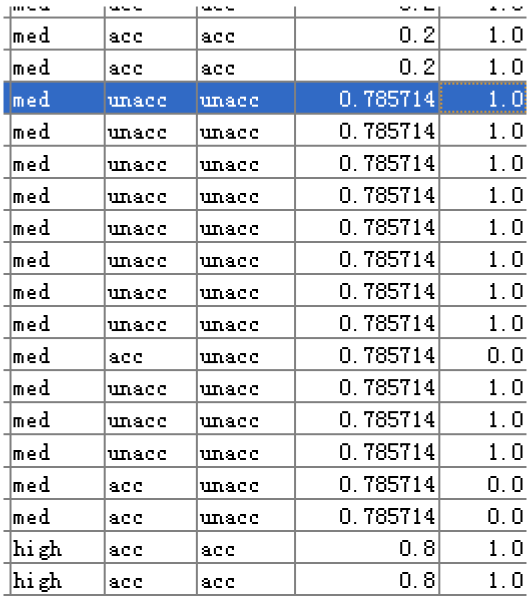
Figure 5.6: Mid-dataset for predictive class and confidence

Confidence Threshold	Correctly Prediction Rate
Confidence >= 69.2%	97.41%
Confidence < 69.2% and Confidence >= 52.8%	74.82%
Confidence < 52.8%	36.70%

Table 5.7: Confidence accuracy

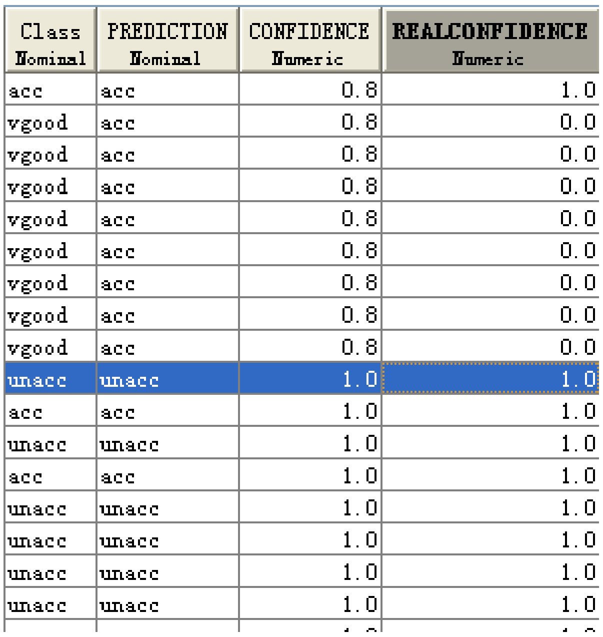
This confidence accuracy table is calculated approximately but different confidence decision threshold may have a large impact on the correctly prediction rate.

The following experiment is based on "Car Evaluation" dataset, which was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: Expert system for decision making [19]. There are 6 nominal attributes and one nominal class to decide whether a car is acceptable or not. Also Bayesian network using *K2* as search algorithm with max three parents per node and DecisionTable method are adopted here shown in Figure 5.7.



med	acc	acc	0.2	1.0
med	acc	acc	0.2	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	acc	unacc	0.785714	0.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	unacc	unacc	0.785714	1.0
med	acc	unacc	0.785714	0.0
med	acc	unacc	0.785714	0.0
high	acc	acc	0.8	1.0
high	acc	acc	0.8	1.0

(a)



Class Nominal	PREDICTION Nominal	CONFIDENCE Numeric	REALCONFIDENCE Numeric
acc	acc	0.8	1.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
vgood	acc	0.8	0.0
unacc	unacc	1.0	1.0
acc	acc	1.0	1.0
unacc	unacc	1.0	1.0
acc	acc	1.0	1.0
unacc	unacc	1.0	1.0
unacc	unacc	1.0	1.0
unacc	unacc	1.0	1.0
unacc	unacc	1.0	1.0

(b)

Figure 5.7: Difference segments of mid-dataset for predictive class and confidence

Confidence Threshold	Average Correctly Prediction Rate
Confidence >= 1%	100.00%
Confidence < 1% and Confidence >= 78.57%	79.79%
Confidence < 78.57%	13.33%

Table 5.8: Confidence accuracy

From these general confidence prediction experiments, we can make further conclusion that confidence can be divided according to different confidence level, which means we can set the threshold automatically and within diverse levels, confidence can result in different correctly prediction rate. According to these results, two confidence threshold can divide the correctly prediction rate into three types: 98% - very good, 75% - good, and 30% or less than 30% - poor. The more different level we create, more accurately we can illustrate the confidence prediction.

Chapter 6

Conclusions

6.1 Conclusions

In this paper, we have shown some important Machine Learning techniques and presented SysWeka platform and experimental evaluation on the performance of this framework. From this SysWeka platform, we can conclude that although linear regression has been widely used in many fields to build models with successful results, it cannot produce benefit outcome in our scenarios. By contrast, Bayesian network acts as a more proper choice and performs much better.

Moreover, confidence prediction is presented and developed to measure the accuracy of predictions, which gives us another opportunity to make a further decision whether to trust the predictive result or not.

Besides, Importance-Aware linear regression has been proposed and derived by mathematics. Experiments evaluation also shows a different view of the Importance-Aware dataset, which indicates a promising application usage in the future work. Instances can be specified by different importance and thus may create more valuable results in further studies.

6.2 Future work

This platform is mainly using these three methods: linear regression, M5P (Model tree) and Bayesian network for prediction and comparison. In the future work, more Machine Learning techniques will be presented and evaluated to study the practical performance in the same scenario. And common testing functionality should be added into this framework that involves training set testing, cross validation, using test set and percentage split testing. Also Importance-Aware linear regression and confidence usage can be further studied.

6.3 Acknowledgements

I am very grateful to Jordi Torres and Ricard Gavaldà for teaching me and giving me plenty of advice and reference on my study and master thesis. Also thanks to Josep Lluís Berral for giving me some tips on working here and to Javier Alonso for his previous work that I can continue this research work.

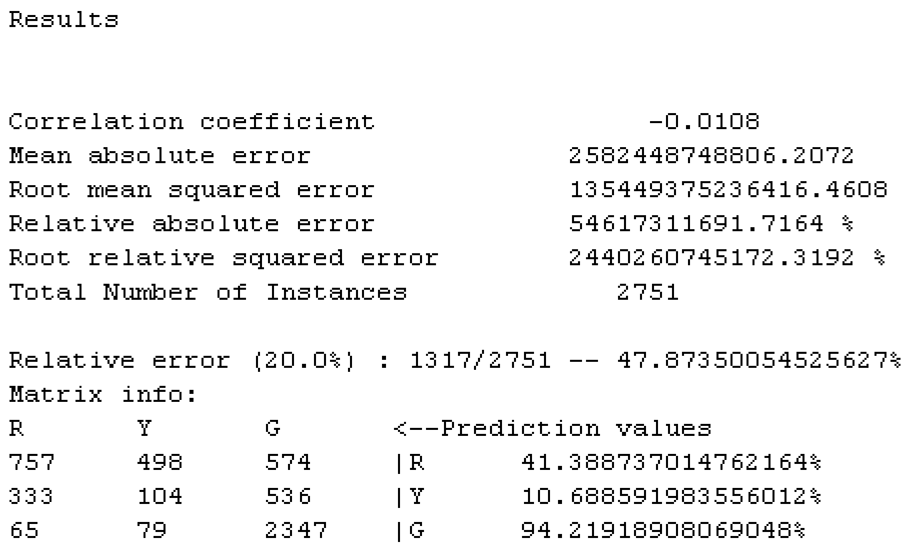
Chapter 7

Appendix

7.1 Model structures

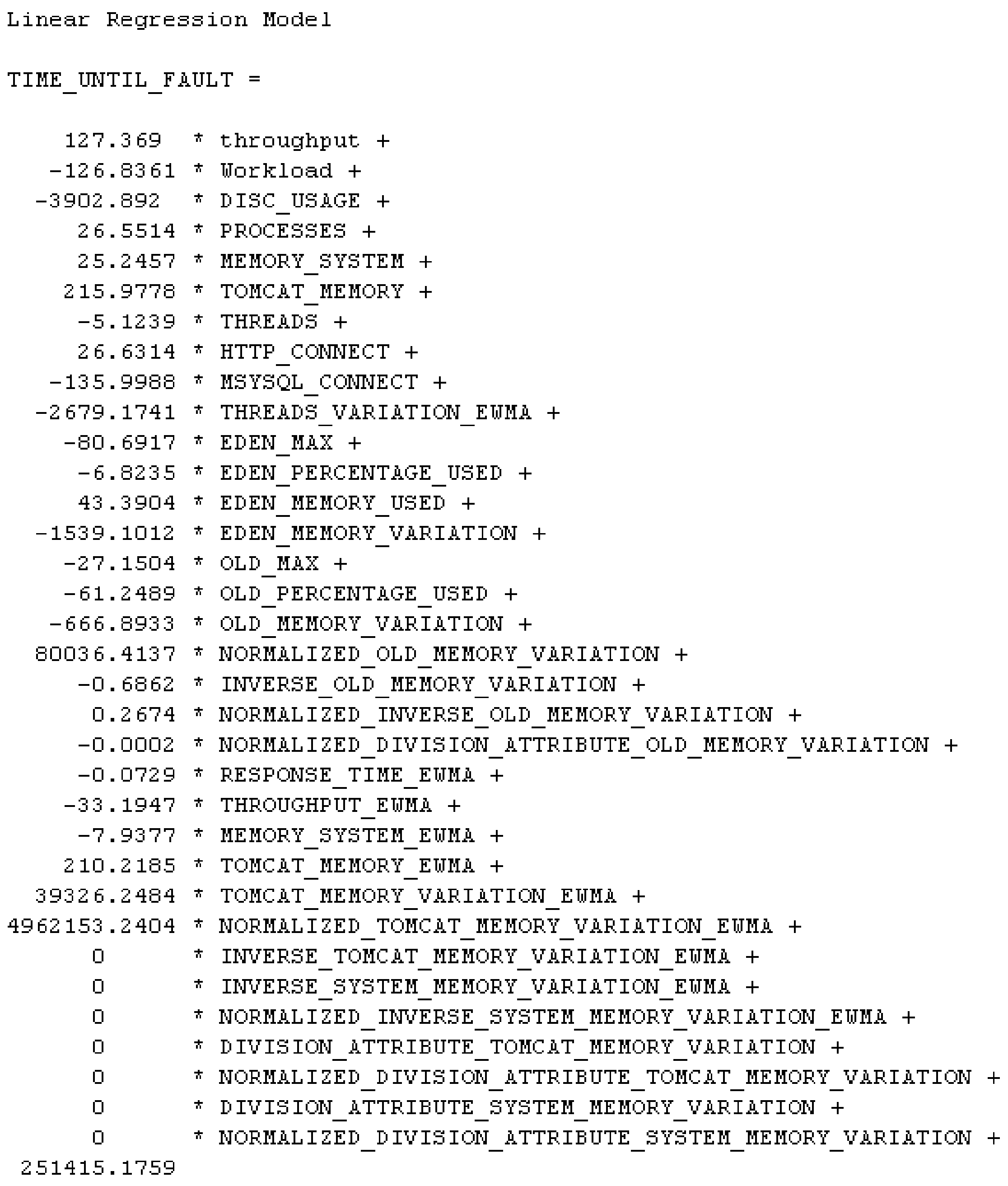
Linear Regression Model:

Time taken to build model: 0.594 seconds



```
Results


Correlation coefficient                   -0.0108
Mean absolute error                  2582448748806.2072
Root mean squared error              135449375236416.4608
Relative absolute error              54617311691.7164 %
Root relative squared error          2440260745172.3192 %
Total Number of Instances               2751

Relative error (20.0%) : 1317/2751 -- 47.87350054525627%
Matrix info:
R       Y       G       <--Prediction values
757     498     574     |R      41.388737014762164%
333     104     536     |Y      10.688591983556012%
65      79      2347    |G      94.21918908069048%
```

Figure 7.1: Linear regression prediction results

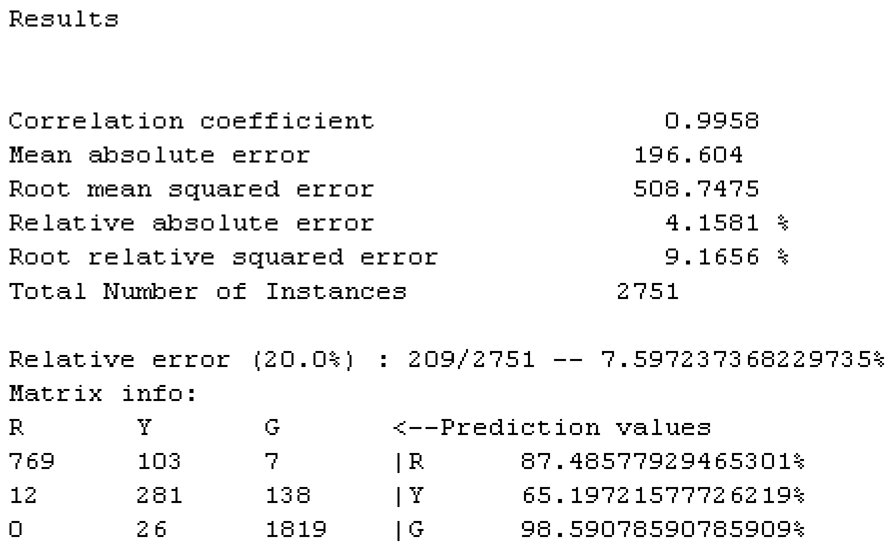


```
Linear Regression Model

TIME_UNTIL_FAULT =

    127.369   * throughput +
   -126.8361 * Workload +
  -3902.892   * DISC_USAGE +
     26.5514 * PROCESSES +
     25.2457 * MEMORY_SYSTEM +
    215.9778 * TOMCAT_MEMORY +
     -5.1239 * THREADS +
     26.6314 * HTTP_CONNECT +
   -135.9988 * MSYSQL_CONNECT +
  -2679.1741 * THREADS_VARIATION_EWMA +
    -80.6917 * EDEN_MAX +
     -6.8235 * EDEN_PERCENTAGE_USED +
     43.3904 * EDEN_MEMORY_USED +
  -1539.1012 * EDEN_MEMORY_VARIATION +
    -27.1504 * OLD_MAX +
    -61.2489 * OLD_PERCENTAGE_USED +
   -666.8933 * OLD_MEMORY_VARIATION +
  80036.4137 * NORMALIZED_OLD_MEMORY_VARIATION +
     -0.6862 * INVERSE_OLD_MEMORY_VARIATION +
      0.2674 * NORMALIZED_INVERSE_OLD_MEMORY_VARIATION +
     -0.0002 * NORMALIZED_DIVISION_ATTRIBUTE_OLD_MEMORY_VARIATION +
     -0.0729 * RESPONSE_TIME_EWMA +
    -33.1947 * THROUGHPUT_EWMA +
     -7.9377 * MEMORY_SYSTEM_EWMA +
    210.2185 * TOMCAT_MEMORY_EWMA +
  39326.2484 * TOMCAT_MEMORY_VARIATION_EWMA +
4962153.2404 * NORMALIZED_TOMCAT_MEMORY_VARIATION_EWMA +
      0       * INVERSE_TOMCAT_MEMORY_VARIATION_EWMA +
      0       * INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0       * NORMALIZED_INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0       * DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0       * NORMALIZED_DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0       * DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
      0       * NORMALIZED_DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
 251415.1759
```

Figure 7.2: Linear regression prediction models

M5P Model:

Time taken to build model: 1.438 seconds



```
Results


Correlation coefficient                  0.9958
Mean absolute error                    196.604
Root mean squared error                508.7475
Relative absolute error                  4.1581 %
Root relative squared error              9.1656 %
Total Number of Instances             2751

Relative error (20.0%) : 209/2751 -- 7.597237368229735%
Matrix info:
R       Y       G       <--Prediction values
769     103     7       |R      87.48577929465301%
12      281     138     |Y      65.19721577726219%
0       26      1819    |G      98.59078590785909%
```

Figure 7.3: M5P prediction results

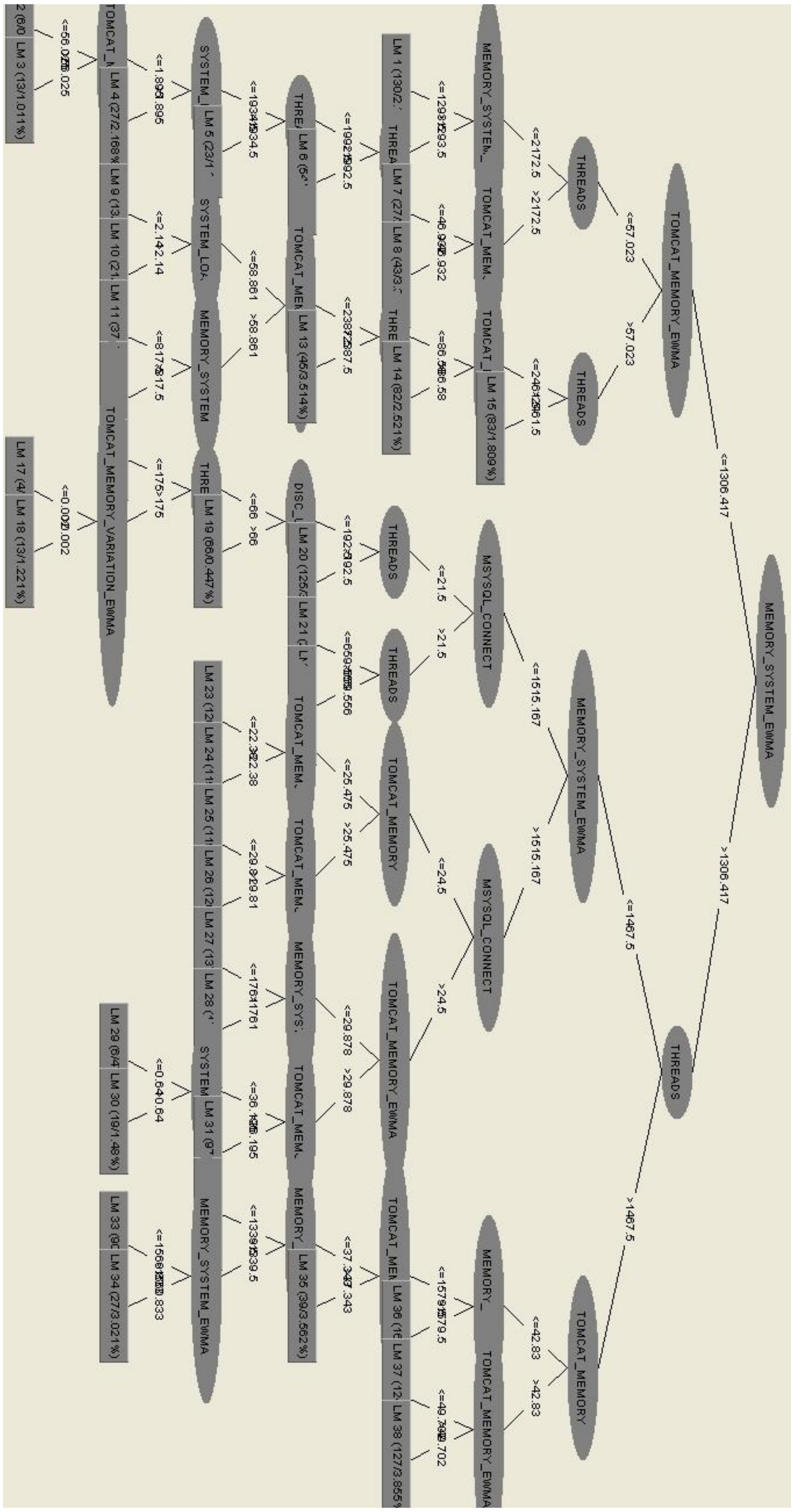
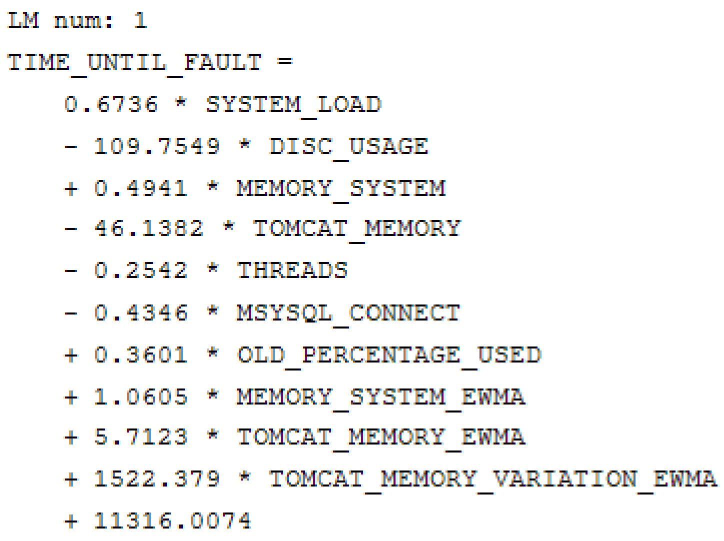


Figure 7.4: M5P prediction model tree with 38 leaves

Each leave contains one linear regression. The following is leaf No.1.

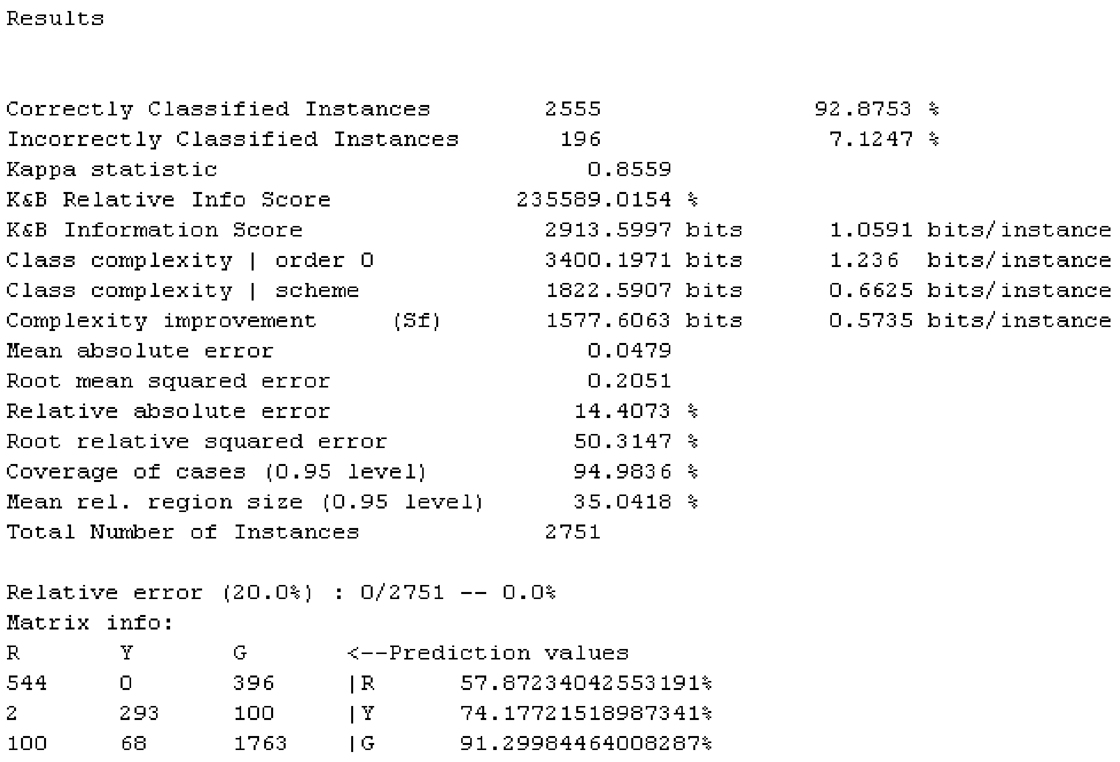


```
LM num: 1
TIME_UNTIL_FAULT =
    0.6736 * SYSTEM_LOAD
    - 109.7549 * DISC_USAGE
    + 0.4941 * MEMORY_SYSTEM
    - 46.1382 * TOMCAT_MEMORY
    - 0.2542 * THREADS
    - 0.4346 * MSYSQL_CONNECT
    + 0.3601 * OLD_PERCENTAGE_USED
    + 1.0605 * MEMORY_SYSTEM_EWMA
    + 5.7123 * TOMCAT_MEMORY_EWMA
    + 1522.379 * TOMCAT_MEMORY_VARIATION_EWMA
    + 11316.0074
```

Figure 7.5: Leaf No.1 linear regression

Bayesian network Model:

Time taken to build model: 0.313 seconds.



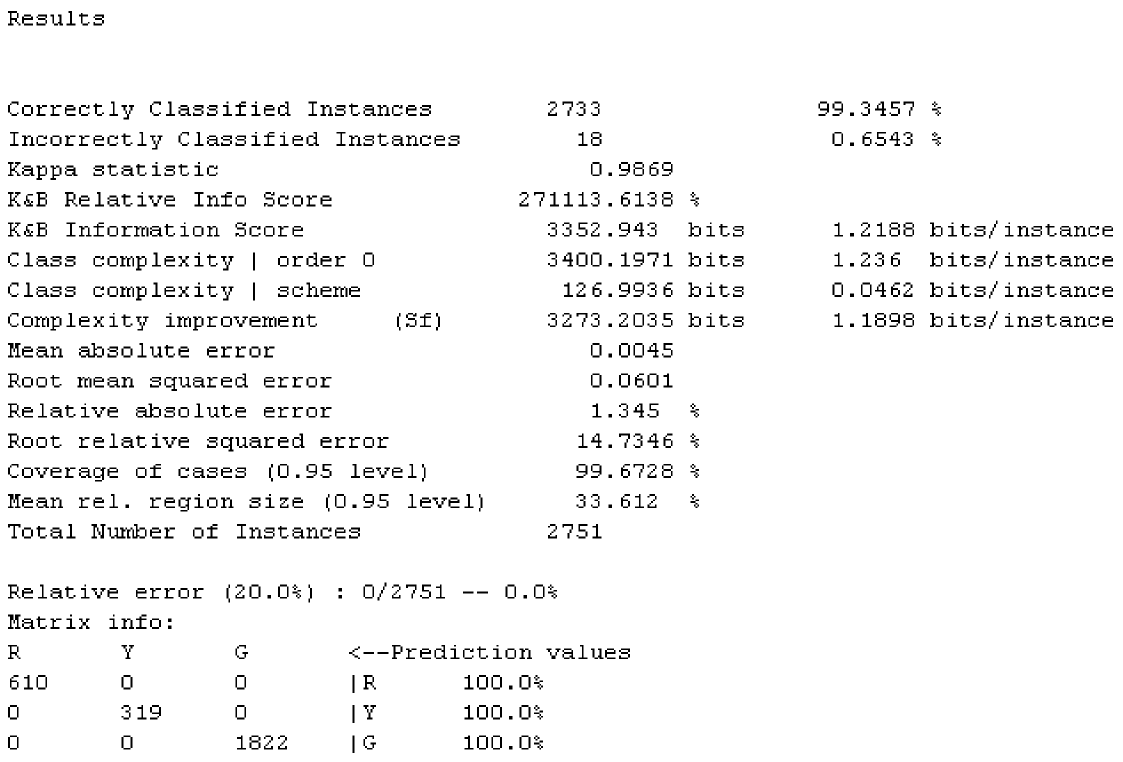
```
Results


Correctly Classified Instances        2555               92.8753 %
Incorrectly Classified Instances       196                7.1247 %
Kappa statistic                          0.8559
K&B Relative Info Score             235589.0154 %
K&B Information Score                 2913.5997 bits      1.0591 bits/instance
Class complexity | order 0            3400.1971 bits      1.236  bits/instance
Class complexity | scheme             1822.5907 bits      0.6625 bits/instance
Complexity improvement     (Sf)       1577.6063 bits      0.5735 bits/instance
Mean absolute error                      0.0479
Root mean squared error                  0.2051
Relative absolute error                 14.4073 %
Root relative squared error             50.3147 %
Coverage of cases (0.95 level)          94.9836 %
Mean rel. region size (0.95 level)      35.0418 %
Total Number of Instances             2751

Relative error (20.0%) : 0/2751 -- 0.0%
Matrix info:
R       Y       G       <--Prediction values
544     0       396     |R      57.87234042553191%
2       293     100     |Y      74.17721518987341%
100     68      1763    |G      91.29984464008287%
```

Figure 7.6: Bayesian network predictions results

This Bayesian network uses default values of Weka and each node has only one parent-class. So this is a tree with one root (TIME_WARNING) and 49 nodes. Each node is the son of the root.

7.2 Bayesian network prediction results

Time taken to build model: 1.375 seconds



```
Results


Correctly Classified Instances        2733               99.3457 %
Incorrectly Classified Instances        18                0.6543 %
Kappa statistic                          0.9869
K&B Relative Info Score             271113.6138 %
K&B Information Score                 3352.943  bits      1.2188 bits/instance
Class complexity | order 0            3400.1971 bits      1.236  bits/instance
Class complexity | scheme              126.9936 bits      0.0462 bits/instance
Complexity improvement     (Sf)       3273.2035 bits      1.1898 bits/instance
Mean absolute error                      0.0045
Root mean squared error                  0.0601
Relative absolute error                  1.345  %
Root relative squared error             14.7346 %
Coverage of cases (0.95 level)          99.6728 %
Mean rel. region size (0.95 level)      33.612  %
Total Number of Instances             2751

Relative error (20.0%) : 0/2751 -- 0.0%
Matrix info:
R       Y       G       <--Prediction values
610     0       0       |R      100.0%
0       319     0       |Y      100.0%
0       0       1822    |G      100.0%
```

Figure 7.7: Bayesian network predictions using training set test with max three parents per node

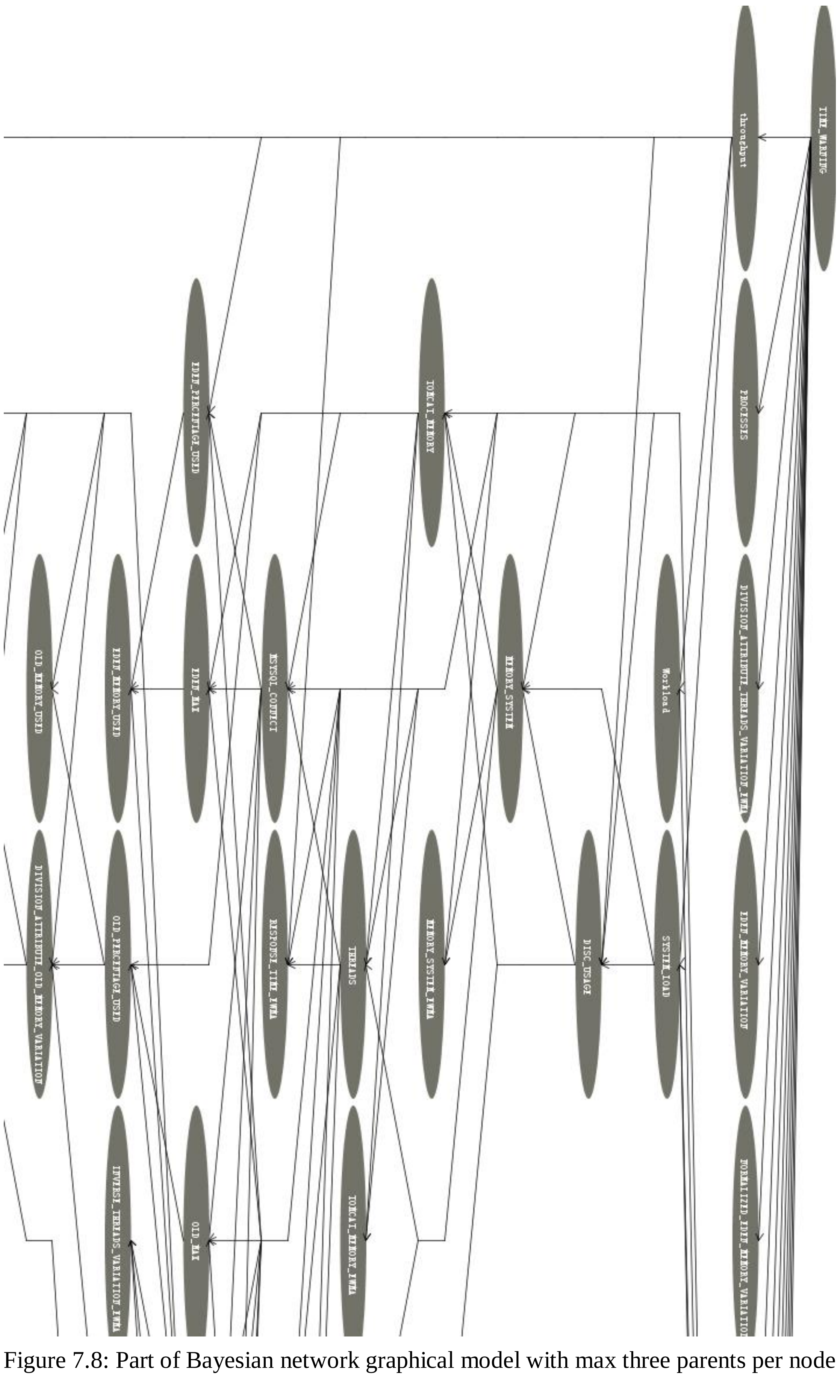
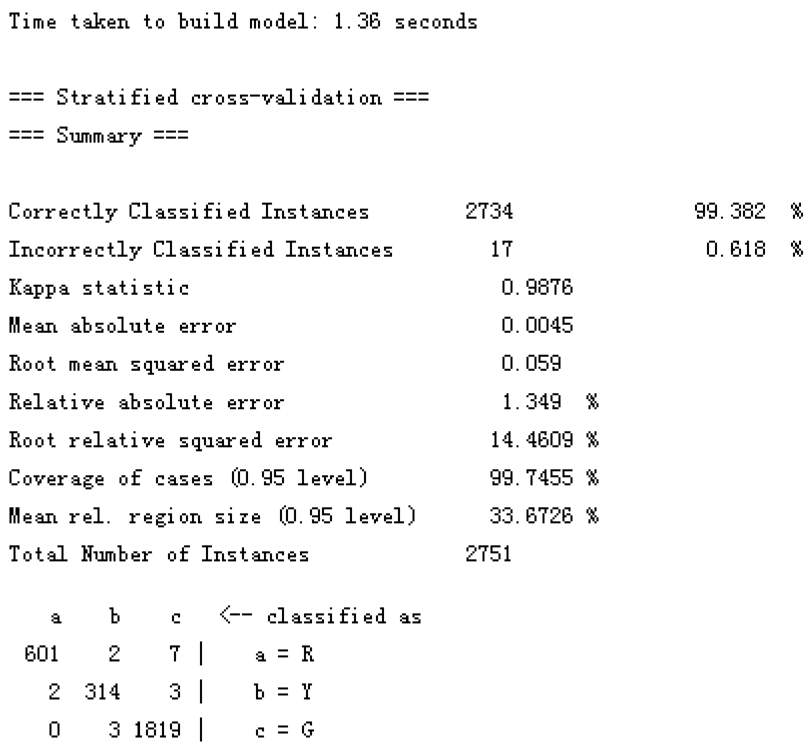
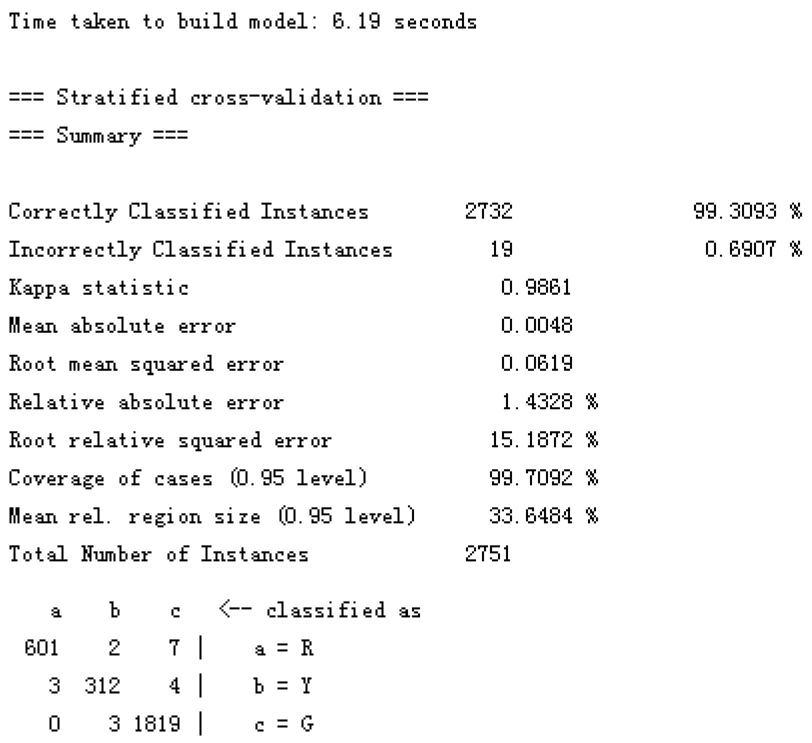


Figure 7.8: Part of Bayesian network graphical model with max three parents per node



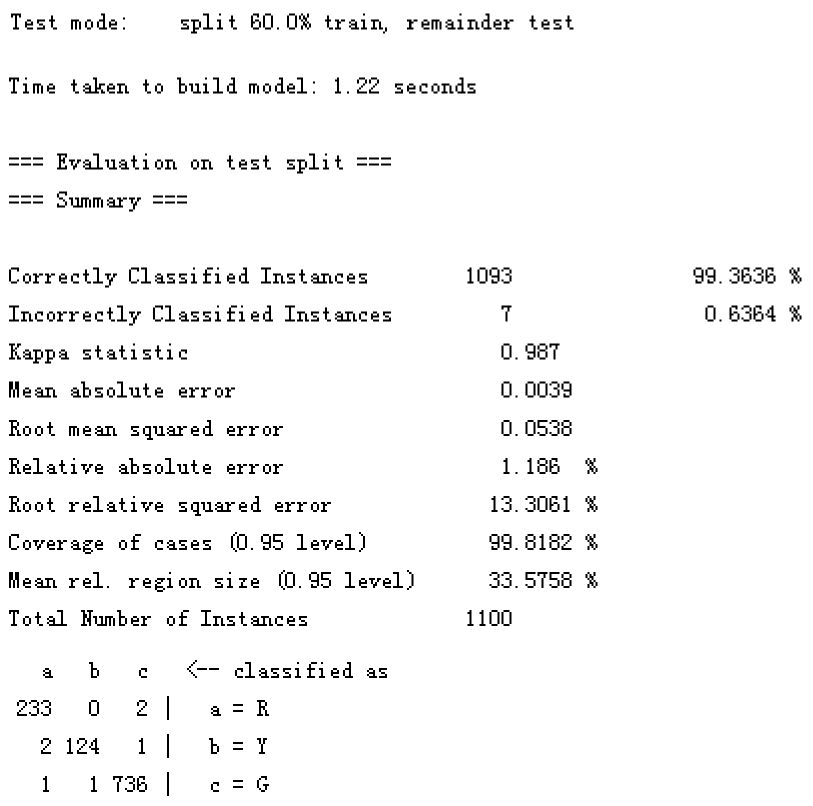
```
Time taken to build model: 1.36 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       2734               99.382  %
Incorrectly Classified Instances       17                0.618  %
Kappa statistic                         0.9876
Mean absolute error                     0.0045
Root mean squared error                 0.059
Relative absolute error                 1.349  %
Root relative squared error            14.4609 %
Coverage of cases (0.95 level)         99.7455 %
Mean rel. region size (0.95 level)     33.6726 %
Total Number of Instances            2751

   a    b    c   <-- classified as
 601    2    7 |    a = R
   2  314    3 |    b = Y
   0    3 1819 |    c = G
```

Figure 7.9: Bayesian network predictions using 10-fold cross-validation with max three parents per node



```
Time taken to build model: 6.19 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       2732               99.3093 %
Incorrectly Classified Instances       19                0.6907 %
Kappa statistic                         0.9861
Mean absolute error                     0.0048
Root mean squared error                 0.0619
Relative absolute error                 1.4328 %
Root relative squared error            15.1872 %
Coverage of cases (0.95 level)         99.7092 %
Mean rel. region size (0.95 level)     33.6484 %
Total Number of Instances            2751

   a    b    c   <-- classified as
 601    2    7 |    a = R
   3  312    4 |    b = Y
   0    3 1819 |    c = G
```

Figure 7.10: Bayesian network predictions using 10-fold cross-validation with max five parents per node

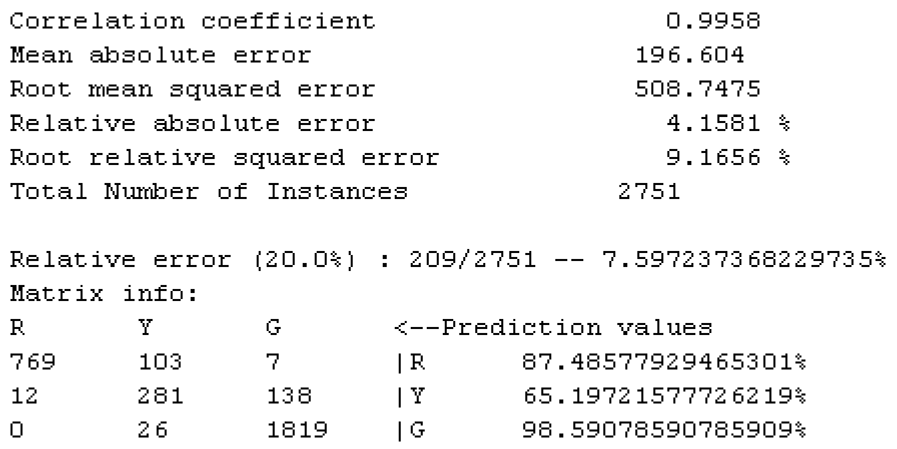


```
Test mode:    split 60.0% train, remainder test

Time taken to build model: 1.22 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances        1093               99.3636 %
Incorrectly Classified Instances         7                0.6364 %
Kappa statistic                          0.987
Mean absolute error                      0.0039
Root mean squared error                  0.0538
Relative absolute error                  1.186  %
Root relative squared error             13.3061 %
Coverage of cases (0.95 level)          99.8182 %
Mean rel. region size (0.95 level)      33.5758 %
Total Number of Instances             1100

   a   b   c   <-- classified as
 233   0   2 |   a = R
   2 124   1 |   b = Y
   1   1 736 |   c = G
```

Figure 7.11: Bayesian network predictions with max three parents per node using 60% percentage split

7.3 Confidence prediction results with varied methods

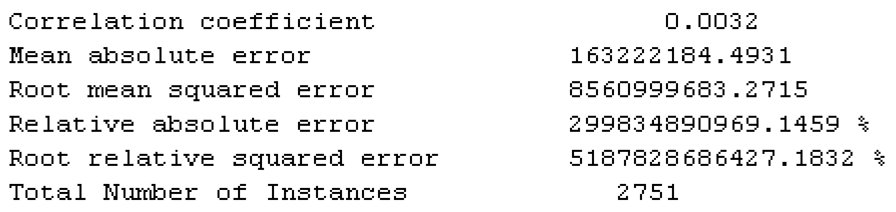
M5P to predict class value



```
Correlation coefficient                   0.9958
Mean absolute error                     196.604
Root mean squared error                 508.7475
Relative absolute error                   4.1581 %
Root relative squared error               9.1656 %
Total Number of Instances              2751

Relative error (20.0%) : 209/2751 -- 7.597237368229735%
Matrix info:
R       Y       G       <--Prediction values
769     103     7       |R      87.48577929465301%
12      281     138     |Y      65.19721577726219%
0       26      1819    |G      98.59078590785909%
```

Figure 7.12: M5P prediction result

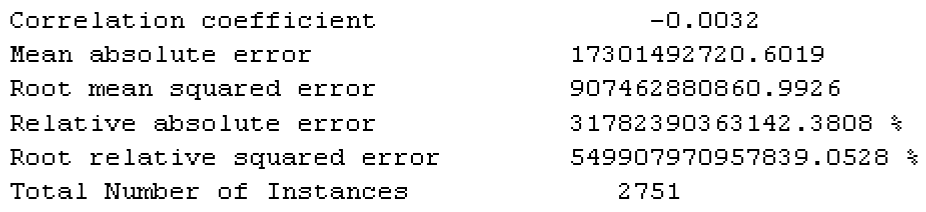
Linear Regression to predict confidence



```
Correlation coefficient                  0.0032
Mean absolute error               163222184.4931
Root mean squared error           8560999683.2715
Relative absolute error           299834890969.1459 %
Root relative squared error       5187828686427.1832 %
Total Number of Instances            2751
```

Figure 7.13: Confidence predictions with linear regression

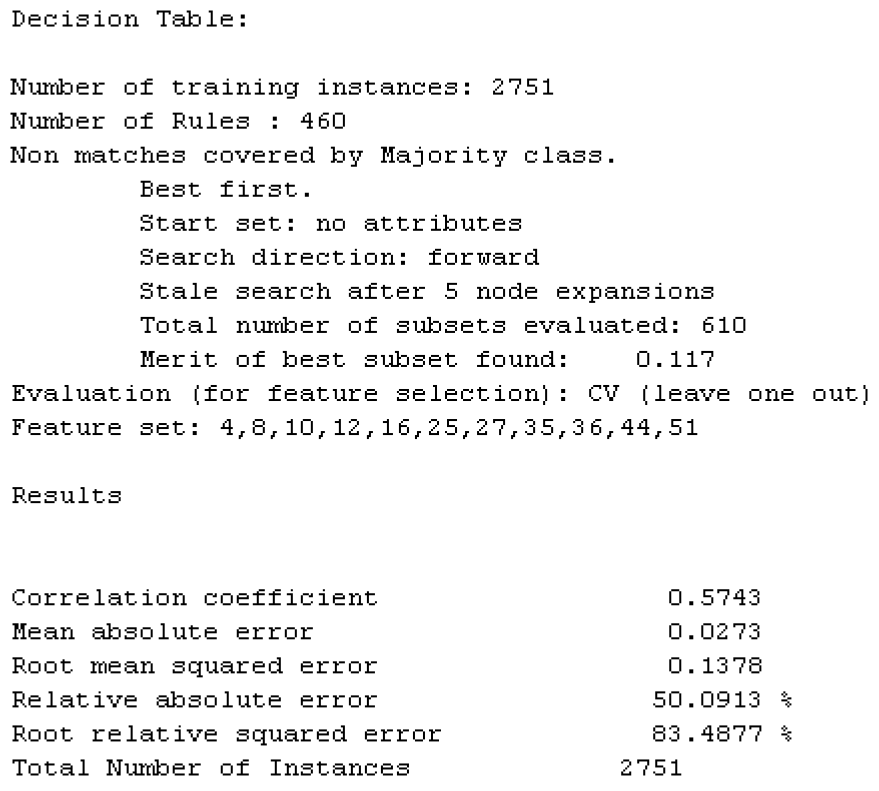
M5P to predict confidence



```
Correlation coefficient                  -0.0032
Mean absolute error               17301492720.6019
Root mean squared error           907462880860.9926
Relative absolute error           31782390363142.3808 %
Root relative squared error       549907970957839.0528 %
Total Number of Instances            2751
```

Figure 7.14: Confidence predictions with M5P

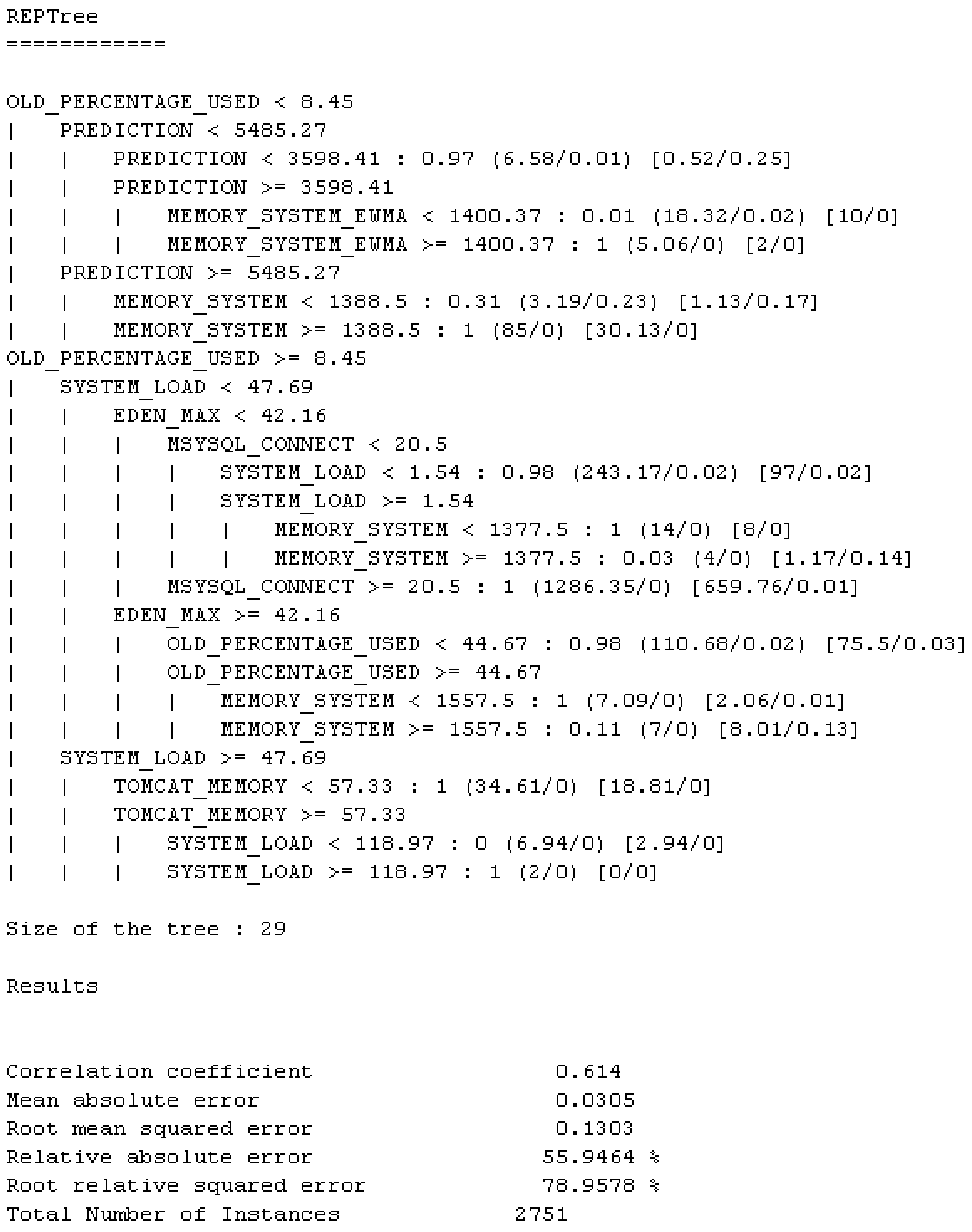
Decision Table to predict confidence



```
Decision Table:

Number of training instances: 2751
Number of Rules : 460
Non matches covered by Majority class.
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 610
        Merit of best subset found:    0.117
Evaluation (for feature selection): CV (leave one out)
Feature set: 4,8,10,12,16,25,27,35,36,44,51

Results


Correlation coefficient                    0.5743
Mean absolute error                        0.0273
Root mean squared error                    0.1378
Relative absolute error                   50.0913 %
Root relative squared error               83.4877 %
Total Number of Instances               2751
```

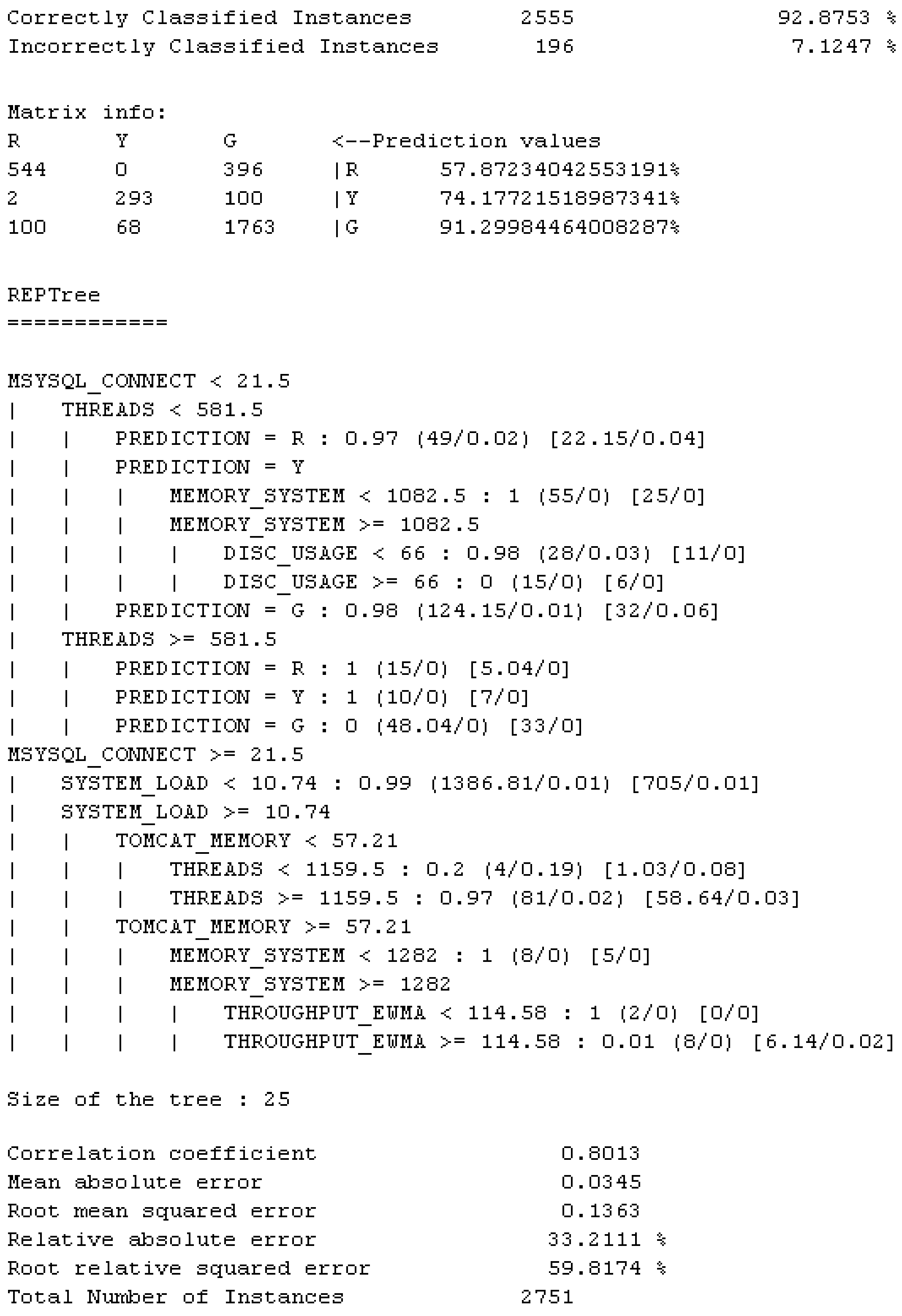
Figure 7.15: Confidence predictions with decision table

REPTree to predict confidence



```
REPTree
============

OLD_PERCENTAGE_USED < 8.45
|   PREDICTION < 5485.27
|   |   PREDICTION < 3598.41 : 0.97 (6.58/0.01) [0.52/0.25]
|   |   PREDICTION >= 3598.41
|   |   |   MEMORY_SYSTEM_EWMA < 1400.37 : 0.01 (18.32/0.02) [10/0]
|   |   |   MEMORY_SYSTEM_EWMA >= 1400.37 : 1 (5.06/0) [2/0]
|   PREDICTION >= 5485.27
|   |   MEMORY_SYSTEM < 1388.5 : 0.31 (3.19/0.23) [1.13/0.17]
|   |   MEMORY_SYSTEM >= 1388.5 : 1 (85/0) [30.13/0]
OLD_PERCENTAGE_USED >= 8.45
|   SYSTEM_LOAD < 47.69
|   |   EDEN_MAX < 42.16
|   |   |   MSYSQL_CONNECT < 20.5
|   |   |   |   SYSTEM_LOAD < 1.54 : 0.98 (243.17/0.02) [97/0.02]
|   |   |   |   SYSTEM_LOAD >= 1.54
|   |   |   |   |   MEMORY_SYSTEM < 1377.5 : 1 (14/0) [8/0]
|   |   |   |   |   MEMORY_SYSTEM >= 1377.5 : 0.03 (4/0) [1.17/0.14]
|   |   |   MSYSQL_CONNECT >= 20.5 : 1 (1286.35/0) [659.76/0.01]
|   |   EDEN_MAX >= 42.16
|   |   |   OLD_PERCENTAGE_USED < 44.67 : 0.98 (110.68/0.02) [75.5/0.03]
|   |   |   OLD_PERCENTAGE_USED >= 44.67
|   |   |   |   MEMORY_SYSTEM < 1557.5 : 1 (7.09/0) [2.06/0.01]
|   |   |   |   MEMORY_SYSTEM >= 1557.5 : 0.11 (7/0) [8.01/0.13]
|   SYSTEM_LOAD >= 47.69
|   |   TOMCAT_MEMORY < 57.33 : 1 (34.61/0) [18.81/0]
|   |   TOMCAT_MEMORY >= 57.33
|   |   |   SYSTEM_LOAD < 118.97 : 0 (6.94/0) [2.94/0]
|   |   |   SYSTEM_LOAD >= 118.97 : 1 (2/0) [0/0]

Size of the tree : 29

Results


Correlation coefficient                  0.614
Mean absolute error                      0.0305
Root mean squared error                  0.1303
Relative absolute error                 55.9464 %
Root relative squared error             78.9578 %
Total Number of Instances             2751
```

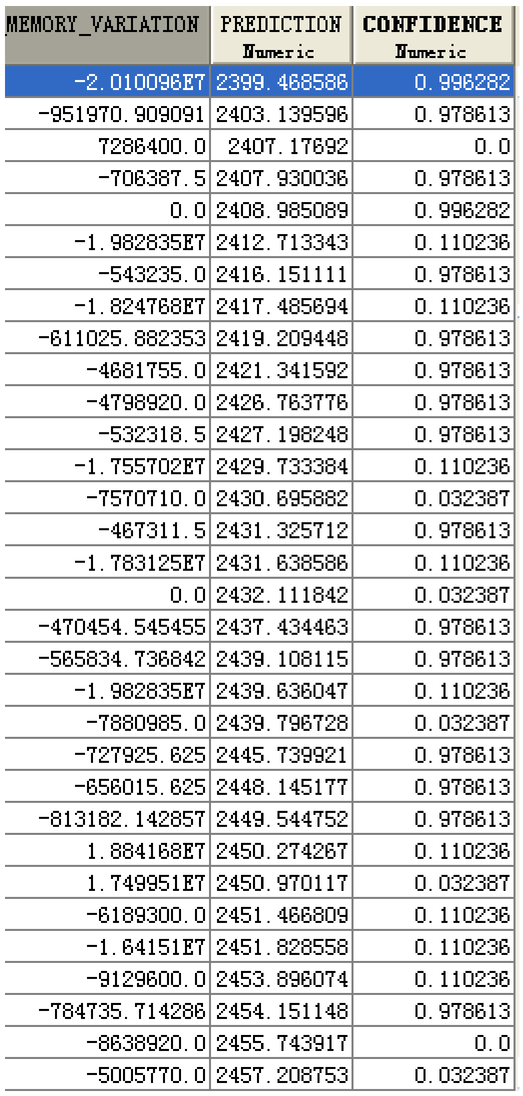
Figure 7.16: Confidence predictions with REPTree



```
Correctly Classified Instances        2555               92.8753 %
Incorrectly Classified Instances       196                7.1247 %

Matrix info:
R       Y       G       <--Prediction values
544     0       396     |R      57.87234042553191%
2       293     100     |Y      74.17721518987341%
100     68      1763    |G      91.29984464008287%

REPTree
============

MSYSQL_CONNECT < 21.5
|   THREADS < 581.5
|   |   PREDICTION = R : 0.97 (49/0.02) [22.15/0.04]
|   |   PREDICTION = Y
|   |   |   MEMORY_SYSTEM < 1082.5 : 1 (55/0) [25/0]
|   |   |   MEMORY_SYSTEM >= 1082.5
|   |   |   |   DISC_USAGE < 66 : 0.98 (28/0.03) [11/0]
|   |   |   |   DISC_USAGE >= 66 : 0 (15/0) [6/0]
|   |   PREDICTION = G : 0.98 (124.15/0.01) [32/0.06]
|   THREADS >= 581.5
|   |   PREDICTION = R : 1 (15/0) [5.04/0]
|   |   PREDICTION = Y : 1 (10/0) [7/0]
|   |   PREDICTION = G : 0 (48.04/0) [33/0]
MSYSQL_CONNECT >= 21.5
|   SYSTEM_LOAD < 10.74 : 0.99 (1386.81/0.01) [705/0.01]
|   SYSTEM_LOAD >= 10.74
|   |   TOMCAT_MEMORY < 57.21
|   |   |   THREADS < 1159.5 : 0.2 (4/0.19) [1.03/0.08]
|   |   |   THREADS >= 1159.5 : 0.97 (81/0.02) [58.64/0.03]
|   |   TOMCAT_MEMORY >= 57.21
|   |   |   MEMORY_SYSTEM < 1282 : 1 (8/0) [5/0]
|   |   |   MEMORY_SYSTEM >= 1282
|   |   |   |   THROUGHPUT_EWMA < 114.58 : 1 (2/0) [0/0]
|   |   |   |   THROUGHPUT_EWMA >= 114.58 : 0.01 (8/0) [6.14/0.02]

Size of the tree : 25

Correlation coefficient                   0.8013
Mean absolute error                       0.0345
Root mean squared error                   0.1363
Relative absolute error                  33.2111 %
Root relative squared error              59.8174 %
Total Number of Instances              2751
```

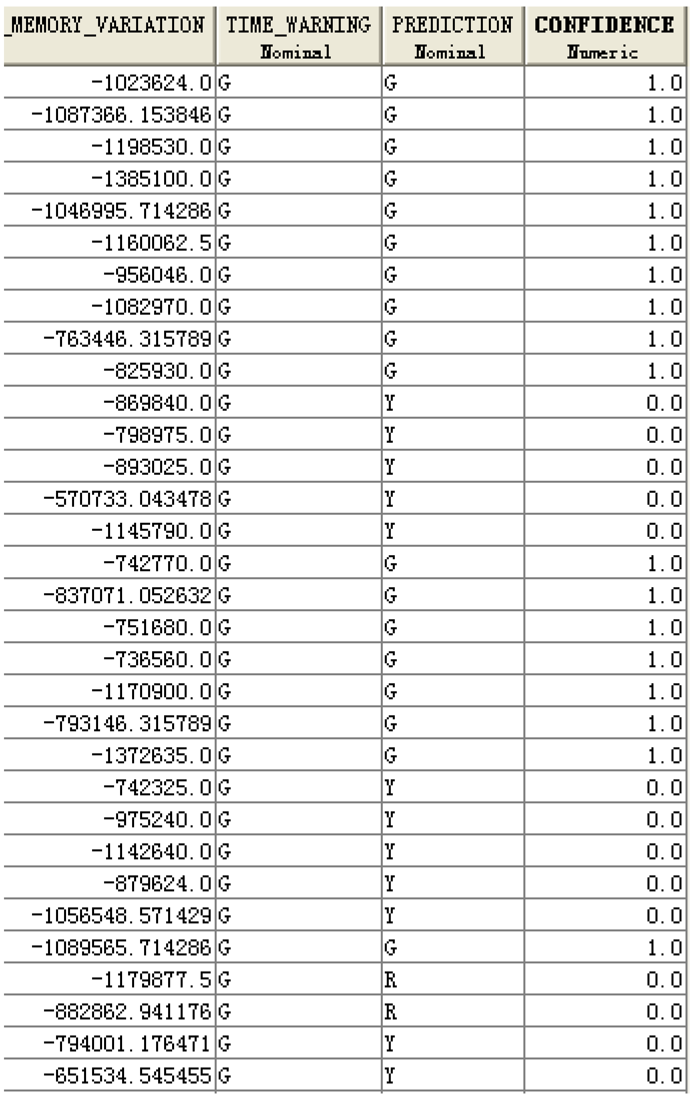
Figure 7.17: Bayesian network prediction with max one parent per node to predict class value and REPTree to predict confidence

7.4 Numeric and nominal class confidence prediction



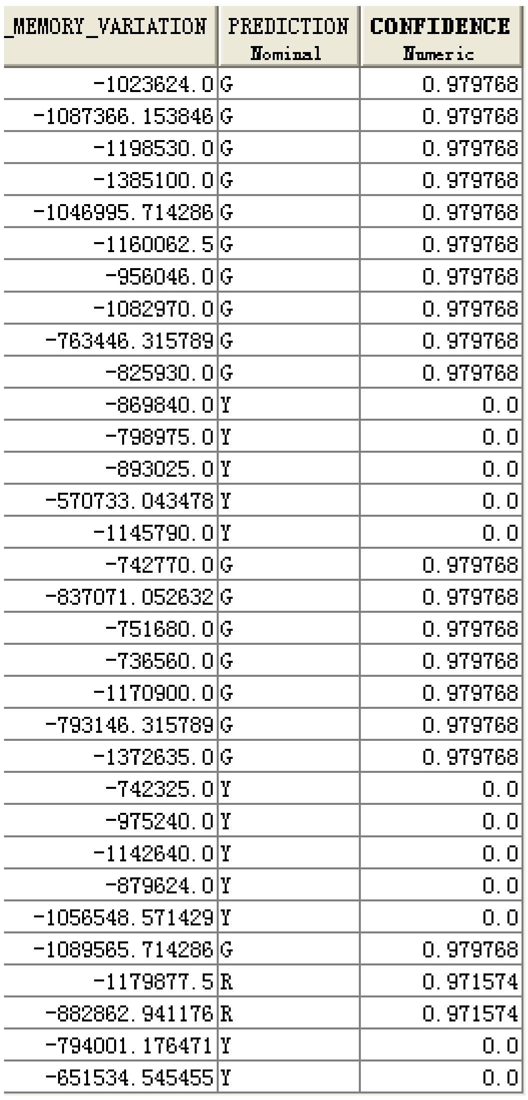
MEMORY_VARIATION	PREDICTION Numeric	CONFIDENCE Numeric
-2.010096E7	2399.468586	0.996282
-951970.909091	2403.139596	0.978613
7286400.0	2407.17692	0.0
-706387.5	2407.930036	0.978613
0.0	2408.985089	0.996282
-1.982835E7	2412.713343	0.110236
-543235.0	2416.151111	0.978613
-1.824768E7	2417.485694	0.110236
-611025.882353	2419.209448	0.978613
-4681755.0	2421.341592	0.978613
-4798920.0	2426.763776	0.978613
-532318.5	2427.198248	0.978613
-1.755702E7	2429.733384	0.110236
-7570710.0	2430.695882	0.032387
-467311.5	2431.325712	0.978613
-1.783125E7	2431.638586	0.110236
0.0	2432.111842	0.032387
-470454.545455	2437.434463	0.978613
-565834.736842	2439.108115	0.978613
-1.982835E7	2439.636047	0.110236
-7880985.0	2439.796728	0.032387
-727925.625	2445.739921	0.978613
-656015.625	2448.145177	0.978613
-813182.142857	2449.544752	0.978613
1.884168E7	2450.274267	0.110236
1.749951E7	2450.970117	0.032387
-6189300.0	2451.466809	0.110236
-1.64151E7	2451.828558	0.110236
-9129600.0	2453.896074	0.110236
-784735.714286	2454.151148	0.978613
-8638920.0	2455.743917	0.0
-5005770.0	2457.208753	0.032387

Figure 7.18: Final numeric confidence prediction results



_MEMORY_VARIATION	TIME_WARNING Nominal	PREDICTION Nominal	CONFIDENCE Numeric
-1023624.0	G	G	1.0
-1087366.153846	G	G	1.0
-1198530.0	G	G	1.0
-1385100.0	G	G	1.0
-1046995.714286	G	G	1.0
-1160062.5	G	G	1.0
-956046.0	G	G	1.0
-1082970.0	G	G	1.0
-763446.315789	G	G	1.0
-825930.0	G	G	1.0
-869840.0	G	Y	0.0
-798975.0	G	Y	0.0
-893025.0	G	Y	0.0
-570733.043478	G	Y	0.0
-1145790.0	G	Y	0.0
-742770.0	G	G	1.0
-837071.052632	G	G	1.0
-751680.0	G	G	1.0
-736560.0	G	G	1.0
-1170900.0	G	G	1.0
-793146.315789	G	G	1.0
-1372635.0	G	G	1.0
-742325.0	G	Y	0.0
-975240.0	G	Y	0.0
-1142640.0	G	Y	0.0
-879624.0	G	Y	0.0
-1056548.571429	G	Y	0.0
-1089565.714286	G	G	1.0
-1179877.5	G	R	0.0
-882862.941176	G	R	0.0
-794001.176471	G	Y	0.0
-651534.545455	G	Y	0.0

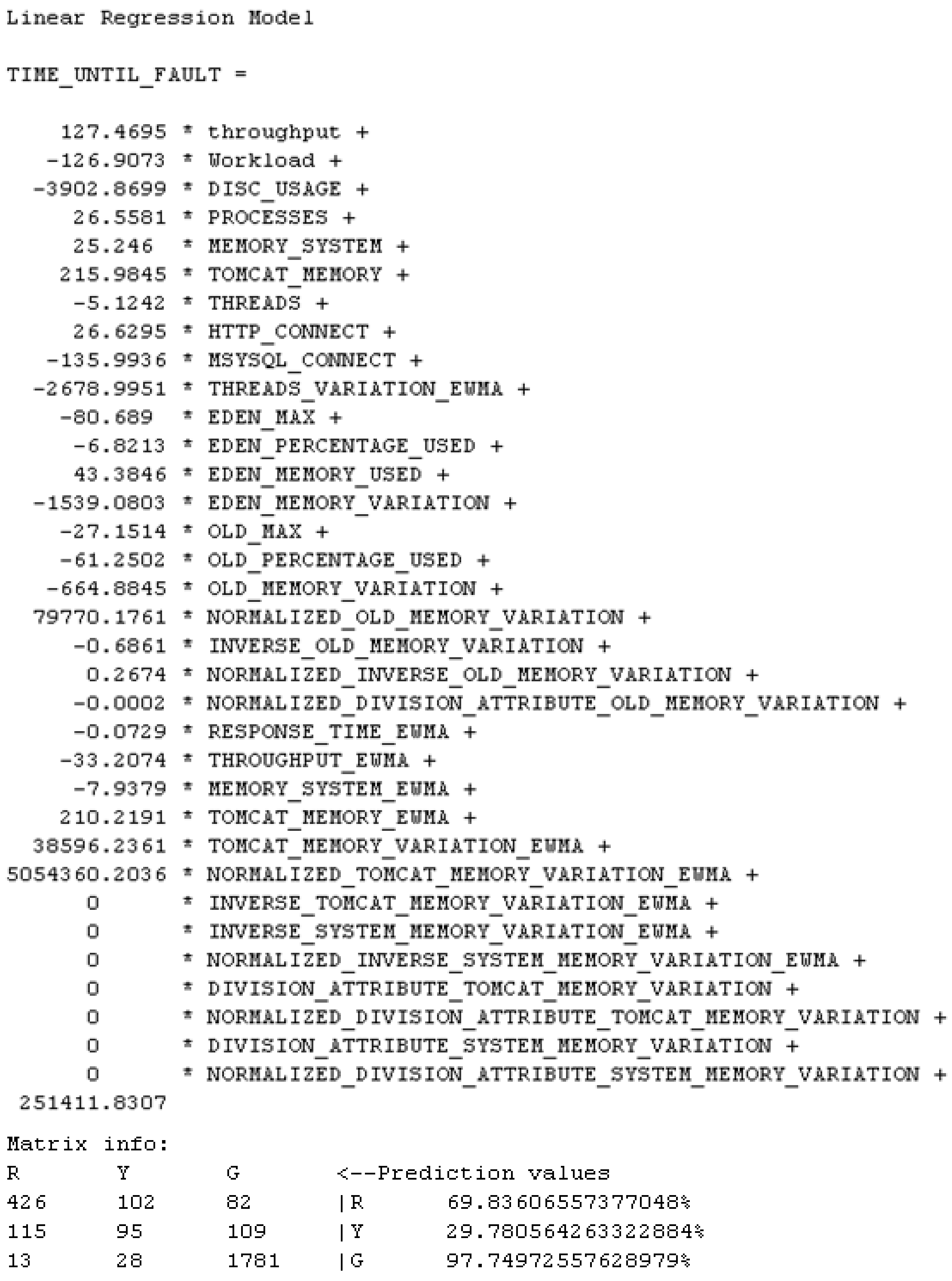
Figure 7.19: Training confidence mid dataset with nominal class



_MEMORY_VARIATION	PREDICTION Nominal	CONFIDENCE Numeric
-1023624.0	G	0.979768
-1087366.153846	G	0.979768
-1198530.0	G	0.979768
-1385100.0	G	0.979768
-1046995.714286	G	0.979768
-1160062.5	G	0.979768
-956046.0	G	0.979768
-1082970.0	G	0.979768
-763446.315789	G	0.979768
-825930.0	G	0.979768
-869840.0	Y	0.0
-798975.0	Y	0.0
-893025.0	Y	0.0
-570733.043478	Y	0.0
-1145790.0	Y	0.0
-742770.0	G	0.979768
-837071.052632	G	0.979768
-751680.0	G	0.979768
-736560.0	G	0.979768
-1170900.0	G	0.979768
-793146.315789	G	0.979768
-1372635.0	G	0.979768
-742325.0	Y	0.0
-975240.0	Y	0.0
-1142640.0	Y	0.0
-879624.0	Y	0.0
-1056548.571429	Y	0.0
-1089565.714286	G	0.979768
-1179877.5	R	0.971574
-882862.941176	R	0.971574
-794001.176471	Y	0.0
-651534.545455	Y	0.0

Figure 7.20: Final confidence prediction dataset with nominal class

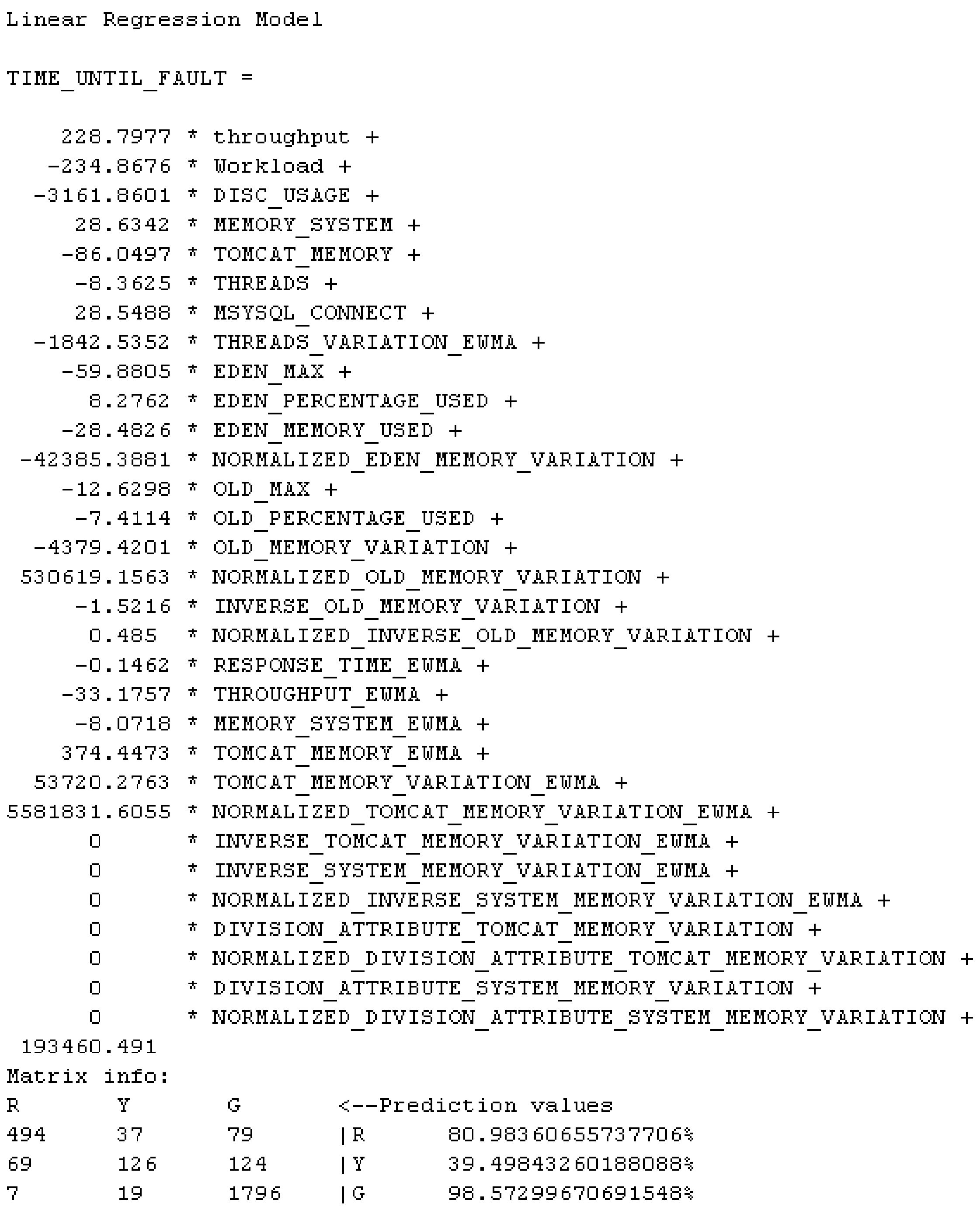
7.5 Importance-Aware Linear Regression model



```
Linear Regression Model

TIME_UNTIL_FAULT =

    127.4695 * throughput +
   -126.9073 * Workload +
  -3902.8699 * DISC_USAGE +
     26.5581 * PROCESSES +
     25.246  * MEMORY_SYSTEM +
    215.9845 * TOMCAT_MEMORY +
     -5.1242 * THREADS +
     26.6295 * HTTP_CONNECT +
   -135.9936 * MSYSQL_CONNECT +
  -2678.9951 * THREADS_VARIATION_EWMA +
    -80.689  * EDEN_MAX +
     -6.8213 * EDEN_PERCENTAGE_USED +
     43.3846 * EDEN_MEMORY_USED +
  -1539.0803 * EDEN_MEMORY_VARIATION +
    -27.1514 * OLD_MAX +
    -61.2502 * OLD_PERCENTAGE_USED +
   -664.8845 * OLD_MEMORY_VARIATION +
  79770.1761 * NORMALIZED_OLD_MEMORY_VARIATION +
     -0.6861 * INVERSE_OLD_MEMORY_VARIATION +
      0.2674 * NORMALIZED_INVERSE_OLD_MEMORY_VARIATION +
     -0.0002 * NORMALIZED_DIVISION_ATTRIBUTE_OLD_MEMORY_VARIATION +
     -0.0729 * RESPONSE_TIME_EWMA +
    -33.2074 * THROUGHPUT_EWMA +
     -7.9379 * MEMORY_SYSTEM_EWMA +
    210.2191 * TOMCAT_MEMORY_EWMA +
  38596.2361 * TOMCAT_MEMORY_VARIATION_EWMA +
5054360.2036 * NORMALIZED_TOMCAT_MEMORY_VARIATION_EWMA +
      0      * INVERSE_TOMCAT_MEMORY_VARIATION_EWMA +
      0      * INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0      * NORMALIZED_INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0      * DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0      * NORMALIZED_DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0      * DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
      0      * NORMALIZED_DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
 251411.8307

Matrix info:
R       Y       G       <--Prediction values
426     102     82      |R      69.83606557377048%
115     95      109     |Y      29.780564263322884%
13      28      1781    |G      97.74972557628979%
```

Figure 7.21: General linear regression model with training set test

Importance-Aware Linear Regression:



```
Linear Regression Model

TIME_UNTIL_FAULT =

    228.7977 * throughput +
   -234.8676 * Workload +
  -3161.8601 * DISC_USAGE +
     28.6342 * MEMORY_SYSTEM +
    -86.0497 * TOMCAT_MEMORY +
     -8.3625 * THREADS +
     28.5488 * MSYSQL_CONNECT +
  -1842.5352 * THREADS_VARIATION_EWMA +
    -59.8805 * EDEN_MAX +
      8.2762 * EDEN_PERCENTAGE_USED +
    -28.4826 * EDEN_MEMORY_USED +
 -42385.3881 * NORMALIZED_EDEN_MEMORY_VARIATION +
    -12.6298 * OLD_MAX +
     -7.4114 * OLD_PERCENTAGE_USED +
  -4379.4201 * OLD_MEMORY_VARIATION +
 530619.1563 * NORMALIZED_OLD_MEMORY_VARIATION +
     -1.5216 * INVERSE_OLD_MEMORY_VARIATION +
      0.485  * NORMALIZED_INVERSE_OLD_MEMORY_VARIATION +
     -0.1462 * RESPONSE_TIME_EWMA +
    -33.1757 * THROUGHPUT_EWMA +
     -8.0718 * MEMORY_SYSTEM_EWMA +
    374.4473 * TOMCAT_MEMORY_EWMA +
  53720.2763 * TOMCAT_MEMORY_VARIATION_EWMA +
5581831.6055 * NORMALIZED_TOMCAT_MEMORY_VARIATION_EWMA +
      0      * INVERSE_TOMCAT_MEMORY_VARIATION_EWMA +
      0      * INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0      * NORMALIZED_INVERSE_SYSTEM_MEMORY_VARIATION_EWMA +
      0      * DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0      * NORMALIZED_DIVISION_ATTRIBUTE_TOMCAT_MEMORY_VARIATION +
      0      * DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
      0      * NORMALIZED_DIVISION_ATTRIBUTE_SYSTEM_MEMORY_VARIATION +
 193460.491
Matrix info:
R       Y       G       <--Prediction values
494     37      79      |R      80.98360655737706%
69      126     124     |Y      39.49843260188088%
7       19      1796    |G      98.57299670691548%
```

Figure 7.22: Importance-Aware Linear Regression with training set test

Bibliography

[1] Green Gird Consortium, 2010. http://www.thegreengrid.org/

[2] Josep Lluís Berral, Inigo Goiri, Ramon Nou, Ferran Julia, Jordi Guitart, Ricard Gavalda and Jordi Torres. Towards energy-aware scheduling in data centers using machine learning. First Intl. Conf. on Energy-Efficient Computing and Networking. Passau (Germany), April 13-15, 2010.

[3] Javier Alonso, Josep Lluís Berral, Ricard Gavaldà and Jordi Torres. Adaptive on-line software aging prediction based on Machine Learning. The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010).

[4] Javier Alonso, R. Gavalda and Jordi Torres. Predicting Web Server Crashes: A Case Study in Comparing Prediction Algorithms. Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems. Pages: 264-269. 2009.

[5] Dustin Amrhein, Scott Quint. Cloud computing for the enterprise - Understanding cloud computing and related technologies: Part 1: Capturing the cloud. http://www.ibm.com/developerworks/websphere/techjournal/0904_amrhein/0904_amrhein.html, 2009.

[6] Liang-Jie Zhang, Carl K Chang, Ephraim Feig, Robert Grossman, Keynote Panel, Business Cloud: Bringing The Power of SOA and Cloud Computing, 2008 IEEE International Conference on Services Computing (SCC 2008), July 2008.

[7] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/

[8] Google Web Applications. http://www.google.com/apps

[9] Data mining From Wikipedia. http://en.wikipedia.org/wiki/Data_mining

[10] Weka, Data Mining with Open Source Machine Learning Software in Java. http://www.cs.waikato.ac.nz/ml/weka/

[11] Ian H.Witten, Eibe Frank. Data Mining Practical Machine Learning Tools and Techniques, 2005.

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

[13] "A Bayesian Method for the Induction of Probabilistic Networks from Data", Gregory F. Cooper and Edward Herskovits, Machine Learning 9, 1992.

[14] Tsamardinos, I., Aliferis, C., StatnikovA. Algorithms for Large Scale Markov Blanket Discovery. The 16th International FLAIRS Conference, St. Augustine, Florida, USA, 2003.

[15] L. Jiang, H. Zhang, Z. Cai and J. Su. Learning Tree Augmented Naive Bayes for Ranking. Proceedings of the 10th International Conference on Database Systems for Advanced Applications, 2005.

[16] Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985.

[17] Auto MPG Data Set. http://archive.ics.uci.edu/ml/datasets/Auto+MPG.

[18] Ron Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996.

[19] M. Bohanec and V. Rajkovic: Knowledge acquisition and explanation for multi-attribute decision making. In 8th Intl Workshop on Expert Systems and their Applications, Avignon, France. Pages: 59-78, 1988.