APPLICATION OF AN ENHANCED DECISION-TREE
LEARNING APPROACH FOR PREDICTION OF PETROLEUM
PRODUCTION

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science

In

Electronic Systems Engineering

University of Regina

By

Xiongmin Li

Regina, Saskatchewan

December, 2007

© Copyright 2007: Xiongmin Li

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-42352-3

Our file Notre référence

ISBN: 978-0-494-42352-3

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

UNIVERSITY OF REGINA
FACULTY OF GRADUATE STUDIES AND RESEARCH
SUPERVISORY AND EXAMINING COMMITTEE

Xiongmin Li, candidate for the degree of Master of Applied Science in Electronic Systems Engineering, has presented a thesis titled, ***Application of an Enhanced Decision-tree Learning Approach for Prediction of Petroleum Production***, in an oral examination held on December 4, 2007. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner:	Dr. Yiyu Yao, Department of Computer Science
Supervisor:	Dr. Christine Chan, Software Systems Engineering
Committee Member:	Dr. Nima Sarshar, Software Systems Engineering
Committee Member:	*Dr. Peter Gu, Petroleum Systems Engineering
Chair of Defense:	Dr. Nader Mahinpey, Industrial Systems Engineering

*Not present at defense

Abstract

Prediction of oil well production is important in estimating economic benefit of a well. However, this prediction task is difficult because of the complex subsurface conditions of wells. In response to the above problems, advancements in data mining technology, in recent years, has improved the ability for discovering information within a database that can then be used to support decisions. Data mining technology is a powerful AI tool that effectively extracts information from massive observational data sets, as well as discovers new and meaningful knowledge for the user.

In this thesis, we adopt an existing neural based decision-learning (NDT) model, which can obtain explicit information on the processing involved in generating predictions of oil production. In our experiment, the NDT model, which uses a neural network to extract the underlying attribute dependencies, was evaluated in comparison to the conventional C4.5 model on different kinds of data set. The results generated by the NDT model are found to be satisfactory.

As well, we explore modeling petroleum production data using the NDT model. We experiment in the modeling process by introducing different strategies and different parameter combinations. First, an overall oil production model is developed using three geoscience parameters (permeability, porosity and first shut-in pressure). Second, two different models, with different input parameters, are developed to predict production in the post water flooding stage only. The results of the above models indicate that the mechanisms used are somewhat superficial and these configurations may not allow the data-driven models to classify and predict oil production. Finally, a trend model is

developed in an attempt to improve the effectiveness and accuracy of the predictive model. The result shows that the trend model demonstrates an improved performance and is comparable to the artificial neural network.

Acknowledgement

I would like to thank my parents for their great, support and encouragement.

I am indebted to my supervisor, Dr. Christine W. Chan, for her guidance and support. She gave me the opportunity to do the research project on prediction of petroleum production, and gave me critical and valuable advice throughout all stages of my study and research.

I would like to thank the members of my thesis committee, Dr. Nima Sarsha and Dr. Peter Gu, for their time and insightful comments and suggestions.

I also thank the Faculty of Engineering, Faculty of Graduate Studies and Research of the University of Regina, the Natural Sciences and Engineering Research Council of Canada for their financial support, and Jon Hromek and Hahn H. Nguyen for assisting me at various stages of research.

Post Defense Acknowledgement

I would like to thank the external examiner Dr. Yiyu Yao for providing insightful feedback concerning my thesis. I would also like to thank defense chair Dr. Nader Mahinpey for presiding the defense.

Table of Contents

Abstract	i
Acknowledgement	iii
Post Defense Acknowledgement	iv
Table of Contents	v
List of Tables	ix
List of Figures	x
 Chapter 1 Introduction	 1
1.1 Problem Statement	1
1.2 Research Objectives and Contributions	2
1.3 Thesis Structure	3
 Chapter 2 Background: Literature on Methodology	 4
2.1 Overview of Classification	4
2.2 Overview of Prediction	6
2.3 System Framework	7
2.4 Applications of Classification and Prediction	11
2.5 General Approaches of Classification and Prediction	14
2.5.1 Decision Tree Learning	14
2.5.2 Regression-Based Analysis	15
2.5.2.1 Linear Regression	15

2.5.2.2 Nonlinear Regression Methods	17
2.5.3 Genetic Algorithms	18
2.6 Evaluating the Accuracy of a Classifier or Predictor	20
2.6.1 Cross-validation	20
2.6.2 Holdout Method	21
Chapter 3 Background Literature on Application Area	22
3.1 Current Methods in Petroleum Prediction	22
3.1.1 Conventional Methods: Curve Estimation	22
3.1.1.1 Arps Decline Analysis	23
3.1.1.2 Fetkovich Type Curve Analysis	24
3.1.1.3 Advanced Methods Used in Water Flooding Oil Fields	24
3.1.2 Intelligent Methods	25
3.1.2.1 Artificial Neural Network in Petroleum Prediction	25
3.1.2.2 Decision Tree Learning for Petroleum Engineering	26
3.1.2.3 Prediction Using Genetic Algorithms	28
Chapter 4 Methodology	29
4.1 Overview of Decision Tree Learning	29
4.1.1 Measurement of Attribute Selection	31
4.1.1.1 Information Gain	32
4.1.1.2 Gain Ratio	34
4.1.1.3 Gini Index	35
4.1.2 Tree Pruning	37

4.1.2.1 Prepruning	37
4.1.2.2 Post Pruning	37
4.1.2.3 From Tree to Rules	39
4.1.3 Limitations of Decision Tree Learning in Prediction of Oil Production	41
4.1.3.1 Categorical-type Dependencies	42
4.1.3.2 Numeric-type Dependencies	43
4.2 Overview of Artificial Neural Network	45
4.2.1 Artificial Neuron	47
4.2.2 Operation of Neural Network Process	48
4.2.2.1 Structure of Neural Network Model	49
4.2.2.2 Neural Network Learning Procedure	50
4.2.3 Weakness of Artificial Neuron Network	53
4.3 The Neuron-based Decision Tree Learning Model (NDT)	53
4.3.1 Motivation	53
4.3.2 Structure of the NDT Model	54
4.3.2.1 Categorical Data	57
4.3.2.2 Numerical Data	61
4.4 Implementation & Tools	65
4.5 Evaluation of the NDT Model: Using Non Petroleum Data Sets	69
4.5.1 Results Summary and Analysis	70
4.5.1.1 Pure Numeric-type Data Set	71
4.5.1.2 Pure Categorical-type Data Set	74

4.5.1.3 Mixed-type Data Set	76
4.5.2 Summary Over All Data Sets	78
4.5.3 Strengths and Limitations	79
Chapter 5 Application of NDT to Petroleum Production	81
5.1 Prediction of Petroleum Production	81
5.1.1 Domain Concepts	81
5.1.2 Study Area	83
5.1.3 Identification of Data Sets and Parameters	84
5.1.4 Data Preprocessing	85
5.2 Experiments and Results	85
5.2.1 Overall Oil Production model	86
5.2.2 Zooming-in: Post-water Flooding Model	88
5.2.2.1 Three Parameters Post-water Flooding Model	88
5.2.2.2 Four Parameters Post-water Flooding Model	90
5.2.3 Trend Predicting Model	91
Chapter 6 Conclusions	94
Chapter 7 Future Work	98
Bibliography	100
Appendix	108

List of Tables

2.1 Picnic dataset 1	6
2.2 Picnic dataset 2	16
4.1 Encoded data set	57
4.2 Normalized dataset for Categorical Type	58
4.3 Weights between the input attributes and hidden node	59
4.4 Distance among attributes	60
4.5 Combined transportation dataset	61
4.6 Normalized dataset for numeric type	62
4.7 Transformed numerical data Set	64
4.8 Result comparison	64
4.9 Description of data sets	70
4.10 Result summary for pure numeric-type data set	72
4.11 Result summary for pure categorical-type data set	75
4.12 Result summary for mixed-Type data set	77
5.1 Comparison of NDT model and C4.5 results	88
5.2 Comparison of NDT model and ANN results	89
5.3 Comparison of NDT model and ANN results	90
5.4 Experimental results of NDT	92
5.5 Experimental results of ANN	92

List of Figures

2.1. Classification process	5
2.2 Generic framework for classification and prediction tasks	9
2.3 A simple linear regression for picnic dataset with two classes	17
2.4 An example of decision boundaries learned by neural network for picnic data set	18
4.1 Pseudo-code of the C4.5 tree-construction algorithm	30
4.2 Decision tree from table 1	40
4.3 Univariate tree	42
4.4 Multivariate tree	43
4.5 Decision tree of table 2	44
4.6 Error graph of decision region and decision rules	45
4.7 Schematic diagram of two bipolar neurons	46
4.8. An artificial neuron architecture	47
4.9 A typical three-layered neural network model	49
4.10 Activation functions	51
4.11 The NDT model	56
4.12 Training results of the categorical data set	59
4.13 Average linkage diagram	60
4.14 Training results for the numerical data	63
4.15 System structure of the NDT neural training preprocessor	67
4.16 Screen of the neural training user interface	68

4.17 NDT learning process	69
4.18 Comparison of tree size complexity for numeric datasets	73
4.19 Comparison of decision rules for numeric datasets	73
4.20 Comparison of classification accuracy for numeric datasets	73
4.21 Comparisons of tree size for mixed-type datasets	77
4.22 Comparison of decision rules for mixed-type datasets	78
4.23 Comparison of classification accuracy for mixed-type datasets	78
4.24 % Improvement/degradation made by NDT for all data sets	79
5.1 Distribution of data in each classes	87

Chapter 1

Introduction

1.1 Problem Statement:

With the recent interest and enthusiasm in oil well production, the oil industry's need for predicting of petroleum production has significantly increased. Unfortunately, this kind of prediction task is extremely difficult because of complex subsurface conditions of reservoirs. Even two wells, located side by side in the same reservoir, may not have the same production [57]. In addition, core analysis data obtained from oil fields are very limited and tend to be biased. While it is commonly observed that the core analysis variables, which are significant factors in oil production, have some dependencies or correlation between them, there is no equation to describe or able to capture such a relationship. The traditional approach of using curve-fitting is complex and time-consuming, and the amount of petroleum data collected in databases today has far exceeded our ability to reduce and analyze data without the use of automated analysis techniques. Hence, an effective and intelligent approach to handle this kind of prediction task is required.

An enhanced decision tree learning approach called Neural-based decision tree (NDT) model is applied in this project in an attempt to investigate its performance in predicting petroleum production. This model is chosen because it is claimed that the NDT model can successfully capture the dependencies among attributes, and therefore, improve the accuracy and effectiveness of the prediction [24]. In our application, we chose five most significant parameters: permeability, porosity, first shut-in pressure, residual oil and saturation of water as input, and oil production as output for modeling. In

this chapter, we present the goals, motivations, and contributions of the project, and a roadmap for the organization of the thesis.

1.2 Research Objectives and Contributions

In recent years, artificial intelligence, in its many integrated flavors, has made solid steps towards becoming more accepted in the mainstream of the oil and gas industry [2]. The performance of the NDT model is studied when it is applied for prediction of petroleum production in the Weyburn Field in Saskatchewan, Canada. The model is derived from artificial neural network and decision tree learning algorithms, where the artificial neural network is used to discover the attributes' dependence and the decision tree-learning algorithm is applied to predict the petroleum production. Details of the methodology are described in the following chapters.

In this chapter, we present the goals, motivations, contributions of the project, and a roadmap to the organization of the thesis.

In summary, the main objectives of this research are:

- To address a weakness in decision tree-learning algorithms, such as C4.5's inability to deal with attribute interdependencies.
- To adopt the Neural-based Decision Tree (NDT) model and implement an NDT system that performs data preprocessing and prediction of petroleum production.
- To test whether the NDT system improves the prediction performance of other methods, such as C4.5 and artificial neural network.

- To apply the developed NDT system in real world application of petroleum production prediction and then demonstrate that the system yields reasonable prediction accuracy and gives explicit information in rules.

1.3 Thesis Structure

The remainder of the thesis is organized as follow: In chapter 2, we present an overview on the principles of classification and prediction. In chapter 3, we present reviews and discuss the leading approaches in petroleum domain that have appeared in past and recent literature. In chapter 4, we present the methodology for data preparation and procedures of the prediction implemented in this research work. In chapter 5, we present case studies of applications in prediction of petroleum production using the system we proposed. In chapter 6, we conclude our work with respect to the results from the models, as well as result analysis. Chapter 7 suggests possible extensions to the project and areas for future research.

Chapter 2

Background: Literature on Methodology

Decision making is a predominant category of human thought processes leading to the selection of a course of action among alternatives. However, making a good decision is not easy, where many factors are claiming your attention. Fortunately, the development of computer intelligent technology has made it possible to help decision making by discovering or articulating the hidden information from data sets. In general, classification and prediction are two forms of data analysis. Classification is the task of assigning objects to one of several predefined categories. And prediction is a statement or claim that a particular event will occur in the future in more certain terms than a forecast. Classification can be used for extracting models that describe important data classes. And prediction can predict future data trends. One of the main objectives of such analysis is to provide us with a better overall understanding of the data sets for decision making.

This chapter provides background literature on the principles of classification and prediction.

2.1 Overview of Classification

Classification, which is the task of assigning objects to one of several predefined categories, is a pervasive problem that encompasses many diverse applications [26]. Such analysis is capable of predicting categorical (discrete, unordered) labels and is growing in popularity. For instance, a classification model can be built to detect spam email messages based upon the message header and content, categorize bank loan application

as either safe or risky according to the customer's credit, and classify cells as malignant or benign based upon the results.

As defined in [26], classification can be considered as the process of learning a target function f that maps each attribute set $\mathbf{x}$ to one of the predefined class labels $\mathbf{y}$, as shown in Figure 2.1.

Figure 2.1 Classification Process

The input data for a classification process is a collection of records. Each record, also known as an instance or example, is characterized by a tuple $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x}$ is the attribute set and $\mathbf{y}$ is a special attribute. A sample data set of eight records is shown in Table 2.1. Humidity, temperature and wind are three categorical-type attributes that need to be evaluated to reach the target decision of Yes or No for the decision class Picnic. Although attributes presented in Table 2.1 are discrete data only, the attribute set can also contain continuous features. However, the class label must be a discrete attribute. The main contribution of classification is that it can be used to predict the class label of unknown data.

Table 2.1 Picnic dataset 1

Sample #	Humidity	Temperature	Wind	Decision: Picnic?
1	High	Cool	Windy	No
2	High	Cool	Windy	No
3	High	Mild	Breezy	No
4	Low	Mild	Breezy	No
5	Low	Cool	Breezy	Yes
6	Low	Cool	Breezy	Yes
7	Low	Mild	Windy	Yes
8	High	Mild	Windy	Yes

2.2 Overview of Prediction

According to [25], prediction or forecast is a statement or claim that a particular event will occur in the future. While classification predicts categorical (discrete, unordered) labels, prediction models continuous valued function. For instance, maybe the marketing manager would like to predict how much a given customer will spend during a sale at the store. This kind of data analysis is an example of numeric prediction, where the model constructed predicts a continuous valued function, or ordered values, as opposed to a categorical label. Similar to classification as described above, prediction can also be viewed as a mapping or function, $y=f(x)$, where x is the input, and the output y is a continuous or ordered value; that is, we wish to learn a mapping or function that models the relationship between x and y [27]. Furthermore, prediction also differs in the method compared with classification. As with classification, the training set used to build a model should not be used to assess its accuracy. An independent test set should be used instead. The accuracy of a prediction model is estimated by computing an error based on the

difference between the predicted value and the actual known value of y for each of the test tuples, x .

2.3 System Framework

Generally speaking, development of a system framework [27] for the classification and prediction consists of four key stages: preprocessing, modeling, evaluation and knowledge representation. We propose a generic framework and figure 2.2 illustrates the key stages of a generic framework for classification and prediction tasks, and data flow between the stages.

- Preprocessing: The original data obtained have the commonplace properties of incomplete (lacking attribute values or certain attributes of interest, or containing only aggregate data), noisy (containing errors, or outlier values that deviate from the expected), and inconsistent (containing discrepancies in the tag names used to label attributes). In other words, there are reported errors, unusual values, and inconsistencies in the data recorded for some transactions. Therefore, the initial processing attempts to “preprocess” the data by the following tasks: cleaning, integration and transformation.

(i) Cleaning: Data cleaning routines work to “clean” the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies.

(ii) Integration: Data from multiple sources may be represented by a given concept with different names causing inconsistencies and redundancies. Data integration combines multiple sources of data into a consistent dataset.

(iii) Transformation: Better results can be obtained in some methods if the data to be analyzed have been normalized or aggregated. Data transformation with these operations would contribute towards the success of the prediction.

- Modeling: This involves training of a classifier or a predictor utilizing an intelligent method to discover valuable information for prediction, such that the learned model is able to classify or predict unseen data under the same condition. As shown in Figure 2.2, the arrows in the modeling stage show the direction of data flow, with the left-hand-side showing the process of classification, and right-hand-side showing the process of prediction. Data flow will go either to one side or the other, by predicting a categorical label on the left-hand-side, or by predicting a numerical label on the right-hand-side.
- Evaluation: After modeling pattern evaluation is needed to identify the truly interesting patterns representing knowledge, based on some measurements. Classification and prediction are assessed in terms of the following measures:

- (i) Classifier accuracy measurement: the misclassification rate, the percentage of the number of objects incorrectly classified out of the total number of objects classified, and the confusion matrix, which displays the distribution of data objects in terms of their actual classes and their predicted classes, are used via cross-validation techniques for assessing the classifier's accuracy.

Figure 2.2 Generic Frameworks for Classification and Prediction Tasks

(ii) Predictor accuracy measurement: the quality of a predictor is measured in terms of mean absolute error (MAE) and root mean square error (RMSE). MAE is defined as the average of the magnitudes of the absolute true errors [57],

$$MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t| \quad (2.1)$$

while RMSE is the positive square root of the mean square error [57]:

$$RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2} \quad (2.2)$$

The holdout method where the given data are randomly partitioned into two independent set, a training set and a test set, is applied for evaluating the predictor's accuracy using the two parameters described above.

- Knowledge representation: Since knowledge has been obtained by the developed model, the goal of knowledge representation here is to represent knowledge in terms of decision rules and visualization techniques.

- (i) Decision rule: Decision rule is the most natural and simple formalism that captures patterns in the given facts and maps it to specific actions. A decision rule is a function that maps the decision-making process from the current state to an agent's decision. In an information system, a decision rule is a mapping from a set of attribute values to a decision. A decision consists of a

condition part and a decision part, and can be expressed in the logical form:

$$\Phi \Rightarrow \Psi$$

where Φ expresses a set of attribute tests that data describing the current condition must pass preceding the assignment of an action or decision values Ψ .

In the prediction framework, the objective of decision rule is to allocate petroleum production to decision classes or predicting values by evaluating the input attributes (permeability, porosity, etc.) relating to the production.

- (ii) Visualization: Although someone can quickly comprehend a single rule and judge whether it is sensible, judging the sensibility of a sequence of simple rules is complicated, and is difficult to comprehend. Visualization of the results helps the analyst to understand the predictions of the complex model and assess the reliability of the model.

2.4 Applications of Classification and Prediction

Due to the large degree of the current interest in classification and prediction, many successful applications of them have been deployed in operational use on large-scale real-world problems in science, industrial and in business.

In science, one of the application areas is astronomy. A notable success was achieved by SKICAT, a system used by astronomers to perform image analysis, classification, and

cataloging of sky objects from sky-survey images [28]. In its first application, the system was used to process the 3 terabytes (10^{12} bytes) of image data resulting from the Second Palomar Observatory Sky Survey, where it is estimated that on the order of 10^9 sky objects are detectable [29] [30]. It can outperform humans and traditional computational techniques in classifying faint sky objects. Similar to the SKICAT, the JPL Adaptive Recognition Tool (JARtool) was developed [30] to automate the search for an important feature on the planet by training the system via examples. After labeling a few images of volcanoes by the astronomers, the JARtool system would automatically construct a classifier which can then proceed to scan the rest of the image database and attempt to locate and measure the estimated 1 million small volcanoes. In the field of biology, many programs have been developed to find the genes in the DNA sequences of various organisms, such as GRAIL [31], GeneID [32], GeneParser [33], GenLang [34], FGENEH [35], Genie [36], and EcoParse [37].

In business, main applications of classification and prediction areas include marketing, finance (especially investment), fraud detection, manufacturing, telecommunications, and Internet agents.

- Marketing: The primary application in marketing is to identify different customer groups and predict their behaviors. For instance, the Leader-Post (2007) predicted that travel price would continue to rise in 2007, along with the hassles [41]. Another notable marketing application is market-basket analysis systems [39], which find patterns such as, “If a customer bought X, he/she is also likely to buy Y and Z.” Such patterns are valuable to retailers.

- Investment: The investment domain is a dynamically changing, stochastic and unpredictable environment. Take the stock market as an example; there are more than two thousand stocks available for a portfolio manager or individual investor to select. This poses a problem of filtering all those available stocks to find the ones that are likely worth investment. One of the famous investment systems is LBS Capital Management [40]. It uses expert systems, neural nets, and genetic algorithms to manage portfolios totaling \$600 million. Tseng and Lin [42] also investigated and compared different existing intelligent systems in the investment domain.
- Manufacturing: The CASSIOPEE troubleshooting system, developed as a part of a joint venture between General Electric and SNECMA, was applied by three major European airlines to diagnose and predict problems for the Boeing 737. This system received the European first prize for innovative application [43].

In other areas, there are still a lot of applications, such as the well-publicized system IBM's ADVANCED SCOUT [44], which is used by the National Basketball Association (NBA) coaching staffs to discover interesting patterns in basketball game data and the telecommunications alarm-sequence analyzer (TASA) [45], which uses a novel framework for locating frequently occurring alarm episodes from the alarm stream and for presenting them as rules.

In all, the examples above are just a few of many systems that use intelligent techniques in classification and prediction. However, it turns out that applications of classification and prediction in science may, in general, be more effective than

applications in business, industry or other areas. Primarily, this is due to the fact that the science end-users typically know the data in intimate detail which allows them to intuitively guess the important transformations, since scientists and researchers are trained to formalize intuitions into procedures / equations making migration to computers an easier matter [38].

2.5 General Approaches of Classification and Prediction

A wide variety of approaches exist for classification and prediction, here just a subset of widely used and popular techniques is outlined.

2.5.1 Decision Tree Learning

Decision tree learning is one of the most prevalent data mining methodologies used in classification problems. The supervised learning process takes a set of input training data consisting of one or more attributes or features, and assigns each of them to one of the predefined discrete labeled classes. The decision-tree classifier is then able to predict the class of unseen data based on their given attribute values. This methodology has been successfully used in many application areas ranging from medicine to game theory and business [27].

An entropy-based decision tree learning algorithm, such as C4.5, employs a greedy search through the space of possible attribute splits for one that best categorizes the given set of training data at each step of the recursive data-partitioning process [23]. The C4.5 algorithm constructs a decision tree starting from a training set TS, which is a set of cases, or tuples in the database terminology. Each case specifies values for a collection of

attributes and for a class. Each attribute may have either nominal or numeric values. The class may have only nominal values and does not admit the unknown value.

In C4.5, each node in the tree is associated with a set of cases. Also, cases are assigned weights to take into account unknown attribute values. In particular, the root is associated with the whole training set TS and all weights are 1.0. Each non-leaf node of the tree specifies a test over an attribute, which splits the cases into subsets associated with the children of the node. The splitting is repeated down the tree trying to produce leaves that contain only nodes belonging to the same class.

More details on decision tree learning will be provided in the next chapter.

2.5.2 Regression-Based Analysis

At present, the most widely used approach to deal with a prediction problem is regression, a statistical methodology that was developed by Sir France Galton. Regression analysis can be used to model the relationship between one or more independent or predictor variables and a dependent or response variable [27]. In general, the values of the input variables are known. The response variable is what we want to predict. Given a tuple described by input variables, we want to predict the associated value of the response variable. Regression-based analysis consists of a family of techniques for prediction that fit linear and nonlinear combinations of basic functions (sigmoids, splines, polynomials) to combinations of the input variables.

2.5.2.1 Linear Regression

Linear regression analysis involves a response variable, y , and several input variables. It allows response variable y to be modeled as a linear function of, say, n input variables

or attributes, $A_1, A_2, \dots, A_n$, describing a tuple, $X = (x_1, x_2, \dots, x_n)$. The training data set, D , contains data of the form $(X_1, y_1), (X_2, y_2), \dots, (X_D, y_D)$, where the X_i are the n -dimensional training tuples with associated class labels, y_i . An example of a multiple linear regression model based on two predictor attributes or variables, A_1 and A_2 , is

$$y = w_0 + w_1x_1 + w_2x_2, \quad (2.3)$$

where x_1 and x_2 are the values of attributes A_1 and A_2 , respectively, in X , w_0, w_1 , and w_2 are regression coefficients that can be solved for by the method of least squares. The method of least squares estimates the best-fitting straight line as the one that minimizes the error between the actual data and the estimate of the line. The regression coefficients, which are tedious to solve by hand, are instead commonly solved with the use of some statistical software packages, such as SAS, SPSS and Sigmat-Plot.

To illustrate the operation, Table 2.2 shows an example dataset of eight records. Humidity, Temperature and Wind are three numeric-type attributes or features that need to be evaluated to reach the target decision of Yes or No for the decision class Picnic.

Table 2.2 Picnic dataset 2

Sample #	Temp. on Day D	Forecast for Day D+1	Decision: Picnic?
1	11.2	13.9	No
2	18.7	14.3	No
3	13.2	19.9	Yes
4	16.1	15.6	No
5	13.6	17.3	Yes
6	12.7	14.2	No
7	17.5	16.2	No
8	15.9	17.6	No
9	11.3	12.3	No

Then, we can get the result of simple linear regression as shown in Figure 2.3. However, the fit may be poor if the two variables are nonlinear.

Figure 2.3 A simple linear regression for picnic dataset with two classes

2.5.2.2 Nonlinear Regression Methods

The straight-line linear regression described above has limited use, since many relationships in the real world do not follow a straight line. The purpose of the nonlinear methods is to get a more accurate model in terms of some nonlinear models including artificial neural network, and classification and regression trees (CART). Consider neural network, for example. Figure 2.4 illustrates the decision boundary that a neural network might find for the example above. The networks of the appropriate size can universally approximate any smooth function to any desired degree of accuracy. In addition, the standard squared error and cross-entropy loss function are used to train neural networks

for prediction and classification, respectively. Back propagation is a parameter-search method that performs gradient descent in parameter space to find a local maximum starting from random initial conditions [29]. Although nonlinear regression methods might prove to be more accuracy, they are difficult to interpret. Some degree of explicit explanation of the result, as a simple form of IF THEN rule, is needed to help people to comprehend.

More details on neural network and CART will be provided in the next chapter.

Figure 2.4 An example of decision boundaries learned by neural network for picnic data set

2.5.3 Genetic Algorithms

Genetic Algorithm (GA) is a widely used evolutionary computation algorithm. It is well suited for solving the Traveling salesperson problem (TSP), in which an optimal solution gives the shortest total distance covered in an ordered traversal through a list of

cities. The optimal solution is considered “hidden” inside the genetic pool of a given population, and can be excavated through associations between chromosomes encoding potential solutions [29]. In general, this method is less commonly used in solving classification and prediction problems than other methods described above. However, it does show its strength in certain applications, and hence it is worthwhile to include it here.

The evolutionary mechanism for classification and prediction is represented and defined as follows:

- **Solution encoding:** Chromosome encoding solutions are operated by permutations of attributes. An initial population of size n is created consisting of randomly generated rules. Each rule can be represented by a string of bits [61]. To illustrate this clearly, we consider the example shown in Table 1 again. Humidity, Temperature and Wind are three categorical-type attributes or features that need to be evaluated to reach the target decision of Yes or No for the decision class Picnic. Then, the sample rule:

“IF Humidity is high, Temperature is cool, and Wind is windy, THEN Picnic= No”

can be encoded as the bit string “1110”, where the three leftmost bits represent attributes Humidity, Temperature and Wind, and the rightmost bit represents the class. Similarly, the sample rule:

“IF Humidity is Low, Temperature is Mild, and Wind is windy, THEN Picnic=Yes ”

can be encoded as “0011”. If an attribute has k values, not just two, then k bits may be used to encode the attribute’s values. Classes can be encoded in a similar fashion.

- **Fitness function:** Each sample rule is evaluated by a fitness function measuring its classification accuracy on a set of training samples, such that a new population is formed to consist of the fittest rules in the current population [61].
- **Genetic operators:** Genetic operators govern the evolutionary process by passing on the fitness of the parent population to the new generation to gain an increased average fitness. In crossover, substrings from pairs of rules are swapped to form new pairs of rules. In mutation, randomly selected bits in a rule’s string are inverted [61].
- **Termination:** The process of generating new populations based on prior populations of rules continues until a population, P , evolves, where each rule in P satisfies a prespecified fitness threshold [61].

2.6 Evaluating the Accuracy of a Classifier or Predictor

There are many different ways to obtain a reliable estimate of classifier accuracy or predictor accuracy. Cross-validation and hold out are common techniques for assessing accuracy based on randomly sampled partitions of the given data. The use of such techniques is useful to determine model election on a particular problem.

2.6.1 Cross-Validation

Cross-validation is the statistical practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset, while the other subset(s) are retained for subsequent use in confirming and validating the initial analysis [46]. The initial subset of data is called the training set; the other subset(s) are called validation or testing sets. In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or “fold”, $D_1, D_2, \dots, D_k$, each of approximately equal size. Training and testing is performed k times. In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data. For prediction, the error estimate can be computed as the total loss from the k iterations, divided by the total number of initial tuples.

2.6.2 Holdout Method

Holdout validation is not cross-validation in common sense, because the data never are crossed over. In this method, the given data are randomly partitioned into two independent sets, a training set and a test set [27]. Typically, two-thirds of the data are allocated to the training set, and the remaining one-third is allocated to the test set. The training set is used to derive the model, whose accuracy is estimated with the test set. The estimate is pessimistic because only a portion of the initial data is used to derive the model.

Chapter 3

Background Literature on Application Area

In recent years, there has been growing research interest in developing techniques for petroleum prediction. This chapter reviews and discusses the leading approaches in petroleum domain that have appeared in past and recent literature.

3.1 Current Methods in Petroleum Prediction

In current research, methods applied to petroleum prediction can be classified into two general categories: conventional methods and intelligent methods. A conventional method is characterized by the statistical nature, in which the flow of control is based on some predefined mathematical functions through production data. On a different note, a system built with intelligent architecture does not only rely on the statistical information of the production data, but also integrate with artificial intelligent techniques that have been widely applied in various industrial, business and scientific domains. The advantages of using an intelligent approach are increased accuracy and reliability, and reduced subjectivity of the production analysis, by developing an automatic or semi-automatic system, based on artificial intelligent algorithms. Conventional approaches, on the contrary, may operate with their simplicity in terms of statistical methods and few reservoir or well parameters, and therefore are still being used nowadays.

3.1.1 Conventional Methods: Curve Estimation

A given set of data can usually fit for some types of curves, and the objective of curve estimation is to find a curve with minimal deviation from all the data points. Curve

estimation is the most common and straightforward approach in petroleum reserve prediction. Some of the widely used techniques are outlined below:

3.1.1.1 Arps Decline Analysis

Arps decline analysis was the first systematic approach to analyzing oil and gas production data in the 1950's [6]. The goal of this method is to use an empirical curve match to predict the future performance of the well. The empirical Arps decline equation represents the relationship between production rate and time for oil wells during pseudosteady-state period and is shown as follow:

$$q(t) = \frac{q_i}{(1 + bD_t)^{1/b}} \quad (3.1)$$

where $q(t)$ is the oil production rate at time t and q_i is the initial oil production rate. In this relationship, $b=0$ and $b=1$ represent exponential and harmonic decline, respectively [3]. Any value of b between 0 and 1 represents a hyperbolic decline.

One of the most attractive features of the Arps methodology is its simplicity, which requires no knowledge of reservoir or well parameters [2]. Thus, it can be adopted for production through any types of reservoir drive mechanism [1]. However this method has two restrictions related to the efforts that focus on production data analysis. The first has to do with the fact that Arps decline analysis is unable to disassociate the production forecast from operating constraints, which means the ultimate recoverable reserves, predicted by the Arps decline must inherently assume that historical operating conditions remain constant in the future. A second restriction of Arps is its limited applicability in the transient state. In other words, the Arps decline analysis, capable only of predicting ultimate recovery, has very limited application.

3.1.1.2 Fetkovich Typecurve Analysis

An improved method proposed by Fetkovich applies the concept of using typecurves to the analysis of production data [4]. The Fetkovich methodology combines some analytical solutions for transient flow with Arps empirical equations on a log-log graph and develops a typecurves matching technique for diagnosing the production data [7]. The most valuable feature of Fetkovich typecurve analysis is that it can show whether or not production data is still in transient or has become boundary dominated, by incorporating the typecurve match [5]. However the Fetkovich method, like Arps equation, calculates expected ultimate recovery based on the assumption of single-phase oil flow in closed boundary systems [3], and the transient portion of the Fetkovich typecurves assumes constant bottomhole flowing pressure.

3.1.1.3 Advanced Methods Used in Water Flooding Oilfields

Since many oilfields are developed by water flooding, two-phase fluid flow instead of single-phase flow occurs in these cases. Many attempts have been made to deal with this problem. Lefkovit et al. proposed an exponential decline form for gravity drainage reservoirs with a free surface by neglecting capillary pressure [8]. Fetkovich et al. include gas-oil relative permeability effect on oil production for solution gas drive through pressure ratio term which assumes that the oil relative permeability is a function of pressure [9]. Aronofsky et al. suggested an empirical model to match oil production by water injection in naturally-fractured reservoirs developed by water flooding [10]. Baker et al. characterized naturally-fractured reservoirs using an analytical decline model that combined decline rate data and imbibition data [11]. Li and Horne proposed a decline analysis model derived from fluid flow mechanisms and analyzed oil production data

from naturally fractured reservoirs developed by water flooding [12]. However, in the application of these methods, it is difficult to predict which equation can optimally describe production of a reservoir. This can result in unreliable predictions [13].

3.1.2 Intelligent Methods

Much research work has been done in application of artificial intelligence (AI) in petroleum engineering due to its advantages of computational efficiency, non-linear characteristics and minimum subjectivity. Several existing popular approaches are described below:

3.1.2.1 Artificial Neural Network in Petroleum Prediction

Artificial neural network (ANN) was first introduced by psychologist Frank Rosenblatt [14] in 1958, and was utilized to predict production or virtually measure formation characteristics such as porosity, permeability and fluid saturation from conventional well log and core analysis [17]. Since the inter-relation of the parameters is complicated, and the process can be time consuming and expensive in terms of labor and computational resources in conventional methods, demand for effective models to predict production increases rapidly. It has been proven that a carefully orchestrated neural network analysis is capable of providing more accurate and repeatable results when compared to the conventional methods [15] [16]. Recently, many attempts have been made to apply ANN in petroleum engineering. Nguyen et al. [20] used multiple neural network model to make both short and long term time series predictions of petroleum production. Mohaghegh et al. applied ANN to predict the permeability of the formations using the data provided by geophysical well logs with good accuracy [21]. Wong et al.

also used ANN to predict permeability with an application to the Ravva oil and gas field, offshore India [22]. Nikraves et al. proposed to use ANN for the prediction of formation damage during fluid injection into fractured, low permeability reservoirs [23].

In summary, advantages with neural networks are [18]:

- They are non-linear: The true power of neural networks lies in their ability to represent both linear and non-linear relationships and to learn these relationships directly from the data being modeled.
- They can classify patterns accurately: The best quality of neural network is the ability to train without much formal statistical knowledge [19].
- They are error-tolerant: Their high tolerance to noisy data makes them well suited to problems including pattern recognition and prediction.

More details on ANN technique will be provided in the next chapter.

3.1.2.2 Decision Tree Learning for Petroleum Engineering

A decision tree is an idea generation tool that generally refers to a graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. In data mining, a decision tree is a predictive model. It's a mapping of observations about an item to conclusions about the item's target value.

One of the great advantages of decision trees is that petroleum engineer find them easy to be interpreted and used. For example, if you are looking for the best potential oil production among the wells, a quick glance might help you to identify the terminal nodes in the tree that have highest percentage of production, and then you can focus your effort on the specific wells described by those nodes. Furthermore, a decision tree can be constructed relatively quickly, compared to other methods.

Some research work has been done in the application of decision trees for petroleum engineering. Examples include: Hector and Akhil [47] used decision tree to classify the data for permeability predictions based on the well-logs. Trond [48] applied decision tree analysis to estimate the range of uncertainty in the reservoir production prognosis. Agbon et al. [49] compared the decision tree model with the fuzzy model for the ranking of reservoir based on the amount of natural gas production in Venezuela. However, much research on decision-tree learning in the past used implementations that support univariate attribute testing at each node. This does not limit the accuracy and expressiveness of the decision tree, if the training data was assumed to exhibit attribute independence. But in petroleum engineering, the available data sets contain attributes that are interdependent, and this makes it necessary to take this matter into consideration when designing decision-tree learning models. Since there is often no prior knowledge of whether the data samples at hand exhibit attribute dependencies, or what type of dependencies exist between which attributes, a mechanism for exploring these factors must be incorporated into the learning process.

More details on decision tree learning will be provided in the next chapter.

3.1.2.3 Prediction Using Genetic Algorithms

Jian et al. [50] proposed a method of using the Genetic Algorithm to predict current permeability field of reservoirs. The predictor integrates the theory of survival of the fittest into geologic understanding of the reservoir to look for the permeability distribution that is closest to the actual reservoir conditions using the three operators of selection, crossover and mutation repeatedly. Thus, a relationship between permeability and field production data can be established based on the Genetic Algorithm. Emera et al.

[51] also presented a new genetic algorithm (GA)-based model to predict Green-House Gas (GHG)-oil interaction parameters utilizing parameters of saturation pressure, temperature, oil specific gravity, oil and GHG compositions. Besides the higher prediction results having outstanding accuracy compared to other models, this GA-based model is also capable of covering a wide range of oil properties for dead and live oil wells and also for different flue gas compositions. Since genetic algorithms are easily parallelizable, this makes them an attractive choice for classification, as well as other optimization problems in petroleum engineering. However, a drawback of Genetic Algorithms is that a solution is "better" only in comparison to other, presently known solutions; such an algorithm actually has no concept of an "optimal solution," or any way to test whether a solution is optimal. This also means that these algorithms never know for certain when to stop, aside from the length of time, or the number of iterations or candidate solutions, that you wish to allow it to explore.

Chapter 4

Methodology

This chapter presents the neural based decision-tree learning (NDT) model used for petroleum prediction in this project. In section 4.1, we provide some general ideas about decision tree learning algorithms. Section 4.2 extends the discussion to introduce the difficulty of decision tree learning in dealing with attribute dependencies in the data sets, which is clarified by two supporting examples illustrating the two types of dependencies considered. Section 4.3 presents the underlying theoretical concepts encompassed by the proposed NDT Model. Section 4.4 describes the NDT framework including its structure and some modifications for petroleum prediction. Finally, section 4.5 discusses the tools to develop the NDT model.

4.1 Overview of Decision Tree Learning

A typical entropy-based decision tree learning algorithm, such as ID3, C4.5 and CART, employs a greedy search through the space of possible attribute splits for one that best categorizes the given set of training data at each step of the recursive data-partitioning process. The C4.5 algorithm constructs the decision tree with a divide and conquer strategy. At each node, the algorithm is executed (as shown in Figure 4.1), trying to exploit the locally best choice, with no backtracking allowed [23].

```

FormTree( $T$ )
(1) ComputeFrequency( $T$ );
(2) if OneClass or FewCases
    return a leaf;
    create a decision node  $N$ ;
(3) ForEach Attribute  $A$ 
    ComputeGain( $A$ );
(4)  $N.test = \text{AttributeWithBestGain}$ ;
(5) if  $N.test$  is continuous
    find Threshold;
(6) ForEach  $T'$  in the splitting of  $T$ 
(7)   if  $T'$  is Empty
        Child of  $N$  is a leaf
    else
(8)   Child of  $N = \text{FormTree}(T')$ ;
(9) ComputeErrors of  $N$ ;
    return  $N$ 

```

Figure 4.1 Pseudo-code of the C4.5 Tree-Construction Algorithm [62]

Let T be the set of cases associated at the node. The weighted frequency $\text{freq}(C_i, T)$ is computed(step(1)) of cases in T whose class is C_i , for $i \in [1, NClass]$.

If all cases(step(2)) in T belongs to a same class C_j (resp., the number of cases in T is less than a certain value) then the node is a leaf, with associated class C_j (resp., the most frequent class). The classification error of the leaf is the number of cases in T whose class is not C_j .

If T contains cases belongs to two or more classes (step (3)), then the information gain of each attribute is calculated, as will be described later. The attribute with the highest information gain(step(4)) is selected for test at the node. Tests of nominal attributes consider partitions of T into n subsets, one for each possible value of the attribute. Tests of numeric attributes partition T into two subsets, namely the set of cases with attribute value not greater than and the set of cases with attribute value greater than a certain threshold which has to be determined(step(5)). A decision node has s children if

$T_1, \dots, T_s$ are the sets of the splitting produced by the test on the selected attribute(step(6)). Obviously, $s=2$ when the selected attribute is numeric, and $s=h$ for nominal attributes with h possible values.

If T_i is empty, (step(7)) the child node is directly set to be a leaf, by classifying the most frequent class at the node.

The divide and conquer approach consists of recursively applying the same operations on each non-empty T_i (step(8)). Finally, the classification error(step(9)) of the node is calculated as the sum of the errors of the child nodes. If the result is greater than the error of classifying all cases in T as belonging to the most frequent class in T , then the node is set to be a leaf, and all sub-trees are removed.

4.1.1 Measurement of Attribute Selection

An attribute selection measure is a heuristic for selecting the splitting criterion that “best” separates a given data set T , of class-labeled training cases into individual classes. If we were to split T into smaller sets according to the outcomes of the splitting criterion, ideally each set would be pure, which means all of the cases falling into a given set would belong to the same class. Attribute selection measures are also known as splitting rules because they determine how the cases at a given node are to be split. The attributes selection measure provides a ranking for each attribute describing the given training cases. The attribute having the best score for the measure is chosen as the splitting attribute for the given cases. There are many measures that can be used to determine the best way to split the records. These measures are defined in terms of the class distribution of the cases before and after splitting. Three popular attribute selection measures: information gain, gain ratio and gini index are described here.

4.1.1.1 Information Gain

ID3 uses information gain as its attribute selection measure. This measure is based on pioneering work by Claude Shannon on information theory, which studied the value or “information content” of messages. Let node N represent or hold the cases of set T. The attribute with the highest information gain is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the cases in the resulting sets and reflects the least randomness or “impurity” in these sets. Such an approach minimizes the expected number of tests needed to classify a given case and guarantees that a simple tree is found. The expected information needed to classify a case is given by

$$\text{Info}(T) = -\sum (\text{freq}(C_i, T)/|T|) * \log_2(\text{freq}(C_i, T)/|T|) \quad i=1, \dots, k \quad (4.1)$$

where $\text{freq}(C_i, S)$ means the frequency of elements in T that belong to class C_i . A log function to the base 2 is used, because the information is encoded in bits. $\text{Info}(T)$ is just the average amount of information needed to identify the class label of a case in T. Note that, at this point, the information we have is based solely on the proportions of cases of each class. $\text{Info}(T)$ is also known as the entropy of T.

Suppose we were to partition the cases in T on some attribute A having v distinct values, $\{a_1, a_2, \dots, a_v\}$, as observed from the training data. If A is discrete-valued, these values correspond directly to the v outcomes of a test on A. Attribute A can be used to split T into v partitions or subsets, $\{T_1, T_2, \dots, T_v\}$, where T_j contains those cases in T that have outcome a_j of A. These partitions would correspond to the branches grown from node N. Ideally, we would like this partitioning to produce an exact classification of the

cases. That is, we would like for each partition to be pure. However, it is quite likely that the partitions will be impure, which means a partition may contain a collection of cases from different classes rather than from a single class. Thus, how much more information would we still need after the partitioning in order to reach an exact classification is measured by

$$\text{Info}_A(T) = \sum (|T_i|/|T|) * \text{Info}(T_i) \quad i=1, \dots, n \quad (4.2)$$

where $|T_i|$ is a subset of T for which attribute A has the same value. The term $|T_i|/|T|$ acts as the weight of the i th partition. Info_A is the expected information required to classify a case from T based on the partitioning by A . The smaller the expected information that is still required, the greater the purity of the partitions.

Based on this measure of information, information gain is defined as the difference between the original information requirement (based on just the proportion of classes) and the new requirement (obtained after partitioning on A). The attribute that gives the highest Gain value is selected, where Gain is obtained from Equation below .

$$\text{Gain}(A) = \text{Info}(T) - \sum (|T_i|/|T|) * \text{Info}(T_i) \quad i=1, \dots, n \quad (4.3)$$

The $\text{Gain}(A)$ tells us how much would be gained by branching on A . It is the expected reduction in the information requirement caused by knowing the value of A . Choosing the attribute A with the highest information gain is equivalent to saying that partitioning on the attribute A that would do the “best classification”, so that the amount of information still required to finish classifying the cases is minimal. The entropy-based process recursively partitions the dataset, where only yet unconsidered attributes are

candidates of the selection process for categorical-type attribute. For attributes with numeric values, there is a need for an exhaustive search for an optimal threshold value Z out of $n-1$ possible choices, which maximizes the Gain measure of Equation shown above. The same attribute may be reused for successive splits again using different Z values.

4.1.1.2 Gain Ratio

The information gain measure described above is biased toward tests with many outcomes. That is, it does not take into account the number of regions. If information gain were to be used “raw” without regard to the number of regions, then this would lead to a bias to having a huge number of regions. Imagine a region for each point in the space, such that each region only has one instantiated feature and therefore one class. Hence, C4.5, a successor of ID3, uses an extension to information gain known as gain ratio, which attempts to overcome this bias. It applies a kind of normalization to information gain using a “split information” value defined analogously with $\text{Info}(D)$ as

$$\text{SplitInfo}_A(T) = -\sum \frac{|T_i|}{|T|} \times \log_2 \left(\frac{|T_i|}{|T|} \right) \quad i=1, \dots, k \quad (4.4)$$

This value represents the potential information generated by splitting the training data set, D , into v partitions, corresponding to the v outcomes of a test on attribute A . Note that, for each outcome, it considers the number of cases having that outcome with respect to the total number of cases in D . It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning. The gain ratio is defined as [27]:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)} \quad (4.5)$$

The attribute with the maximum gain ratio is selected as the splitting attribute. The gain ratio compensates for the number of attributes by normalizing by the information encoded in the split itself.

4.1.1.3 Gini Index

The Gini index is used in another kind of decision tree, called Classification and Regression Tree (CART). In CART, the Gini index measures the impurity of T , a data set of training cases as

$$Gini(T) = 1 - \sum_{i=1, \dots, k} \left(\frac{T_i}{T} \right)^2 \quad (4.6)$$

where $\frac{T_i}{T}$ is the probability that a case in T belongs to class C_i . The Gini index considers a binary split for each attribute. First, consider the case where A is a discrete-valued attribute having v distinct values, $\{a_1, a_2, \dots, a_v\}$, occurring in D . To determine the best binary split on A , all of the possible subsets that can be formed using known values of A are examined. Each subset, T_A , can be considered as a binary test for attribute A of the form “ $A \in T_A$ ”. Given a case, this test is satisfied if the value of A for the case is among the values listed in T_A . If A has v possible values, then there are 2^v possible subsets. For example, if Humidity has three possible values, namely $\{\text{Low}, \text{Medium}, \text{High}\}$, then the possible subsets are $\{\text{Low}, \text{Medium}, \text{High}\}$, $\{\text{Low}, \text{Medium}\}$, $\{\text{Low}, \text{High}\}$, $\{\text{Medium}, \text{High}\}$, $\{\text{Low}\}$, $\{\text{Medium}\}$, $\{\text{High}\}$, and $\{\}$. We exclude the power set: $\{\text{Low}, \text{Medium}, \text{High}\}$, and the empty set from consideration since, conceptually, they do not

represent a split. Therefore, there are $2^n - 2$ possible ways to form two partitions of the data set, T , based on a binary split on A .

When considering a binary split, a weighted sum of the impurity of each resulting partition is computed. For instance, if a binary split on A partitions T into T_1 and T_2 , the gini index of T given that portioning is

$$Gini_A(T) = \frac{|T_1|}{|T|} Gini(T_1) + \frac{|T_2|}{|T|} Gini(T_2) \quad (4.7)$$

For each attribute, each of the possible binary splits is considered. For a discrete-valued attribute, the subset that gives the minimum gini index for that attribute is selected as its splitting subset. For continuous-valued attributes, each possible split-point must be considered. The strategy is similar to that described above for information gain, where the midpoint between each pair of adjacent values is taken as a possible split-point. The point giving the minimum Gini index for a given attribute is taken as the split-point of that attribute. Recall that for a possible split-point of A , D_1 is the set of cases in D satisfying $A \leq \text{split_point}$, and D_2 is the set of cases in D satisfying $A > \text{split_point}$.

The reduction in impurity that would be incurred by a binary split on a discrete- or continuous- valued attribute A is [26]:

$$\Delta Gini(A) = Gini(T) - Gini_A(T) \quad (4.8)$$

The attribute that maximizes the reduction in impurity is selected as the splitting attribute. This attribute and either its splitting subset or split-point together form the splitting criterion.

The reason the entropy-based method was used for attribute selection is mainly because decision tree learning chooses a test condition that maximizes the information gain, which is equivalent to minimizing the weighted average impurity measures of the child nodes [27]. In this way, the attribute selected in such procedure is optimal. Furthermore, decision tree learning is the most popular approach among the data mining algorithms; therefore, we choose the entropy-based method for attribute selection measurement.

4.1.2 Tree Pruning

When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers. Tree pruning methods address this problem of overfitting. Such methods typically use statistical measures to remove the least reliable branches. Pruned trees tend to be smaller and less complex and, thus, easier to comprehend. They are usually faster and better at correctly classifying independent test data than unpruned trees. In general, there are two common approaches to tree pruning: prepruning and postpruning.

4.1.2.1 Prepruning

In this approach, the tree-growing algorithm is halted before generating a fully grown tree that perfectly fits the entire training data. In order to do this, a more restrictive stopping condition must be used to stop expanding a leaf node when the observed gain in impurity measure falls below a certain threshold. The advantage of this approach is that it avoids generating overly complex subtrees that overfit the training data. Nevertheless, it is difficult to choose the right threshold for early termination. Too high of a threshold will

result in underfitted models, while a threshold that is set too low may not be sufficient to overcome the model overfitting problem. Furthermore, even if no significant gain is obtained using one of exiting attribute test conditions, subsequent splitting may result in better subtrees [63].

4.1.2.2 Post Pruning

The second and more common approach is post pruning. In this approach, the decision tree is initially grown to its maximum size. This is followed by a tree-pruning step, which proceeds to trim the fully grown tree in a bottom-up fashion. A subtree which does not contribute to improving the accuracy of classification on unseen data is replaced by a new leaf node whose class label is determined from the majority class of records affiliated with the subtree or the most frequently used branch of the subtree. The replacement integrates all the data found at the leaves of the subtree, and is labeled with the class through a majority vote. This necessarily leads to misclassification of some instances at the new leaf. If let E denote the number of misclassified examples out of a total of N examples at the leaf, the decision of whether pruning takes place at a leaf is determined by the equation below [63]:

$$e = [f + z^2/2N + z \cdot \sqrt{(f/N - f^2/N + z^2/4N^2)}] / (1 + z^2/N) \quad (4.9)$$

where $f = E/N$ is the observed error rate for a test sample. Hence $f = 0$ if the same training set is used as a test set. Eq. [63] gives the pessimistic estimate for the error rate e using the upper confidence limit of confidence $c = 25\%$ (as used by Quinlan for C4.5), with a corresponding statistic z value of 0.69, which is used as a constant in the pruning

algorithm. CART uses a method called cost complexity pruning, which is similar to the one used in C4.5. The post pruning process in CART is different. It considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree, where the error rate is the percentage of cases misclassified by the tree. It starts from the bottom of the tree. For each internal node, N , it computes the cost complexity of the subtree at N , and the cost complexity of the subtree at N if it were to be pruned. The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept. A pruning set of class-labeled cases is used to estimate cost complexity. This set is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation. The algorithm generates a set of progressively pruned trees. In general, the smallest decision tree that minimizes the cost complexity is preferred.

4.1.2.3 From Tree to Rules

Decision trees can become large and difficult to interpret. In comparison with a decision tree, the IF-THEN rules from a decision tree are easier for humans to understand, particularly if the decision tree is very large.

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent (“IF” part). The leaf node holds the class prediction, forming the rule consequent (“THEN” part). Figure 4.2 is the decision tree constructed from Table 1. It can be converted to classification IF-THEN rules by tracing the path from the root node to each leaf node in the tree. The rules extracted from Figure 1 are:

Rule #1: IF Humidity = Low AND Temp = Cool THEN Picnic = Yes

Rule #2: IF Humidity = Low AND Temp = Mild AND Wind = Windy
THEN Picnic = Yes

Rule #3: IF Humidity = High AND Temp = Mild AND Wind = Breezy
THEN Picnic = No

Rule #4: IF Humidity = High AND Temp = Cool THEN Picnic = No

Rule #5: IF Humidity = High AND Temp = Mild AND Wind = Breezy
THEN Picnic = No

Rule #6: IF Humidity = High AND Temp = Mild AND Wind = Windy
THEN Picnic = Yes

A disjunction (logical OR) is implied between each of the extracted rules. Because the rules are extracted directly from the tree, they have two main features:

- Mutually exclusive: That means we won't have rule conflicts here because no two rules will be triggered for the same case.
- Exhaustive: There is one rule for each possible attribute-value combination, so that this set of rules does not require a default rule. Thus, the order of the rules does not matter.

Figure 4.2 Decision Tree from Table 1

4.1.3 Limitations of Decision Tree Learning in Prediction of Oil

Production

The decision-tree learning algorithms (such as C4.5) derive decision trees and rules based on the training dataset. These derived trees and rules can be used to predict the classes of new instances. Since the decision-tree learning algorithms usually work with only one attribute at a time, without considering the dependencies among attributes, the results generated by those algorithms are not optimal. However, considering multiple attributes simultaneously can introduce a combinatorial explosion problem. Furthermore, it is commonly observed that the core analysis variables, which are significant factors in prediction of oil production, have some interdependencies or shared correlation among each other. Thus, the process of oil production is influenced by many interdependent factors, and an approach which can handle the complexity is needed.

Attribute dependency is defined as the ability for some attributes to control the behavior of other attributes [63]. In petroleum prediction, the relationship between the input variables of the permeability and porosity is as follow: If the permeability of a well is high, then likely the porosity would go high. Generally speaking, these dependencies can be classified into two types: categorical-type and numerical-type dependencies. Kim and Lee [53] proposed a method for numerical-type dependencies using multilayer perceptron. However, the categorical-type dependencies are not considered in their work. Chen [54] proposed the concepts for categorical-type dependencies. However, they did not describe how these dependencies could be obtained efficiently.

The following two examples provided illustrate the limitations of using the decision tree learning algorithm alone in dealing with both attribute types described above.

4.1.3.1 Categorical-type Dependencies

To illustrate the operation of a decision-tree learning algorithm on categorical-type dataset, let's use table 2.1 again as an example. The C4.5 decision learning algorithm is limited to considering a single attribute at a time at each given node, and the decision tree based on such univariate splits has relatively simple representational structure from which logical rules can be easily induced, as shown in Figure 4.3. The tree structure shows that each attribute is considered once at most to reach a target decision for a categorical-type-only data set. This property of C4.5 guarantees that, for a univariate model, the tree height complexity has an upperbound on the number of attributes in the data set.

Figure 4.3 Univariate Tree

Figure 4.4 Multivariate Tree

In spite of the simple representational form of the univariate decision tree, a disadvantage would be its inability to capture dependencies among attributes. As a result, it is restricted in its function as a prediction model. Figure 4.4 shows the multivariate decision tree, where the two attributes Temperature and Wind are tested in parallel at the root node. This illustrates the limitations of the univariate model in expressing the relationships between attributes, and the feasibility of less complex tree structures in multivariate models.

4.1.3.2 Numeric-type Dependencies

To illustrate the limitations of the functionality of decision tree learning on numeric-type data samples, let's consider the Table 2.2 as an example again. The set of rules used to generate the data in Table 2.2 using two numerical attributes are as follows,

IF Forecast Temp. for Day D+1 > (Temp. on Day D) + 3
 THEN Picnic = Yes
 ELSEIF Forecast Temp. for Day D+1 ≤ (Temp. on Day D) + 3
 THEN Picnic = NO

The resulting decision tree constructed from the recursive partitioning of the dataset in Table 2.2 by C4.5 is shown in Figure 4.5. The decision rules are shown below:

```

IF Forecast for D+1  $\leq$  15.6
    THEN Picnic = NO
ELSEIF Forecast for D+1 > 15.6
    IF Temp. on D  $\leq$  13.6
        THEN Picnic = YES
    ELSEIF Temp. on D > 13.6
        THEN Picnic = NO
  
```

Figure 4.5 Decision Tree of Table 2.2

Based on the induction rules generated from the decision tree, Figure 4.6 illustrates the decision regions separated by decision boundaries lines. The equation of the decision boundary (shown in bold) satisfies the equation: Forecast for D+1 = Temp. on day D + 3. The two Error regions are bounded by the resulting decision rules generated by C4.5. This figure demonstrates the relationship between the decision rules and decision regions, where the data found in the Error regions will result in making incorrect predictions. The flaws of the C4.5 algorithm on numerical dataset are revealed in the fact that, depending on the distribution of the given test data samples, there is a high risk of committing classification errors when using the output rules of the algorithm.

Figure 4.6 Error Graph of Decision Region and Decision Rules

4.2 Overview of Artificial Neural Network

In the real world, a ‘neural network’ is a collection of neurons, the tiny cells our brains are comprised of. A typical neuron contains a cell body where the nucleus, dendrites and an axon are located [17]. Figure 4.7 shows a schematic diagram of two bipolar neurons. A Signal or information in the form of a train of electro-chemical pulses, enters the cell body from the dendrite and activates the neuron by means of a state of excitation or inhibition, providing an output that travels through the axon and links to other neurons via synaptic connections, where it becomes the input to the receiving neuron. This architecture makes it possible to transmit the signal or information between the neurons in the network, and is thus the main driving force behind the complex behavior that comes so natural to us.

Figure 4.7 Schematic diagram of two bipolar neurons [17]

In the configuration of ANN, an information system has certain features in common with biological neural networks. They have been developed from generalization of mathematical models of human cognition or neural biology, based on the following assumptions [17].

- Information processing occurs in many simple elements that are called neurons (processing elements).
- Signals are passed between neurons over connection links.
- Each connection link has an associated weight, which, in a typical neural network, multiplies the signal being transmitted.
- Each neuron applies an activation function (usually non-linear) to its net input to determine its output signal

4.2.1 Artificial Neuron

Figure 4.8. shows the architecture of a typical neuron (processing elements) in an artificial network. This artificial neuron simulates the process of the ‘real’ biological neuron initiated by the multiplied output training data (instead of signal or information in the real neuron) from other neurons as inputs that are summed and pass to the activation function, the output of which is then passed to next neuron. The arrow shows the direction of data flow from the input to output.

Figure 4.8 An artificial neuron architecture [17]

Each link is associated with a weight. To compute the net input to the unit, each input connected to the unit is multiplied by its corresponding weight, and this is summed as shown by Equation below.

$$I_j = \sum_i w_{ij} O_i + \theta_j \quad (4.10)$$

where w_{ij} is the weight of the connection from unit i in the previous layer to unit j ; O_i is the output of unit i from the previous layer; and θ_j is the bias of the unit. If a weight is

negative, then the connection is inhibitory; if it is positive, it has an excitatory connection. The bias acts as a threshold in that it serves to vary the activity of the unit.

4.2.2 Operation of Neural Network Process

An artificial neural network is composed of a large number of connected neurons working together in a specific formation. In a multi-layer network there is usually an input layer along with one or more hidden layers which correspond to the number of parameters that are being presented to the network as input. The same is true for the output layer. Neural network analysis is not limited to a single output by the fact that neural nets can be trained to build neuron-models with multiple outputs. The neurons in the hidden layer or layers are mainly responsible for feature extraction. They are suitable for tasks such as classification, prediction and pattern recognition.

There are many kinds of neural networks. Neural network scientists and practitioners have provided different classifications for neural networks. One of the most popular classifications is based on the training methods. Neural nets can be divided into two major categories based on the training methods, namely supervised and unsupervised neural networks. Unsupervised neural networks are mainly clustering and classification algorithms. They have been used in the oil and gas industry to interpret well logs and to identify lithology. They are called unsupervised simply because no feedback is provided to the network. The network is asked to classify the input vectors into groups and clusters. This requires a certain degree of redundancy in the input data and hence the notion that redundancy is knowledge.

Most of the neural network applications in the oil and gas industry are based on supervised training algorithms. During a supervised training process, both input and

output are presented to the network to permit learning on a feedback basis. Specific architecture, topology and training algorithm are selected, and the network is trained until it converges. During the training process, the neural network tries to converge to an internal representation of the system behavior.

4.2.2.1 Structure of Neural Network Model

Figure 4.9 gives the architecture of a typical three-layered neural network model. The layer that receives the inputs is called the input layer. It typically performs no function other than the buffering of the input signal. The network outputs are generated from the output layer. Any other layers are called hidden layers because they are internal to the network and have no direct contact with the external environment. Sometimes they are likened to a “black box” within the network system. There may be zero to several hidden layers. In a fully connected network, every output from one layer is passed along to every node in the next layer.

Figure 4.9 A typical three-layered neural network model

Before training can begin, the network topology must be decided by the user to specify the number of units in the input layer, the number of hidden layers, the number of units in each hidden layer, and the number of units in the output layer. There are no clear rules as to the “best” number of hidden layer units. Network design is a trial-and-error process and may affect the resulting trained network. The initial values of the weights may also affect the resulting accuracy. Once a network has been trained and its accuracy is not acceptable, it is common to repeat the training process with a different network topology, or a different set of initial weights. Normalizing the input values for each attribute measured in the training cases will help speed up the learning phase. Typically, input values are normalized so as to fall between 0.0 and 1.0. Discrete-valued attributes may be encoded such that there is one input unit per domain value.

4.2.2.2 Neural Network Learning Procedure

In a typical neural data processing procedure, the database is divided into three separate portions called training, calibration and verification sets. The training set is used to develop the desired network. In this process, the desired output in the training set is used to help the network adjust the weights between its neurons or processing elements. During the training process the question arises as to when to stop the training, and how many times the network should go through the data in the training set in order to learn the system behavior? These questions are usually referred to as over-training or memorization. Once the network memorizes a data set, it would be incapable of generalization.

In order to avoid over training or memorization, it is a common practice to stop the training process every so often and apply the network to the calibration data set. Since the

output of the calibration data set is not presented to the network, one can evaluate a network's generalization capabilities by how well it predicts the calibration set's output. Once the training process is completed successfully, the network is applied to the verification data set.

During the training process, each artificial neuron handles several basic functions. The function symbolizes the activation of the neuron represented by the unit. First, it evaluates input signals and determines the strength of each one. Second, it calculates a total for the combined input signals and compares that total to some threshold level. Finally, it determines what the output should be. The transformation of the input to output within a neuron takes place using an activation function. Figure 4.10 below shows the commonly used activation functions.

Figure 4.10 Activation functions

All the inputs come into a processing element simultaneously. In response, a neuron either “fires” or “doesn’t fire”, depending on some threshold level. The neuron will be allowed a single output signal, just as in a biological neuron – many inputs, one output signal. In addition, just as things other than inputs affect real neurons, some networks

provide a mechanism for other influences. Sometimes this extra input is called a bias term, or a forcing term, when a system needs to unlearn something.

Initially each input is assigned a random relative weight. During the training process the weight of the inputs is adjusted. The weight of the input represents the strength of its connection to the neuron in the next layer. The weight of the connection will affect the impact and the influence of that input. This is similar to the varying synaptic strengths of biological neurons. Some inputs are more important than others in the way they combine to produce an impulse. Weights are adaptive coefficients within the network that determine the intensity of the input signal. The initial weight for a processing element could be modified in response to various inputs and according to the network's own rules for modification.

The inputs and the weights on the inputs are considered as vectors, such as $I_1, I_2 \dots I_n$ for inputs and $W_1, W_2 \dots W_n$ for weights. The total input signal is the dot, or inner product of the two vectors. The inner product is at its maximum if the vectors point in the same direction. If the vectors point in opposite directions, their inner product is at its minimum. Signals coming into a neuron can be positive or negative. A positive input promotes the firing of the processing element, whereas a negative input tends to keep the processing element from firing. Some local memory is attached to the processing element to store the result of previous computations during the training process. Training is accomplished by modification of the weights on a continuous basis until convergence is reached. The ability to change the weights allows the network to modify its behavior in response to its inputs, or to learn.

In the back propagation algorithm, the network output is compared with the desired output which is part of the training data set, and the difference is propagated backward through the network. Therefore, the weights of the connections between neurons are adjusted during this back propagation of error.

4.2.3 Weakness of Artificial Neuron Network

While the artificial neural network technology is a powerful tool for predicting oil production, there are still some disadvantages in using neural networks. Firstly, ANN models are not transparent (“black box”) and the user finds it difficult to understand the nature of the solution inside. Secondly, the process of finding a suitable topology and identifying the network parameters is empirical and often time-consuming. Thirdly, a neural network can over-fit the data, i.e. the network memorizes the training data, but has low forecast ability. Finally, a neural network can easily become stuck in a local minimum.

4.3 The Neuron-based Decision Tree Learning Model

This section presents our neuron-based decision tree learning model including the motivation and the structure.

4.3.1 Motivation

To overcome the weakness and constraints of the decision-tree learning and artificial neuron network described above, the proposed solution is an aggregation of a neural network with decision tree into a single entity, which will be referred to as the Neural-Decision Tree (NDT) model [24]. The NDT model should be interpreted as a set of

learning procedures, which is based on the same theoretical foundations as Decision Tree and Neural Network learning, rather than a new algorithmic approach.

4.3.2 Structure of the NDT Model

The Neural Decision Tree Learning (NDT) proposed (Lee, et al 2002, 2006) here combines neural network and decision tree algorithms to handle the problem. The NDT model is discussed as follows. The architecture of the NDT model for a mix-type data set, with squared boxes showing the main processing components of the neural network and decision tree is shown below. The NDT algorithm models a single pass process initialized by the acceptance of training data as input to the neural network model. The generated output is then passed on for decision tree construction, and the resulting rules will be the net output of the NDT model. The arrows show the direction of data flow from the input to the output, with the left-hand-side downward arrows showing the processing stages of nominal attributes, and the right-hand-side arrows indicating the corresponding process for numeric-type data. As illustrated in the Figure 4.11 below, the rule extraction steps in this NDT model include the following:

1. Divide the numerical-categorical-mixed dataset into two parts, e.g. numerical subset and nominal subset. For a pure-type data set, no division of the data set is necessary.
2. For the numerical subset, train a feed-forward back-propagation neural network and collect weights between the input layer and the first hidden layer. Then, change each attribute value according to the weights generated by the neural network model.
3. For the categorical subset, each attribute of the dataset for training a neural network needs to be normalized because the neural network only accepts numerical data as input. Based on the normalized dataset, we then train a back-propagation neural

network and classify categorical attributes according to the weights generated by the neural network model. If weight measures suggest a need for merging, specify which two attributes will be merged. A merge can be repeated so it is not constrained to form only bivariate models.

4. Combine the new numerical subset and new categorical subset into a new numerical-categorical-mixed dataset.
5. The new dataset is fed as input. For classification, it is sent to the C4.5 system to generate the decision tree and rules. For prediction, classification and regression tree (CART) is used instead of the C4.5.
6. Some evaluation approaches are executed to obtain a reliable estimate of classifier accuracy or predictor accuracy.

Figure 4.11 The NDT model

4.3.2.1 Categorical Data

To clearly describe these steps, the followings will illustrate the details about how to infer the decision tree and rule for the given example data set in Table 2.1.

For Table 2.1, we firstly encode each categorical attribute value into a numerical one. The encoded results are shown in Table 4.1.

Table 4.1 Encoded Data set.

Sample #	Humidity	Temperature	Wind	Decision: Picnic?
1	0	0	0	0
2	0	0	0	0
3	0	1	1	0
4	1	1	1	0
5	1	0	1	1
6	1	0	1	1
7	1	1	0	1
8	0	1	0	1

Secondly, in order to improve the performance of the network training, the encoded values are passed on for normalization using Equation 4.11 [27], so that the categorical values are between -1 and 1 . The normalized data set is shown in Table 4.2.

$$V_{norm} = \frac{V_{old} - (Max + Min) / 2}{(Max - Min) / 2} \quad (4.11)$$

Table 4.2 Normalized Dataset for categorical type.

Sample #	Humidity	Temperature	Wind	Decision: Picnic?
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	1	1	-1
4	1	1	1	-1
5	1	-1	1	1
6	1	-1	1	1
7	1	1	-1	1
8	-1	1	-1	1

Based on Table 4.2, we then train a feed-forward back-propagation neural network with the following parameter:

- Number of Hidden Layers: 1
- Input Nodes: 3
- Hidden Nodes: 3
- Learning Rate: 0.3
- Momentum: 0.2
- Activation function: Sigmoid Function

The training result of the model is shown in Figure 4.12. Based on the result, we examine the link weights between the input layer and hidden layer. The weights specify the strength of the influence that the input attributes contribute to the first hidden layer nodes. The link weights between the input attributes and the hidden nodes are shown in Table 4.3.

Table 4.3 Weights between the Input Attributes and Hidden Node

	Humidity	Temperature	Wind
Hidden Node 1	-2.99	-3.81	-2.86
Hidden Node 2	-3.19	4.45	-3.3
Hidden Node 3	1.44	0.18	-1.53

Figure 4.12 Training results of the categorical data set

The similarity of input attributes can be calculated by the closeness of the weights contributions. Considering the values of the attributes: in an n-dimension space, the closeness measure between any two attributes can be calculated by Euclidean distance as shown in Equation below:

$$D(a,b) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2} \quad (4.12)$$

Based on this measurement, we can calculate the distance between the three attributes: $V_{Humidity} = (-2.99, -3.19, 1.44)$, $V_{Temperature} = (-3.81, 4.45, 0.18)$, $V_{Wind} = (-2.86, -3.3, -1.53)$ as shown in Table 4.4.

Table 4.4 Distance among attributes.

	Humidity	Temperature	Wind
Humidity	0	8.70	4.62
Temperature	8.70	0	6.52
Wind	4.62	6.52	0

Figure 4.13 is the average linkage diagram to depict the sequence of attributes clustering using average linkage method of cluster analysis.

Figure 4.13 Average Linkage Diagram

In this example, if we cluster three attributes into two clusters: Cluster 1: (Attributes Humidity and Wind, and Cluster 2: Attribute Temperature) based on the average linkage diagram, then the original dataset will be combined and transformed into Table 4.5.

Table 4.5 Combined Transportation Dataset

Sample #	Humidity-Wind	Temperature	Decision: Picnic?
1	High-Windy	Cool	No
2	High-Windy	Cool	No
3	High-Breezy	Mild	No
4	Low-Breezy	Mild	Yes
5	Low-Breezy	Cool	Yes
6	Low-Breezy	Cool	Yes
7	Low-Windy	Mild	Yes
8	High-Windy	Mild	Yes

After putting the combined dataset into decision tree, we get the compact decision tree as shown in Figure 4.4.

4.3.2.2 Numerical Data

To illustrate explicitly the operation of numerical-type data in NDT model, the example data set in Table 2.2 is used here again. First of all, we also use Equation 4.11 to normalize the input attributes' values into the range between -1 and 1 . For categorical-type output attribute, we encoded first before normalization by Equation 4.11.

The normalized data set is shown in Table 4.6.

Table 4.6 Normalized dataset for numeric type

Sample #	Temp. on Day D	Forecast for Day D+1	Decision: Picnic?
1	-1	-0.58	-1
2	1	-0.47	-1
3	-0.47	1	1
4	0.31	-0.13	-1
5	-0.36	0.32	1
6	-0.6	-0.5	-1
7	0.68	0.03	-1
8	0.25	0.39	-1
9	-0.97	-1	-1

According to this normalized data set, we then train a neural network with the following parameters:

- Number of Hidden Layers: 1
- Input Nodes: 2
- Hidden Nodes: 2
- Learning Rate: 0.3
- Momentum: 0.2
- Activation function: Sigmoid Function

Based on these parameters, the training results are shown in Figure 4.14.

Figure 4.14 Training results for the numerical data

Based on the link weights between the input layer and the first hidden layer, we change the attribute values according to these weights. That is, we use the equations below to transform the original data set into a new data set which has equivalent values as the inputs to the hidden layer of the trained neural network.

$$H_1 = 1.53 + 3.76 \times \text{Temp. on Day} + 3.73 \times \text{Forecast on Day} + 1$$

$$H_2 = 1.51 + (-5.04) \times \text{Temp. on Day} + -5.0 \times \text{Forecast on Day} + 1$$

In these equations, H_1 means “new temperature values on Day”; H_2 means “new forecast values on Day+1”. The transformed results in this way are shown in Table 4.7.

Table 4.7 Transformed numerical data set

Sample #	Temp. on Day D	Forecast for Day D+1	Decision: Picnic?
1	-4.39	9.45	No
2	3.54	-1.18	No
3	3.49	-1.12	Yes
4	2.22	0.6	No
5	1.37	1.72	Yes
6	-2.59	7.03	No
7	5.2	-2.07	No
8	3.92	-1.7	No
9	-5.85	11.4	No

Then the transformed data set is used as input for generating the decision tree model. Based on the decision tree, the rules can be derived. A comparison to the results between original and the transformed datasets are shown in Table 4.8.

Table 4.8 Result comparison

Measures	C4.5	NDT
Tree Size (# of nodes)	3	1
Number of Rules (# of leaves)	2	1
Correct Classification Rate (%)	44.44	77.78
Misclassification Rate (%)	55.56	22.22
Mean Absolute Error	0.44	0.37
Mean Squared Error	0.56	0.46

By comparing the C4.5 / NDT results, it is notable that the NDT model reduces the error rate of classification rate from 55.56% to 22.22%. Another notable observation made is the amount by which NDT reduced the complexity of the tree, as shown by the reduced number of decision rules, and the number of leaves.

From the above examples, it is obviously that the NDT model remarkably improves the accuracy, decision tree size, and the number of decision rules based on decision tree learning algorithms.

4.4 Implementation & Tools

This section discusses the implementation details and tools for the development of the NDT model.

- **Data file formats:** The datasets acquired from the real world originate from various data sources and come in a variety of different file formats, presented in a spreadsheet or database. Most spreadsheet and database programs are allowed to export data into a file in comma-separated value (CSV) format as a list of records with commas between items. Therefore, CSV file format is used as a simple input file format in this project to transfer a large volume of spreadsheet or database information between different data sources, without worrying about special file types.
- **NDT model structure:** The NDT development includes two stages. The initial stage of NDT development is neural training, where the link weights between the attributes are obtained. The second stage is to build the decision trees using the transformed dataset. The detail procedure of these two stage is described and customized as follows:

- **Neural-training:** The neural- training system for preprocessing the data was written in Java language and was designed based on the model-view-control (MVC) design pattern. It consists of two main parts: user interface, and the neural network system for merging the categorical data and adjusting the numerical data. The structure of the system is shown in Figure 4.15. The DataController module, which includes component classes of DataSplitter, NeuroTrainer, Merger and Adjuster, is responsible for translating interactions with the graphical user interface into actions that are performed by the MVC Model. The DataSplitter class splits the original numerical-categorical-mixed dataset into two parts: categorical subset and numeric subset. The NeuroTrainer class implements neural network training to extract the link weights between attributes. The Merger Class merges the categorical subset and the Adjuster class adjusts the numeric subset based on the link weights of the attributes.

The user interface includes receiving an input file, setting the options to transform data and saving the output file. The screen of the interface is shown below in Figure 4.16. The user not only can choose whether to merge the nominal data and adjust numerical data together, but also determine the number of times merging is conducted.

Figure 4.15 System structure of the NDT neural training preprocessor

Figure 4.16 Screen of the neural training user interface

- **Decision Tree Learning:** The second stage of decision tree learning is performed using the Weka Toolkit which was developed at the University of Waikato in New Zealand. Weka Toolkit is a collection of state-of-the-art machine learning algorithm tools providing extensive support for the whole process of experimental data mining. It includes tools for data preprocessing, evaluating learning schemes, visualizing the input and the result of learning, and implementations of many different learning algorithms. In addition, a graphical user interface is provided to access to all of its facilities. For classification, we apply the C4.5 decision tree which is called J48 tree in Weka to classify the transformed data set. For prediction problems, the model tree named M5 is used for predicting the numeric data set. The Model tree, which is a type of decision tree with

linear regression functions at the leaves, has been used in a recent successful effort for predicting numeric values [54].

Below is the diagram for the learning process. The initial stage of NDT development is using NDT Data Preprocessor to perform neural training; the second stage is applying the decision tree learning for the preprocessed data.

Figure 4.17 NDT Learning Process

4.5 Model Evaluation: Using Non Petroleum Data Sets

This section extends the theoretical approach to the practical experiment of evaluating the performance of the model on real-world data sets.

The NDT model was applied on data sets from the UCI Machine Learning Repository [55]. Details on the data sets are shown in Table 4.9. Instances with missing attribute have been removed from the data set.

Table 4.9 Description of Data Sets

Data Set	# Instances after Removal of Ones with Missing Values	Nominal Attributes	Numeric Attributes
Glass	214	0	10
Balance-Scale	625	0	4
Bupa	345	0	7
Car Evaluation	1728	4	0
tic-tac-toe	958	10	0
Congressional Vote	435	17	0
Auto Import	159	11	15
Bridge	108	9	4

Based on these datasets, we compare the proposed NDT model with the traditional decision tree algorithm: C4.5. A 10-fold cross-validation was used in all the experiments, where the 10 samples are randomly sampled without replacement from the training set. Each run of the cross-validation process uses 9 of these samples to create a classifier model, and the 10th sample was used as a hold-out test set. The process is repeated exactly 10 times, so that each instance is used exactly once for both training and testing. The average classification accuracy is measured to validate the overall performance of the model.

4.5.1 Results Summary and Analysis

Result summary tables for the three datasets used in the experiments are shown and described separately in each of the following subsections, where the tabularized results are the averages of 10-fold cross-validations for all data sets.

4.5.1.1 Pure Numeric-type Data Set

The results from the data sets that contain numeric-type data only, are shown in Table 4.10. The classification accuracy is defined as the percentage of data tuples which are correctly classified by the model, and this is adopted as the basis for comparison. It can be observed from Table 4.10 that the NDT model demonstrated an improvement in classification accuracy over the C4.5 algorithm in both the pruned and unpruned cases. The values of classification accuracy in the “glass” data set are 68.7% and 68.22% for the NDT model results, 66.35% and 65.88% for the use of the C4.5 algorithm. Similarly, in the “Balance-Scale” data set, the values are 94.88% and 94.56% for the NDT model results, 79.2% and 76.64% for the C4.5 model; and in the “Bupa” data set, they are 71.01% and 70.14% for NDT, 68.98% and 68.11% for C4.5. Also, the NDT model consistently reduced the complexity of the decision tree, as shown by the reduced number of decision rules and the reduced tree size in all three cases. However, by comparing pruned/unpruned results in the NDT model, it can not be said that pruning for the NDT model improves the accuracy measure; hence, the unpruned NDT model is recommended for pure numeric-type data sets. The graphical comparisons of the obtained results by bar chart for decision tree size, numbers of rules and classification accuracy are shown in Figure 4.17, 4.18 and 4.19 respectively.

Table 4.10 Result Summary for Pure Numeric-Type Data Set

Dataset	Measure	C4.5 without Pruning	C4.5 with Pruning	NDT without Pruning	NDT with Pruning
Glass	Tree Size (# of nodes)	30	30	28	28
	Number of Rules	59	59	55	55
	Test Set Size	214	214	214	214
	Correct Classification Rate (%)	66.35	65.88	68.70	68.22
	Mean Absolute Error	0.1029	0.1062	0.0953	0.0971
	Mean Squared Error	0.2936	0.2922	0.2826	0.284
Balance- Scale	Tree Size (# of nodes)	60	58	18	18
	Number of Rules	119	115	35	35
	Test Set Size	625	625	625	625
	Correct Classification Rate (%)	79.2	76.64	94.88	94.56
	Mean Absolute Error	0.1595	0.188	0.0433	0.0466
	Mean Squared Error	0.3487	0.3688	0.1815	0.1858
Bupa	Tree Size (# of nodes)	27	11	8	5
	Number of Rules	53	21	15	9
	Test Set Size	345	345	345	345
	Correct Classification Rate (%)	68.98	68.11	71.01	70.14
	Mean Absolute Error	0.3662	0.3951	0.3749	0.3699
	Mean Squared Error	0.5089	0.4864	0.4545	0.4659

Tree Size Complexity

Figure 4.18 Comparison of tree size complexity for numeric datasets

of Rules

Figure 4.19 Comparison of decision rules for numeric datasets

Classification Accuracy %

Figure 4.20 Comparison of classification accuracy for numeric datasets

4.5.1.2 Pure Categorical-Type Data Set

The experimental results using the categorical-type-only data sets are shown in Table 4.11, where the NDT process required merging of categorical-type attributes. For the “car” data set, the classification rate decreased from 94.155 to 81.713 for the NDT model, while the tree size increased from 186 to 293 and the number of rules also increased from 134 to 237. For the “tic-tac-toe” data set, the classification rate decreased from 86.43 to 82.15, while the number of rules increased from 208 to 295 with the same tree size. For the “Vote” data set, the classification rate also decreased from 96.32 to 94.02, while the number of rules also decreased from 37 to 13 with the same tree size. In [24], Lee and Yen suggested that grouping of attributes which favours a balanced contribution of the corresponding link weights to the hidden nodes should result in a higher classification result. However, this did not result in better performance in these data sets, the NDT columns in Table 4.11 show the best observed performance from testing all possible permutations of attribute groupings. Furthermore, Lee and Yen’s proposal of NDT did not provide a detailed description of how links weights from neural networks on nominal data can be utilized and applied to the C4.5 algorithm. As the user often lacks prior knowledge about the statistical distributions and attribute-dependency relationships of the data samples, a decision made on attribute grouping is only as good as the user’s knowledge.

Table 4.11 Result Summary for Pure Categorical-Type Data Set

Dataset	Measure	C4.5 without Pruning	C4.5 with Pruning	NDT without Pruning	NDT with Pruning
Car	Tree Size (# of nodes)	186	182	293	181
	Number of Rules	134	131	237	147
	Test Set Size	1728	1728	1728	1728
	Correct Classification Rate (%)	94.155	92.361	82.870	81.713
	Mean Absolute Error	0.0343	0.0421	0.0802	0.1033
	Mean Squared Error	0.1593	0.1718	0.2513	0.2627
tic-tac-toe	Tree Size (# of nodes)	139	75	241	115
	Number of Rules	208	112	295	142
	Test Set Size	958	958	958	958
	Correct Classification Rate (%)	86.43	84.03	85.69	82.15
	Mean Absolute Error	0.1485	0.2046	0.1525	0.2195
	Mean Squared Error	0.343	0.357	0.337	0.371
Vote	Tree Size (# of nodes)	19	5	13	7
	Number of Rules	37	9	23	13
	Test Set Size	435	435	435	435
	Correct Classification Rate (%)	96.32	95.63	95.40	94.02
	Mean Absolute Error	0.0552	0.0712	0.059	0.0788
	Mean Squared Error	0.1748	0.1915	0.2114	0.2227

4.5.1.3 Mixed-Type Data Set

The results on the mixed-type data sets in Table 4.12 shows a superior performance of the NDT model compared to the regular C4.5 as reflected by an increase in correct classification rate. However, the tree size and number of rules increase as well, which means the tree complexity generated by the NDT model is much higher than the regular C4.5. The graphical comparison of tree size, number of rules and classification accuracy are shown in Figure 4.23, 4.24 and 4.25 respectively. For the “auto import” data set, the classification rate increased from 70.18 to 71.93 for the NDT model, while the tree size increased from 19 to 25 and the number of rules increased from 17 to 24. For the “bridge” data set, the classification rate increased from 81.88 to 83.62 for the NDT model, while the number of rules increased from 135 to 209 with the same tree size. This provides supportive evidence towards the hypothesis made by Lee [24] that NDT improves the average results of C4.5 in terms of prediction accuracy. When the NDT model was applied to both the categorical and numerical types of data in the data set, the results generated are superior to those from modeling only one kind of data. In comparing application of NDT to each of the two types of data, the results from modeling the numeric data only are better than modeling the categorical data in some cases. It seems the contribution of the numeric data modeling is greater than that from the categorical data when the NDT model is applied. The reason for the better performance from modeling the numeric data set is likely to be because the numeric data has more direct influence in generating the link weights between the input and hidden nodes in the neural network.

Table 4.12 Result Summary for Mixed-Type Data Set

Dataset	Measure	C4.5	NDT on numeric subset	NDT on categorical subset	NDT on entire dataset
Auto Import	Tree Size (# of nodes)	19	17	25	25
	Number of Rules	17	16	24	24
	Test Set Size	114	114	114	114
	Correct Classification Rate (%)	70.18	70.18	72.81	71.93
	Mean Absolute Error	0.3374	0.3396	0.345	0.356
	Mean Squared Error	0.445	0.448	0.452	0.456
Bridge	Tree Size (# of nodes)	101	191	135	177
	Number of Rules	135	227	170	209
	Test Set Size	108	108	108	108
	Correct Classification Rate (%)	81.88	83.18	82.46	83.62
	Mean Absolute Error	0.202	0.1896	0.1983	0.1876
	Mean Squared Error	0.3982	0.371	0.3942	0.3714

Figure 4.21 Comparisons of tree size for mixed-type datasets

Figure 4.22 Comparison of decision rules for mixed-type datasets

Figure 4.23 Comparison of classification accuracy for mixed-type datasets

4.5.2 Summary over All Data Sets

Figure 4.26 gives a summary graph of the relative percentages of improvement/degradation made by the NDT compared to the regular C4.5, where a positive measure is interpreted as improvement, such as simpler tree structure and higher accuracy, and a degraded performance is given a negative value.

Toward the numeric data (Glass, Balance-Scale and Bupe data set), the NDT demonstrates an improvement in terms of tree size, numbers of rules and prediction accuracy. This proves that the NDT is fit for handling the numeric data. Toward the categorical data (Car, Tic-tac-toe and Vote data set), the NDT presented a negative result

which may be due to the weakness of NDT in handling categorical data. Toward the mixed-type data (Auto Import and Bridge data set), the actual treatment of NDT resulted in a slight improvement in terms of the prediction accuracy, where the number of the tree size and rules give rise heavily.

The graph also shows that NDT resulted in the highest improvement in the Balance-Scale and Bupa data sets, which are characterized by medium data set size and small number of attributes. This suggests that NDT may be more adapted to handling simple data sets that have fewer attributes.

Figure 4.24 % Improvement/Degradation made by NDT for all Data Sets.

4.5.3 Strengths and Limitations

The results of NDT on numeric-type data show a distinct improvement over C4.5 in terms of both reducing the complexity of the tree and increasing prediction accuracies in numeric data. This makes the utilization of link weights for modifying numeric data an attractive approach, as it is able to improve the accuracy of C4.5, as well as simplify the decision tree structure. This likely would produce rules with less redundant information. An added advantage of using neural network as a pre-processor of C4.5 is that there is an

improved robustness against noisy data, since attribute testing in C4.5 is difficult with data containing missing attribute values.

A limitation of the NDT model on numeric-type data set in the proposed approach of transforming the original numeric input to that of the input to the hidden layer using the inner product rule, requires the first hidden layer to have the same number of nodes as the number of inputs. This reduces the flexibility of the neural network learning environment, and the user is confined to experiment with varying the number of additional hidden layers and nodes used.

As discussed earlier in the experimental results, the major weakness of the NDT model is expressed in the abstractness of its handling of categorical-type attributes. Lee and Yen's proposal of NDT did not provide a detailed description of how links weights from neural networks on nominal data can be utilized and applied to the C4.5 algorithm. As the user often lacks prior knowledge of the statistical distributions and attribute-dependency relationships of the data samples used, a user's decision made on attribute grouping cannot guarantee good results would be generated.

The comparison above shows that the NDT model can improve upon the classification accuracy of the C4.5 algorithm for numeric type and mixed-type datasets. Since the petroleum production data consist of primarily numeric data, the NDT model is deemed to be applicable for petroleum prediction.

Chapter 5

Application of NDT to Petroleum Production

In this chapter, the NDT model and framework were applied for prediction of oil well production in southwestern region of the province of Saskatchewan. A comparison of the results obtained using our method to the tradition decision tree method will be presented as well.

5.1 Prediction of Petroleum Production

This section presents a prediction of petroleum production using our NDT model, which is capable of identifying the core variables most relevant to oil production. The performance of the NDT model is assessed using data sets obtained from a pool of wells in Saskatchewan, Canada.

5.1.1 Domain Concepts

In designing a system for prediction of petroleum production, many of the concepts identified and addressed by existing literature are considered in this project. These issues are discussed in the following subsections.

1. **Petroleum:** Petroleum is a liquid commonly called “crude oil”. However, both natural gas and crude oil are often called petroleum. Neither natural gas nor crude oil is a single chemical compound—both are mixtures of numerous compounds called hydrocarbons [56]. Oil can be categorized into four crude types based on density: light, medium, heavy and bitumen. In this study, only medium oil was considered.

2. **Reservoir:** A petroleum reservoir is a volume of porous sedimentary rock that has been filled by petroleum and possibly other fluids. Oil, along with varying amounts of water and gas, reside in the porous spaces of the rock [57].
3. **Well:** Many wells can be drilled to recover fluids, including oil and gas, inside the boundary of each reservoir. Wells can be categorized according to their usages. In this study, the term ‘well’ means producer wells [57].
4. **Porosity:** In reservoir geology, porosity is a measure of the void space within a rock, which is expressed as a fraction (or percentage) of the bulk volume of that rock. The general expression for porosity ϕ is

$$\phi = \frac{V_b - V_s}{V_b} = \frac{V_p}{V_b} \quad (5.1)$$

where V_b is the bulk volume of the rock, V_s is the volume occupied by solids, and V_p is the pore volume.

5. **Permeability:** Permeability is defined as a measure of the ability of a rock to permit passage of fluids. Permeability reflects the connected character of pores, fractures, joints, and faults within a rock unit. The quantitative definition of permeability was first presented in an empirical relationship by Darcy’s Law, which describes the flow of fluids through porous media: permeability is computed as the factor k in Darcy’s formula shown in Eq.

$$Q = -\frac{k}{\mu} \times \frac{dP}{dl} \quad (5.2)$$

where Q is the apparent flow velocity (cm/s), μ is the viscosity of the flowing fluid, p is the pressure, l is the length, k is permeability.

6. **First Shut-in Pressure:** Pressure is the force per unit area exerted by or on a fluid. It is a proxy variable for initial formation pressure.
7. **Water Saturation:** The fraction of water in a given pore space. It is expressed in volume, percent or saturation units. Water saturation is the fraction of formation water in the undisturbed zone. The saturation is known as the total water saturation if the pore space is the total porosity, and the effective water saturation in the pore space is the effective porosity. If used without qualification, the term usually refers to the effective water saturation [20].
8. **Water flooding:** A method of secondary recovery in which water is injected into the reservoir formation to displace residual oil. The water from injection wells physically sweeps the displaced oil to adjacent production wells.
9. **Residual oil:** Oil that does not move when fluids flow through the rock in normal conditions, for example during primary and secondary recovery, and invasion.

General speaking, the permeability and porosity values are obtained from laboratory core analysis, and the first shut-in pressure data are derived from drilled stem test (DST) analysis.

5.1.2 Study Area

The applicability and performance of the NDT model is investigated in the Weyburn Field. The Weyburn Field is one of the largest reservoirs located in the north of the Williston Basin in southeastern Saskatchewan, Canada. Petroleum production of this reservoir is from the Mississippian age carbonates, at a depth of more than 1300 meters.

It was discovered in 1954 and produced oil until February 1963, when water flooding was implemented. The maximum production appeared in 1965 at 46,000 barrels/day.

5.1.3 Identification of Data Sets and Parameters

The data set is obtained from Saskatchewan Industry and Resources. It consists of 320 data records of production rate, core analysis and pressure data from the Weyburn Midale field in Saskatchewan, Canada. There was 50 years of production data available: from 1960's to 2006. The core and DST analysis provide us with the geoscience parameters, such as permeability, porosity and first shut-in pressure.

Since the modeling process requires accurate identification of input parameters, all the relevant parameters related to petroleum prediction should be considered. However, due to data limitations, we chose five most significant parameters: permeability, porosity, first shut-in pressure, residual oil and saturation of water for modeling. In our investigation on modeling production data, four different models were developed based on different combinations of parameters and data. The first model predicted total production using three geosciences parameters of permeability, porosity and first shut-in pressure as input variables. Then, two different models with different input parameters were developed to predict production in the post water flooding stage only: (1) one model includes only three geosciences parameters of permeability, porosity and saturation water, and (2) another model extends the first model by using four geosciences parameters of permeability, porosity, water saturation and residual oil. The fourth model was built based on an assumption that the production data exhibit a harmonic trend in the particular area of study. Therefore, instead of directly estimating the wells' production, the last model predicted the behaviors or the harmonic trends of the wells' production

data based on permeability and porosity. The model predicted the production trend of oil in the pre-water flooding stage. In the model, the geoscience data were selected and grouped into specific rock formations by a petroleum engineer. Details of the variations in these models will be discussed in the following subsections.

5.1.4 Data Preprocessing

The original data we captured were incomplete (i.e. they lack attribute values or certain attributes of interest, or containing only aggregate data), noisy (they containing errors, or outlier values that deviate from the expected), and inconsistent (they contain discrepancies in the tag names used to label the attributes). In other words, there are reported errors, unusual values, and inconsistencies in the data recorded for some transactions. In addition, a large amount of redundant data can slow down or confuse the analysis process. Therefore, data preprocessing was conducted to improve the efficiency and ease of the analysis process. Since the geoscience values of permeability, porosity, water saturation, and residual oil are measured in different depth ranges through each well in the core analysis, we calculated the weighted averaged value and fed them as the corresponding values of each geoscience parameter for each well.

5.2 Experiments and Results

This section will present the experimental results and analysis of the four models built by the NDT structures. These four models can be classified into three different types: overall oil production model, post-water flooding model and trend predicting model. Details of each model are presented below.

5.2.1 Overall Oil Production Model

First, the entire oil production data set was used, so the model was called an “Overall oil production model”. The model includes both production rates and the three geoscience parameters of permeability, porosity and first shut-in pressure as input variables to perform the prediction task. In this study, we attempt to build the classification model by using the data discretization technique to divide the numeric production data into intervals. Entropy-based discretization technique [27] is used here to discretize the numeric class attribute into different specific sub-ranges. This is the most commonly used splitting technique which explores distribution information in its calculation and determination of split-points which are data values for partitioning an attribute range [27]. For example, to discretize the numeric class attribute, A, the method selects the value of A that has the minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Entropy-based discretization uses class information to reduce size of data set, hence it is more likely that the interval boundaries, or split-points, are in places that may help improve classification accuracy [27]. Using the entropy-based method, six interval ranges of the production data are generated as follows, ordering from low to high.

Class A: [2,000-97,767], Class B: [97,767-199,633], Class C: [199,633-269,082],

Class D: [269,082-373,839], Class E: [373,839-469,633], Class F: [469,633-620,000].

Figure 5.1 shows the data distribution of each class. As seen from the graph, most of the data fall into Classes A and B. This is due to the fact that most of the oil wells in the real world produce small amounts of oil.

Figure 5.1 Distribution of data in each classes

The NDT model generates 24 rules and 17 rules by the unpruned / pruned tree respectively (it is not shown due to the large size of the tree). The topmost splitting attribute is permeability, which means that the permeability has the maximum correlation with the predicted production. The attribute at the lower level is porosity which frequently appears in the sub-branches. First shut-in pressure is the less important attribute and appears at or near the leaf node in the trees.

As shown in Table 5.1, in comparing the results generated by the C4.5 decision tree learning algorithm with the NDT model, the NDT model reduces the tree size and number of rules by half with only 5% decrease in classification accuracy. A decrease in the number of rules is an improvement because they are more easily validated by petroleum engineers. Hence, the NDT model proposed in this study provides better explanation capability, since it generates a comparative smaller rule set with an acceptable level of classification accuracy. In other words, with the sacrifice of a little classification accuracy, the NDT model is able to provide some explicit heuristics for classification that support predicting oil production from a new well.

Table 5.1 Comparison of NDT model and C4.5 results

Measures	C4.5 without Pruning	C4.5 with Pruning	NDT without Pruning	NDT with Pruning
Tree Size (# of nodes)	131	117	67	53
Number of Rules (# of leaves)	52	45	24	17
Test Set Size	320	320	320	320
Correct Classification Rate (%)	86.88	85.31	80.94	79.69
Misclassification Rate (%)	13.12	14.69	19.06	20.31
Mean Absolute Error	0.13	0.15	0.19	0.20
Mean Squared Error	5.51	6.90	11.63	13.20
Computation Time (milliseconds)	15692	14070	28211	26468

5.2.2 Zooming-in: Post-Water Flooding Models

Water flooding is often used to enhance oil recovery. After water is injected into the well, the mechanism of the well is changed and oil well production is significantly increased. In order to better model the oil production of the post-water flooding stage, two models were built: one with three geoscience parameters permeability, porosity and saturation water only, and the other contained the four geoscience parameters of permeability, porosity, saturation water and residual oil. Only 121 wells in total are available, of which 81 wells were used for training and 40 for testing. The two post water flooding models are discussed as follows:

5.2.2.1 Second Model: Post-Water Flooding Model with Three Parameters

This model was set up for the stage of post-water flooding prediction with the three most significant geoscience parameters of permeability, porosity and saturation water. The reason why we use saturation water instead of first shut-in pressure as an input

geoscience parameter is because first shut-in pressure only happens in the stage of pre-water flooding and hence it is useless in the stage of post-water flooding.

Interval ranges of the data set generated by the method described above are:

Class A: [0-6707.9], Class B: [6707.9-18001.85], Class C: [18001.85-36400.2],

Class D: [36400.2-55736.95], Class E: [55736.95-65484.95],

Class F: [65434.95-83956.2].

An artificial neural network (ANN) model was built, as well, using the same input and output variables as the NDT model. The popular three-layer feed forward ANN topology was employed, and the sigmoid activation function was used. The classical backpropagation training method of ANN was adopted. The Weka [59] software tool kit was used. The performance of the model is summarized in Table 5.2.

Table 5.2 Comparison of NDT model and ANN results

Measures	NDT cross-valid	ANN cross-valid
Number of Rules or hidden nodes in ANN	35	3
Number of Samples	122	122
Misclassification Rate (%)	71.0744	71.1
Root mean square error(RMSE)	0.4033	0.3465
Mean Absolute Error	0.2132	0.2277
Relative absolute error (%)	89.4032	95.5131
Root relative squared error (%)	116.8512	100.4011

From the table, it can be seen the NDT model prediction result is similar to that of the ANN model trees with almost the same misclassification rate, and its RMSE is only 0.056% higher than that of ANN, the mean absolute error is 0.0145 lower, the relative absolute error is 6.1% lower and the root relative squared error is 16.45% higher. However, both NDT and ANN have a high error in predicting the stage of post-water

flooding production. The results suggest that either the input data were not processed efficiently or crucial attributes were missing.

5.2.2.2 Third Model: Four Parameters Post-water Flooding Model

In order to improve the effectiveness of the predictive model, it was extended by adding another geoscience parameters: residual oil. The same configuration was built and generated the results as shown in Table 5.3

Table 5.3 Comparison of NDT model and ANN results

Measures	NDT cross-valid	ANN cross-valid
Number of rules or hidden nodes in ANN	30	3
Number of samples	122	122
Misclassification rate (%)	76.8595	72.7273
Root mean square error(RMSE)	0.4233	0.3488
Mean absolute error	0.2215	0.223
Relative absolute error (%)	92.884	93.5436
Root relative squared error (%)	122.6586	101.0489

As shown in Table 3, both NDT and ANN model have even higher error compared with the previous one. This means the parameter of residual oil has little influence on oil production and more attributes cannot lead to a better performance. Interestingly, in the nodes of NDT model corresponding to higher production, instead of permeability, porosity begins to appear at the top layer of the generated NDT model tree. This means that this attribute of porosity became the most important one for predicting the post-water flooding production. The physical explanation of this fact is that the oil flow in the rock rapidly increase after the water flooding, and with more void space within a rock, the oil can be easier pushed onto the surface.

5.2.3 Fourth Model: Trend Predicting Model

The error analysis of the post-water flooding model reported above led us to the conclusion that the mechanisms used so far were somewhat superficial, and these configurations may not allow the data-driven models to classify oil production into physically interpretable classes.

In order to improve the effectiveness of the predictive model, we attempt to predict the trend of the oil well instead of directly estimating its production. We worked with petroleum engineers who preprocessed the geoscience data by filtering the data based on their expertise. After preprocessing the data, 31 oil wells, where the production values have a harmonic relationship over time, were chosen and used to build the model. The empirical Arps decline equation which represents the relationship between production rate and time for oil wells during the pseudosteady-state period, is shown as follows [6]:

$$q(t) = \frac{q_i}{(1 + D_i t)} \quad (5.3)$$

where $q(t)$ is the oil production rate at time t , D_i is the initial nominal decline rate and q_i is the initial oil production rate. In addition, the values for the parameters of permeability and porosity will not be averaged from all depths, instead, the average of values from one particular formation only will be taken. The production value will be calculated as the summation of productions from different formations of one well.

To build this model, we first calculated the q_i and D_i of each well. The Sigma Plot software package [60] was used to calculate the values of q_i and D_i based on the production value of each well. Instead of directly predicting the production, we focused on predicting the q_i and D_i values in the harmonic equation which would give us a better understanding of the trend and behavior of an oil well. In this way, we can also capture

the characteristics of the reservoir and dataflow organization. In the model, we only used permeability as the input variable to predict the value of D_i , and used porosity to predict the value of q_i , 28 wells were used for training and 3 for testing. The value range of D_i is: $[0, 0.38]$ and the value range of q_i is: $[23.798, 161.699]$.

The experimental results of NDT and ANN model are shown in Table 5.4 and 5.5, respectively.

Table 5.4 Experimental results of NDT

Measures	NDT Qi Prediction		NDT Di Prediction	
	Training	Testing	Training	Testing
Number of Samples	28	3	28	3
Root mean square error(RMSE)	17.17	42.31	0.0653	0.0094
Mean Absolute Error	13.48	38.32	0.0315	0.0086
Relative absolute error (%)	62.18	124	92.3	62.8
Root relative squared error (%)	61.39	127.86	94.6	51.3
Correlation coefficient	0.79	-0.50	0.52	0.95

Table 5.5 Experimental results of ANN

Measures	ANN Qi Prediction		ANN Di Prediction	
	Training	Testing	Training	Testing
Number of Samples	28	3	28	3
Root mean square error(RMSE)	32.43	38.46	0.0607	0.0546
Mean Absolute Error	25.37	32.11	0.039	0.0395
Relative absolute error (%)	117.03	104	115.36	287.83
Root relative squared error (%)	115.96	116.23	87.82	298.03
Correlation coefficient	0.337	0.91	0.484	0.94

It can be observed from the above table, the trend model demonstrated an improved performance in prediction of Q_i and D_i by both NDT and ANN models. In the

experiment for Di prediction, the Mean Absolute Error of the NDT model is 0.0086, while the ANN model is 0.0395. Thus, the NDT model is better than ANN model in Di prediction. In comparison with the value range of Di (0-0.38), the Mean Absolute Error in the NDT model is about 10%, which is small. And in Qi prediction, the Mean Absolute Error of the NDT model is 38.32, while the ANN model is 32.11. Since the value range of the Qi value is from 23.798 to 161.699, the Mean Absolute Error is also about 10%, which is still acceptable when comparing with the range of the Qi. However, the performance of ANN model is better than NDT model in Qi prediction.

On the other hand, the relative absolute error and the root relative squared error are just the total absolute error, with the same kind of normalization. In these two error measures, the errors are normalized by the error of the predictor that predicts average values [63]. If values of these relative errors are high, it means the predicted values are scattered away from the average predicted value. If their values are low, then the predicted value tend to cluster close to the predicted average. Hence, the relative absolute error and root relative squared error only reflect the basic predictability or unpredictability of the output variable, not the accuracy of the model. From Table 4.5 and Table 4.5, it shows that for testing, these relative errors of Di in ANN model are higher than NDT model, which means the output value of the NDT model tends to lie fairly close to its average value, and therefore easier to predict compared with the ANN model. However, in Qi prediction, these relative errors of both NDT and ANN are high, which demonstrates that predictability of the model is low. In summary, the performance of NDT and ANN in predicting Qi and Di is inconclusive in that: the NDT is better in Di prediction according to the Mean Absolute Error, and the reverse is true for Qi prediction.

Chapter 6

Conclusion

The initial research goal of this thesis is to perform prediction modeling for petroleum production, and the approach we used is the neural decision tree (NDT) model. It is shown that the NDT model, being analogous to decision tree algorithms, have certain advantages compared to artificial neural network.

We first evaluated the NDT model by applying it to different kinds of data, and comparing the result to those from the regular decision tree model. At the beginning, the NDT model finds dependencies among attributes using the neural network approach, where categorical data and numeric data are trained separately. Then, the traditional decision-tree learning algorithm is applied to the trained data sets pre-processed by the neural network.

In our evaluation experiment, three different types of data are used and the NDT process proves it can perform better by capturing the dependencies among attributes for the data set that involves both numeric data and mix-type data. The hypotheses of decreased tree complexity and increased classification accuracies made by Lee and Yen [24] are verified by the experiments on pure numeric-type and mix-type data sets. However, the results on categorical-type data sets suggest that the NDT process does not offer a single ‘best’ approach to handling only nominal data, and an improved mechanism for handling categorical-type data is needed. In addition, the dependencies among categorical data and numeric data cannot be captured in this model. For example, if there are dependencies between attributes “humidity” (categorical data) and “temperature”

(numeric data), the model fails to capture this kind of dependency when the original mixed-type data is divided into categorical and numeric data subsets.

In a next step of this research project, the NDT model was extended to focus on petroleum prediction, where different strategies and different parameters combinations are employed.

Firstly, an overall oil production model was developed using three geoscience parameters (permeability, porosity and first shut-in pressure) with average classification accuracy of 80.31%. Although this classification accuracy has a 5% decrease compared with the regular C4.5 model, the NDT model reduces the tree size and number of rules by half, which makes it easier for petroleum engineers to analyze results from the rules it generated. In addition, permeability is shown to be more important in predicting the petroleum production.

Secondly, post-water flooding models were developed using three geoscience parameters and four geoscience parameters respectively. In spite of low classification accuracy in these models, the models demonstrate porosity is the most significant factor in prediction of petroleum production instead of permeability. This finding is consistent with the fact that the oil well had different characteristics after water-flooding was applied. Also, the reason for the low classification accuracy using a model with only three attributes is likely because the input data have not been pre-processed adequately due to lack of domain knowledge about the post-water flooding conditions of the well. The oil well may contain different rock formations which have different physical properties. However, this difference is ignored.

Thirdly, a trend model was developed in order to improve the effectiveness of the prediction model. In this model, the oil wells with harmonic relationships between the variables of production rate and time were selected for prediction of parameters (Q_i and D_t) using the empirical Arps decline equation. The model demonstrated an improved performance in prediction, with the Mean Absolute Error of 38.32 in predicting Q_i , and 0.0086 in predicting D_t respectively. By comparing these results with the value range of Q_i [0-0.38] and D_i [23.798, 161.699], we found that the Mean Absolute Error of them is small.

In addition, we also found that the performance of NDT model is comparable to the artificial neural network. The advantages of the NDT model when compared to ANN model are:

- The generated tree-like structure is easy to understand for decision makers. It makes it possible for a petroleum engineers to have a good overview of the relationship between the geoscience attributes.
- The training process of the NDT model is much faster than ANN and always converges.
- The knowledge inside a NDT tree may also help in parameters selection and assessing the interdependent relationships among attributes.

Last but not least, we observed that although the general prediction performances of the NDT model are good, the prediction inaccuracies of some models are high. The reasons for this high inaccuracy may be due to data-related problems, which include:

- The limited available data in the model. For example, only 31 oil wells, with a harmonic relationship between production and time, can be identified for training the trend model after preprocessing the geoscience data.

- The oil well may contain different rock formations having different physical properties. However, this difference is ignored.
- The weighted average data may be too rough to capture the oil well property. Although the pressure data took this into consideration for different rock formations in our experiment, however, this is not a significant attribute in predicting production rate.

In order to validate the model or improve the prediction accuracy, the following recommendation could be given:

- Instead of emphasizing the technique's ability to on generate perfect rules that guarantee to give the correct classification on all instances in the training set, it is preferable to generate "good" rules that avoid overfitting the training set and thereby stand a better chance of performing well on new test instances. Some mechanisms can be used for achieving this goal, such as reduced-error pruning [63], this topic will be investigated in the future.
- Since relation between the geoscience parameters are highly nonlinear, the prediction would likely be improved by using more geoscience parameters as input attributes. In this way, more dependencies among attributes are considered and identified in the NDT model.
- In the situation when data-driven models are used (NDT or ANN etc.), some domain knowledge of petroleum engineering is needed to analyze and validate the rules and results that are generated by these models.

Chapter 7

Future Work

It is shown that the problem of predicting oil production is not trivial. Although we have utilized different approaches on different aspects of the problem, such as data preprocessing by NDT model, different mechanisms of input parameters, and AI techniques, the results are still not as accurate as desired. It seems that production depends on too many factors that cannot all be taken into account. Moreover, each factor such as permeability cannot be measured accurately because the values vary at different locations and in different rock formations.

The NDT model which involved primarily numeric data was applied to of petroleum prediction in this research project. As future work, the project can be extended to include more data, so that further application on categorical data and mixed-type data can be conducted. Further research is needed to define formal processes of integrating attribute grouping into the construction of a multivariate decision tree for categorical data modelling. Without a defined process, the multivariate tree model generated is entirely dependent on the user's interpretation of link weights, which are used to prune errors as discussed earlier. A heuristic search based on link weights, should also be considered to construct a multivariate tree that favours low class entropy, as well as supports meaningful attribute groupings. It is our belief that, if all the dependencies among input attributes can be captured, more improvement in data classification accuracy can be realized.

For future work, some models with high reported errors are needed to be further investigated in different ways. On the other hand, it would also be of interest to apply the approach for prediction of gas and water in the oil wells, integrating other geoscience parameters from various sources of petroleum information, including DST and well log, in addition to core analysis. More variables in data modeling may improve the accuracy of the model. Also, we will work with petroleum engineers to group geoscience data into different rock formation groups and develop one model for each group. Permeability and porosity will not be averaged from all depths but instead averaged from values derived from one formation only. Also, the production will be calculated as the summation of productions from different formations of one well.

Furthermore, the NDT model implemented in this project can not deal with the missing attribute values. Thus, this issue needs to be investigated to define an effective process to deal with missing values to increase prediction accuracy. One possible way is to use the attribute mean to fill in the missing value or to use the attribute mean for all samples belonging to the same class as the given tuple. However, this method biases the data, and the filled-in value may not be correct. Another popular solution to this problem is using the most probable value to fill in the missing value [27], which uses all the information from the present data set to predict the missing value. The judgment on “the most probable value” is made by the user.

Bibliography

- [1] L. Mattar. A Systematic and Comprehensive Methodology for Advanced Analysis of Production Data. SPE paper 84472 presented at the SPE Annual Technical Conference and Exhibition held in Denver, Colorado, USA, 5-8 October 2003.
- [2] Mohaghegh, S. D., Gaskari, R. and Jalali, J. New Method for Production Data Analysis to Identify New Opportunities in Mature Fields: Methodology and Application. SPE paper 98010 presented at the 2005 SPE Eastern Regional Meeting held in Morgantown, W. V., 14-16 September 2005.
- [3] Li, K. and Horne, R.N, A Decline Curve Analysis Model Based on Fluid Flow Mechanisms, SPE 83470, presented at the 2003 SPE/AAPG Western Regional Meeting, Long Beach, California, 19–24 May 2003.
- [4] Fetkovich M.J. Decline Curve Analysis Using Type Curve, J. Petroleum Tech, p 1065-1077, June 1980.
- [5] Fetkovich, M.J., Fetkovich, E.J., and M.D. Fetkovich. Useful Concepts for Decline Curve Forecasting, Reserve Estimation, and Anaylsis. SPE paper 28628 presented at the SPE 69th Annual Technical conference and Exhibition held in New Orleans, LA, p. 217-232, Sep. 25-28, 1994.
- [6] Arps, J. J., Analysis of Decline Curves, Trans AIME 160, 228-247, 1945.
- [7] Nguyen H.H and Chan C.W., Applications of data analysis techniques for oil production prediction. Engineering Applications of Artificial Intelligence, Volume 18, Issue 5, Pages 549-558, Aug 2005.
- [8] Lefkovits, H.C. and Matthews, C. S. Application of Decline Curves to Gravity-Drainage Reservoirs in the Stripper Stage. Trans. AIME, 213, 275-280, 1958.

- [9] Fetkovich, M.D., Guerrero, E.D., Fetkovich, M.J., and Thomas, L.K. Oil and Gas Relative Permeabilities Determined from Rate-Time Performance Data. SPE paper 15431, presented at the 61st SPE Annual Technical Conference and Exhibition, New Orleans, LA, USA, October 5-8, 1986.
- [10] Aronofsky, J. S., Masse, L., Natanson, S. G. A model for the mechanism of oil recovery from the porous matrix due to water invasion in fractured reservoirs. Trans. AIME 213: 17-14, 1958.
- [11] R.O. Baker and N.K. Spenceley. Using An Analytical Decline Model to Characterize Naturally Fractured Reservoirs. SPE paper 39623 presented at the SPE/DOE Improved Oil Recovery Symposium held in Tulsa, Oklahoma, USA, 19-22 April 1996.
- [12] Li, K. and Horne, R.N. Extracting Capillary Pressure from Spontaneous Imbibition Data in Oil-Water-Rock Systems. SPE paper 80553 presented at SPE Asia Pacific Oil and Gas Conference and Exhibition held in Jakarta, Indonesia, 15-17 April 2003.
- [13] Ahmed H. el-Banbi, and Robert A. Wattenbarger. Analysis of Commingled Tight Gas Reservoirs. SPE paper 36736, October 1996.
- [14] Minsky, Marvin and Seymour Papert. Perceptrons: An introduction to Computational Geometry, MIT Press, 1969.
- [15] Chan C.W. and Nguyen H.H. An Analysis of Artificial Neural Networks versus Curve Estimation Techniques in Predicting Petroleum Production. INTERNATIONAL SOCIETY FOR ENVIRONMENTAL INFORMATION SCIENCES. Paper EIA03-038, Volume 1, Pages 375-385, 2003.
- [16] Mohaghegh, S., Balan, B., Ameri, S., West Virginia University. State-of-the-art in Permeability Determination From Well Log Data: Part 2: Verifiable, Accurate

Permeability Prediction, the Touch-Stone of All Models. SPE Page 30979, proceedings at the SPE Eastern Regional Conference & Exhibition held in Morgantown, West Virginia, U.S.A., 17-21 September 1995.

[17] Virtual Intelligence and its Applications in Petroleum Engineering. JPT - Journal of Petroleum Technology, September 2000. Available at <http://www.intelligentsolutionsinc.com/part1.htm>

[18] Neural Networks : Introduction. Available at <http://library.thinkquest.org/C007009/introduction/types/types-5.html>

[19] Tu, JV. Advantages and disadvantages of using artificial neural networks versus logistic regression in predicting medical outcomes. J Clin Epidemiol, 49: 1225–31 1996.

[20] Hanh. Nguyen, Christine W. Chan, Malcolm Wilson: Prediction of oil well production: A multiple-neural-network approach. Intell. Data Anal. 8(2): 183-196, 2004.

[21] Mohaghegh, S., Arefi, R., Ameri, S., and Rose, D., Design and Development of an Artificial Neural Network for Estimation of Formation Permeability, SPE paper 28237 presented at the SPE Petroleum Computer Conference held in Dallas, USA., 31 July-3 August, 1994.

[22] Wong, P. M., Henderson, D. H. and Brooks, L. J., Permeability Determination using Neural Networks in the Ravva Field, Offshore India, SPE Reservoir Evaluation and Engineering 1, 99-104, 1998.

[23] Nikraves, M., A. R. Kovscek and T. W. Patzek. Prediction of Formation Damage During Fluid Injection in to Fractured Low Permeability Reservoirs Via Neural Networks. Paper SPE 31103, in the Proceedings of the SPE Formation Damage Symposium, Lafayette, Louisiana, February 14-15, 1996.

- [23] J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA. 1993.
- [24] Y-S. Lee and S-J. Yen. Neural-Based Approaches for Improving the Accuracy of Decision Trees. In Proceedings of the Data Warehousing and Knowledge Discovery Conference, pages 114-123, 2002.
- [25] Wikipedia online encyclopedia. <http://en.wikipedia.org/wiki/Prediction>
- [26] Pang-Ning Tan, Michael Steinbach and Vipin Kumar, Introduction to Data Mining. Addison Wesley, May 2, 2005.
- [27] JiaWei Han and Micheline Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers, 2001.
- [28] Fayyad, U.M., Djorgovski, S.G., Weir, N., From Digitized Images to Online Catalogs: Data Mining a Sky Survey, AIMag(17), No. 2, pp. 51-66, 1996.
- [29] Fayyad, U., G. Piatetsky-Shapiro, and P. Smyth, From Data Mining to Knowledge Discovery in Databases. AI Magazine, 17: p. 37-54, 1996.
- [30] M. C. Burl, U. Fayyad, P. Smyth, P. Perona and M.P. Burl, Automating the hunt for volcanoes on Venus. In Proceedings of the 1994 Computer Vision and Pattern Recognitino Conference, pages 302-309, Los Alamitos, CA, IEEE Computer Society Press, 1994.
- [31] Xu, Y., Einstein, J. R., Mural, R.J., Shah, M. and Uberbacher, E.C. An Improved System for Exon Recognition and Gene Modeling in Human DNA Sequences. Published Presentation: The Second International Conference on Intelligent Systems for Molecular Biology, Stanford University, San Francisco, CA, August 14-17, 1994.

- [32] R Guigó , S Knudsen , N Drake , T Smith. Prediction of Gene Structure. J Mol Biol. 226(1):141-57, July 1992.
- [33] E E Snyder and G D Stormo. Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks. Nucleic Acids Res, 21(3): 607–613, Feb 1993.
- [34] Dong S, Searls DB. Gene Structure Prediction by Linguistic Methods. Genomics. 23(3):540-51, Oct 1993.
- [35] V. Solovyev, S. A., and C. Lawrence. Predicting internal exons by oligonucleotide composition and discriminant analysis of splicable open reading frames. Nucl. Acids Res., 22:5156-5163, 1994.
- [36] Kulp D, Haussler D, Reese MG, Eeckman FH. A generalized hidden Markov model for the recognition of human genes in DNA. Proc Int Con Intell Syst Mol Biol, 4:134–142, 1993.
- [37] Krogh A, Mian IS, Haussler D. A Hidden Markov Model that finds genes in E. coli DNA. Nucleic Acids Res, 22(22):4768–4778, Nov 11, 1994.
- [38] Fayyad, U.M., Haussler, D. and Stolorz, Z. KDD for Science Data Analysis; Issues and Examples. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining KDD, Menlo Park, CA: AAAI Press, 1996.
- [39] Rakesh Agrawal , Hiekk Mannila , Ramakrishnan Srikant , Hannu Toivonen , A. Inkeri Verkamo, Fast discovery of association rules, Advances in knowledge discovery and data mining, American Association for Artificial Intelligence, Menlo Park, CA, 1996.

- [40] Hall, J., Mani, G., and Barr, D. Applying Computational Intelligence to the Investment Process. In Proc. of CIFER: Computational Intelligence in Financial Engineering. Piscataway, NJ: IEEE Press, 1996.
- [41] Leader-Post, Dec 2006. <http://www.canada.com/reginaleaderpost/index.html>
- [42] Chiu-Che Tseng and Yu-Chieh Lin. Financial Computational Intelligence. Computing in Economics and Finance 2005 from Society for Computational Economics, No 42, 2005.
- [43] Manago, M., and Auriol, M. Mining for OR. ORMS Today (Special Issue on Data Mining), Feb 1996.
- [44] Bhandari I.; Colet E.; Parker J.; Pines Z.; Pratap R.; Ramanujam K Advanced Scout: Data Mining and Knowledge Discovery in NBA Data, Data Mining and Knowledge Discovery, Volume 1, Number 1, , pp. 121-125(5), 1997.
- [45] Hatonen, K. Klemettinen, M. Mannila, H. Ronkainen, P. Toivonen, H. Knowledge discovery from Telecommunication Network Alarm Databases, Data Engineering, Proceedings of the Twelfth International Conference, 1996.
- [46] Wikipedia online encyclopedia. <http://en.wikipedia.org/wiki/Cross-validation>
- [47] Perez, H., Datta-Gupta, A. and Misra, S., The Role of Electrofacies, Lithofacies and Hydraulic Flow Units in Permeability Predictions from Well Logs: A Comparative Analysis Using Classification Trees. SPE Reservoir Engineering and Evaluation, 8 (2), April 2005.
- [48] Jensen, T. B. Estimation of Production Forecast Uncertainty for a Mature Production License, SPE Annual Technical Conference and Exhibitions, New Orleans, USA, SPE 49091, September, 1998.

- [49] I. S. Agbon, G. J. Aldana and J. C. Araque. Fuzzy Ranking of Gas Exploitation Opportunities in Mature Oil Fields in Eastern Venezuela. SPE paper 84337 presented at SPE Annual Technical Conference and Exhibition held in Denver, Colorado, USA, 5-8 October 2003.
- [50] Jia Zheng, Yusong Liu, Jiyong Li, Daisen Zhang, Dehong Chen and Hongyun shao. Application of Permeability In Profile Modification And Water Shutoff Using Genetic Algorithms. SPE paper 77883 presented at SPE Asia Pacific Oil and Gas Conference and Exhition held in Melbourne, Australia, 8-10 October 2002.
- [51] A New Genetic Algorithm (GA) Model Predicts Green-House Gas (GHG) and Reservoir Oil Interaction Parameters More Accurately in Geosequestration Applications. SPE paper 100808 presented at SPE Asia Pacific Oil & Gas Conference held in Adelaide, Australia, 11-13 September 2006.
- [52] DaeEun Kim, Jaeho Lee: Handling Continuous-Valued Attributes in Decision Tree with Neural Network Modelling. ECML : 211-219, 2000.
- [53] Ming-Syan Chen. On the Evaluation of Using Multiple Attributes for Mining Classification Rules, Tools with Artificial Intelligence. Proceedings. Tenth IEEE International Conference on Nov 1998.
- [54] Technical Note: Using Model Tree for Classification. Machine Learning, Volume 32, Number 1 / July, 1998.
- [55] C.L. Blake, and C.J. Merz. UCI Repository of Machine Learning Databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
University of California, Department of Information and Computer Science, Irvine, CA.
- [56] D.R.Skinner. Introduction to Petroleum Production. Gulf Pub Co, June 1981.

- [57] Nguyen H.H. A Study of Neural Networks and Multiple Neural Networks in Making Short-term and Long-term Time-series Prediction of Petroleum Production and Gas Consumption. Master's Thesis, University of Regina, 2004.
- [58] Definition of Water Saturation. Available at <http://www.glossary.oilfield.slb.com/Display.cfm?Term=water%20saturation>
- [59] Weka tool kit. Available at http://www.cs.waikato.ac.nz/ml/weka/index_home.html
- [60] Sigma Plot. <http://www.systat.com/products/sigmaplot/>
- [61] Genetic Algorithms. Available at <http://www.comp.lancs.ac.uk/~dixa/teaching/AI355/pdfs/CSc355-GAs.pdf>
- [62] Savlvatore Ruggieri. Efficient C4.5. IEEE Transactions On Knowledge And Data Engineering, Vcol, 14, No.2. March/April 2002.
- [63] I.H. Witten, and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques with JavaImplementations. Morgan Kaufmann, 2st Edition. 2005.

Appendix

The Source Code of the Implementation of the NDT Pre-processor

- DataSplitter.java: this class implements for splitting the data into numeric subset and categorical subset (attached).
- Adjuster.java: this class implements for handling the numeric data (attached).
- Merger.java: this class implements for handling the categorical data (attached).
- CsvFilter.java: this class filters the non csv format file (attached).
- CsvChooser.java: this class choose the csv format file (attached).
- DataController.java: this class controls the data processing (attached).
- DataPreprocessor: this class provides the user interface for user (attached).

DataSplitter.java

```
package preprocessing;

import java.io.File;
import java.io.IOException;
import java.util.Enumeration;
import java.util.HashMap;

import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;
import weka.core.converters.CSVLoader;

/**
 *
 * @author Xiongmin Li
 *
 * Energy Informatics Laboratory
 * Faculty of Engineering
 * University of Regina
 * http://www.uregina.ca/engg
 */

/**
 * This class implements for splitting the data into numeric subset and categorical
 * subset.
 */
public class DataSplitter {

    public DataSplitter() {
    }

    public Instances getnominalData(Instances dataSet) {

        final FastVector attributes = getNominalAttributes(dataSet);
        final HashMap map = new HashMap();
        for (int i = 0; i < attributes.size(); i++) {
            Attribute attr = (Attribute) attributes.elementAt(i);
            map.put(i, attr.index());
        }

        Instances nominalDataSet = new Instances("nominalData", attributes,
            dataSet.numInstances());
    }
}
```

```

Instance[] instance = new Instance[dataSet.numInstances()];

for (int i = 0; i < dataSet.numInstances(); i++) {

    instance[i] = new Instance(nominalDataSet.numAttributes());
    instance[i].setDataset(nominalDataSet);
    for (int j = 0; j < nominalDataSet.numAttributes(); j++) {
        int index = (Integer) (map.get(j));
        double value = dataSet.instance(i).value(index);
        instance[i].setValue(j, value);
    }
    nominalDataSet.add(instance[i]);
    //System.out.println(instance[i].toString());
}

int k = nominalDataSet.numAttributes();
nominalDataSet.setClassIndex(k-1);
return nominalDataSet;
}

public Instances getnumericData(Instances dataSet) {
    final FastVector attributes = getNumericAttributes(dataSet);
    final HashMap map = new HashMap();
    for (int i = 0; i < attributes.size(); i++) {
        Attribute attr = (Attribute) attributes.elementAt(i);
        map.put(i, attr.index());
    }

    Instances numericDataSet = new Instances("numericData", attributes,
        dataSet.numInstances());
    Instance[] instance = new Instance[dataSet.numInstances()];

    for (int i = 0; i < dataSet.numInstances(); i++) {
        instance[i] = new Instance(numericDataSet.numAttributes());
        instance[i].setDataset(numericDataSet);
        for (int j = 0; j < numericDataSet.numAttributes(); j++) {
            int index = (Integer) (map.get(j));
            double value = dataSet.instance(i).value(index);
            instance[i].setValue(j, value);
        }
        numericDataSet.add(instance[i]);
        //System.out.println(instance[i].toString());
    }
    int k = numericDataSet.numAttributes();
    numericDataSet.setClassIndex(k-1);
}

```

```

        return numericDataSet;
    }

    public FastVector getNominalAttributes(Instances dataSet) {
        FastVector b = new FastVector();
        for (int i = 0; i < dataSet.numAttributes(); i++) {
            if (dataSet.attribute(i).isNominal()) {
                b.addElement(dataSet.attribute(i).copy());
            }
        }
        return b;
    }

    public FastVector getNumericAttributes(Instances dataSet) {
        FastVector b = new FastVector();

        for (int i = 0; i < dataSet.numAttributes(); i++) {
            if (dataSet.attribute(i).isNumeric()) {
                b.addElement(dataSet.attribute(i).copy());
            }
        }
        int i = dataSet.numAttributes();
        b.addElement(dataSet.attribute(i - 1).copy());
        return b;
    }

    public static Instances encodeNominalAttributes(Instances dataSet) {

        Attribute[] attrs = new Attribute[dataSet.numAttributes()];
        Enumeration e = dataSet.enumerateAttributes();

        FastVector attributes = new FastVector(dataSet.numAttributes());
        int r = 0;
        do{
            Attribute att = (Attribute)e.nextElement();
            attrs[r] = new Attribute(att.name());
            attributes.addElement(attrs[r]);
            r++;
        } while(e.hasMoreElements());

        attributes.addElement(dataSet.classAttribute());

        Instances encodedDataSet = new Instances("encodedDataSet",
            attributes, dataSet.numInstances());
    }

```

```

Instance [] instance = new Instance[dataSet.numInstances()];

    for (int k = 0; k < dataSet.numInstances(); k++) {

        instance [k] = new Instance(dataSet.numAttributes());
        instance [k].setDataset(encodedDataSet);
        for (int c = 0; c < dataSet.numAttributes(); c++){
            String[] values = new
String[dataSet.attribute(c).numValues()];
            Enumeration attValues =
dataSet.attribute(c).enumerateValues();
            int j = 0;
            while (attValues.hasMoreElements()) {
                values[j] = (String) attValues.nextElement();
                j++;
            }
            for (int n=0; n<values.length; n++){
                if
(dataSet.instance(k).stringValue(c).equals(values[n])){

                    double value =
dataSet.instance(k).value(c);

                    instance[k].setValue(c, value);

                    //System.out.println(instance[k].attribute(c).isNumeric());
                }
            }
            encodedDataSet.add(instance [k]);
            System.out.println(instance[k].toString());
        }

        encodedDataSet.setClass(dataSet.classAttribute());
        return encodedDataSet;
    }

public static Instances normalizeNumericSAttributes(Instances dataSet) {
    double mins;
    double rangs;
    String[] values = null;
    for (int i = 0; i < dataSet.numAttributes() - 1; i++) {
        mins = dataSet.attribute(i).getLowerNumericBound();
        rangs = dataSet.attribute(i).getUpperNumericBound() - mins;
        for (int j = 0; j < dataSet.numInstances(); j++) {
            double c = (dataSet.instance(j).value(i) - mins) / (rangs);

```

```

        dataSet.instance(j).setValue(i, c);
    }
}

Enumeration attValues = dataSet.classAttribute().enumerateValues();
for (int j = 0; attValues.hasMoreElements() == true; j++) {
    values[j] = (String) attValues.nextElement();
    for (int k = 0; k < dataSet.numInstances(); k++) {
        if (values[j].equals(dataSet.instance(k).toString(
            dataSet.classIndex())) {
            dataSet.instance(k).setValue(dataSet.classIndex(),
j);
        }
    }
}

return dataSet;
}

public File creatFileFromCsv(File creatFile, String csvPathName)
{
    File inputfile = new File(csvPathName);
    CSVLoader atf = new CSVLoader();
    try {
        atf.setSource(inputfile);
        Instances a = atf.getDataSet();
        a.setClassIndex(a.numAttributes()-1);

    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return null;
}
}

```

Adjuster.java

```
package preprocessing;

import weka.core.FastVector;
import weka.core.Instance;
import weka.core.Instances;

/**
 *
 * @author Xiongmin Li
 *
 * Energy Informatics Laboratory
 * Faculty of Engineering
 * University of Regina
 * http://www.uregina.ca/engg
 */

/**
 * This class implements for handling the numeric data
 */

public class Adjuster {

    private NeroTrainer trainer;

    public Adjuster(NeroTrainer trainer) {
        super();
        this.trainer = trainer;
    }

    public Instances adjustNumericData(Instances numericDataSet) throws
    Exception{

        trainer.buildClassifier(numericDataSet);
        Instances dataSet = trainer.setClassType(numericDataSet);
        Instances newDataSet = new Instances(dataSet, dataSet.numInstances());
        FastVector attriweights = trainer.getWeights();

        for (int i=0; i<dataSet.numInstances(); i++){

            Instance newInstance = new Instance(dataSet.numAttributes());
            newInstance.setDataset(newDataSet);
        }
    }
}
```



```

        for (int j=0; j<dataSet.numAttributes()-1; j++){

            double [] weights = (double[]) attriweights.elementAt(j);
            double value=weights[0];

            for(int k=0; k<dataSet.numAttributes()-1; k++){

                value +=dataSet.instance(i).value(k)*weights[k+1];

            }
            newInstance.setValue(j,value);

        }
        newInstance.setClassValue(dataSet.instance(i).classValue());
        newDataSet.add(newInstance);

    }

    return newDataSet;

}

public void displayWeights(Instances numericDataSet)
{
    FastVector attriWeights = trainer.getWeights();
    for (int i=0; i<numericDataSet.numAttributes()-1; i++)
    {
        double [] weights = (double [])attriWeights.elementAt(i);
        System.out.println("The weights of Hidden Node " + i+ " is :");
        for (int j=0; j < numericDataSet.numAttributes(); j++)
        {
            System.out.println(weights[j]);
        }
    }
}

public static void main(String[] args) {

}

}

```

Merger.java

```
package preprocessing;

import weka.core.Attribute;
import weka.core.FastVector;
import weka.core.Instances;
import Jama.*;

/**
 *
 * @author Xiongmin Li
 *
 * Energy Informatics Laboratory
 * Faculty of Engineering
 * University of Regina
 * http://www.uregina.ca/engg
 */

/**
 * This class implements for handling the categorical data.
 *
 */

public class Merger {

    private NeroTrainer trainer;
    private Instances nominalDataSet;
    private boolean merged = false;

    public Merger(NeroTrainer trainer, Instances nominalDataSet) {

        super();
        this.trainer = trainer;
        this.nominalDataSet = nominalDataSet;
    }

    public Merger(NeroTrainer trainer)
    {
        super();
        this.trainer = trainer;
    }

    public void distanceMeasure() throws Exception{
```

```

trainer.buildClassifier(nominalDataSet);
FastVector attriWeights = trainer.getWeights();

for (int i=0; i<nominalDataSet.numAttributes()-1; i++)
{
    double [] weights = (double [])attriWeights.elementAt(i);

    for (int j=i+1; j<nominalDataSet.numAttributes()-1; j++)
    {
        double [] postweights = (double [])
attriWeights.elementAt(j);
        double result = caculater (weights, postweights);
        displayDistance(result, i, j);
    }
}

public void displayWeights() throws Exception
{
    trainer.buildClassifier(nominalDataSet);
    FastVector attriWeights = trainer.getWeights();
    for (int i=0; i<nominalDataSet.numAttributes()-1; i++)
    {
        double [] weights = (double [])attriWeights.elementAt(i);
        System.out.println("The weights of Hidden Node " + i+ " is :");
        for (int j=0; j < nominalDataSet.numAttributes()-1; j++)
        {
            System.out.println(weights[j]);
        }
    }
}

public double getMinDistance() throws Exception
{
    trainer.buildClassifier(nominalDataSet);
    FastVector attriWeights = trainer.getWeights();
    double minDistance = 10000;

    for (int i=0; i<nominalDataSet.numAttributes()-1; i++)
    {
        double [] weights = (double [])attriWeights.elementAt(i);

```

```

        for (int j=i+1; j<nominalDataSet.numAttributes()-1; j++)
        {
            double [] postweights = (double [])
attriWeights.elementAt(j);
            double result = caculater (weights, postweights);

            if (minDistance > result)
            {
                minDistance = result;
            }
        }
    }
    return minDistance;
}

public String[] getMergeAttributes()
{
    String [] mergeAttributes = null;
    FastVector attriWeights = trainer.getWeights();
    double minDistance = 10000;

    for (int i=0; i<nominalDataSet.numAttributes()-1; i++)
    {

        double [] weights = (double [])attriWeights.elementAt(i);

        for (int j=i+1; j<nominalDataSet.numAttributes()-1; j++)
        {

            double [] postweights = (double [])
attriWeights.elementAt(j);
            double result = caculater (weights, postweights);

            if (minDistance > result)
            {

                minDistance = result;
                mergeAttributes = new String[2];
                mergeAttributes[0] = nominalDataSet.attribute(i).name();
                mergeAttributes[1] = nominalDataSet.attribute(j).name();
            }
        }
    }
}

```

```

    }
    }
}
return mergeAttributes;
}

```

```

public double caculater(double[] a, double [] b){

```

```

    double temp = 0;
    for (int i=1; i<a.length; i++){
        temp += (a[i]-b[i])*(a[i]-b[i]);
    }
    double result = Math.sqrt(temp);

```

```

    return result;

```

```

}

```

```

public void displayDistance(double result, int i, int j){

```

```

    System.out.println("The distance between " +
nominalDataSet.attribute(i).name() + " and " + nominalDataSet.attribute(j).name()+ " is :
" );

```

```

    System.out.println(result);
    System.out.print("\n");

```

```

}

```

```

public Instances combineNominalDataSetOnce(String attri1, String attri2){

```

```

    FastVector attriValues = new FastVector();
    for (int i=0; i<nominalDataSet.attribute(attri1).numValues(); i++){
        String value1 = nominalDataSet.attribute(attri1).value(i);
        for (int j=0; j<nominalDataSet.attribute(attri2).numValues(); j++){
            String value2 = nominalDataSet.attribute(attri2).value(j);
            String newValue = value1+"_"+value2;

```

```

            attriValues.addElement(newValue);

```

```

        }

```

```

    }

```

```

Attribute newAttri = new Attribute(attri1+"_"+attri2, attriValues);

nominalDataSet.insertAttributeAt(newAttri,0);

for (int i=0; i<nominalDataSet.numInstances(); i++){

    String value1 =
nominalDataSet.instance(i).stringValue(nominalDataSet.attribute(attri1).index());
    String value2 =
nominalDataSet.instance(i).stringValue(nominalDataSet.attribute(attri2).index());
    String newValue = value1+"_"+value2;

    nominalDataSet.instance(i).setValue(0,newValue);
}

nominalDataSet.deleteAttributeAt(nominalDataSet.attribute(attri1).index());

nominalDataSet.deleteAttributeAt(nominalDataSet.attribute(attri2).index());

merged = true;
return nominalDataSet;
}

public Instances combineNominalDataSet(int times)
{
    int i = 0;
    while (i < times)
    {
        try
        {
            trainer.buildClassifier(nominalDataSet);
            String [] mergedAttributes = getMergeAttributes();
            nominalDataSet =
combineNominalDataSetOnce(mergedAttributes[0], mergedAttributes[1]);
            i++;
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    nominalDataSet.compactify();
    return nominalDataSet;
}

```

```
    }  
    public boolean isMerged()  
    {  
        return merged;  
    }  
}
```

CsvChooser.java

```
package preprocessing;

import javax.swing.JFileChooser;

/**
 *
 * @author Xiongmin Li
 *
 * Energy Informatics Laboratory
 * Faculty of Engineering
 * University of Regina
 * http://www.uregina.ca/engg
 */

/**
 * This class choose the csv format file.
 *
 */

public class CsvChooser extends JFileChooser
{

    /***/
    /* Members */
    /***/
    private CsvFilter m_filter = new CsvFilter ();

    /** Creates a new instance of CsvChooser */
    public CsvChooser ()
    {
        super ();

        setAcceptAllFileFilterUsed ( false );
        setFileFilter ( m_filter );
        setMultiSelectionEnabled ( false );
    }
}
```


DataController.java

```
package preprocessing;
```

```
import java.io.File;  
import java.io.IOException;
```

```
import weka.core.Instance;  
import weka.core.Instances;  
import weka.core.converters.CSVLoader;  
import weka.core.converters.CSVSaver;
```

```
/**  
 *  
 * @author Xiongmin Li  
 *  
 * Energy Informatics Laboratory  
 * Faculty of Engineering  
 * University of Regina  
 * http://www.uregina.ca/engg  
 */  
  
/**  
 * This class controls the data processing.  
  
 */
```

```
public class DataController {
```

```
    private DataSplitter dataSplitter = new DataSplitter();  
    private CSVLoader atf = new CSVLoader();  
    private NeroTrainer neuroTrainer = new NeroTrainer();  
    private Adjuster adjuster = new Adjuster(neuroTrainer);  
    private Merger merger;  
    private Instances finalDataSet;
```

```
    public void creatFileFromCsv(File creatFile, String csvPathName, boolean  
isMerged, boolean isAdjusted, int times) throws Exception
```

```
    {  
        File inputfile = new File(csvPathName);  
  
        try {  
            atf.setSource(inputfile);  
            Instances a = atf.getDataSet();  
            a.setClassIndex(a.numAttributes()-1);
```

```

        combineDataSet(a, isMerged, isAdjusted, times);
        CSVSaver csv = new CSVSaver();
        csv.setInstances(finalDataSet);
        csv.setFile(creatFile);
        csv.setDestination(creatFile);
        csv.writeBatch();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

    public void combineDataSet(Instances dataSet,boolean isMerged, boolean
isAdjusted, int times) throws Exception
    {
        if (checkForNumericAttributes(dataSet) &&
checkForStringAttributes(dataSet))
        {
            Instances numericDataSet = dataSplitter.getnumericData(dataSet);
            Instances nominalDataset = dataSplitter.getnominalData(dataSet);
            merger = new Merger(neuroTrainer, nominalDataset);

            if (isAdjusted && isMerged)
            {

                Instances adjustedNumericDataSet =
adjuster.adjustNumericData(numericDataSet);
                Instances mergedNominalDataSet =
merger.combineNominalDataSet(times);
                finalDataSet = reuniteDataSet(adjustedNumericDataSet,
mergedNominalDataSet);

            }

            else if (!isAdjusted && isMerged)
            {
                Instances mergedNominalDataSet =
merger.combineNominalDataSet(times);
                finalDataSet = reuniteDataSet(numericDataSet,
mergedNominalDataSet);
            }
            else if (isAdjusted && !isMerged)
            {
                Instances adjustedNumericDataSet =
adjuster.adjustNumericData(numericDataSet);

```

```

        finalDataSet = reuniteDataSet(adjustedNumericDataSet,
nominalDataSet);
    }
    else
    {
        finalDataSet = dataSet;
    }
}

    else if (!checkForNumericAttributes(dataSet) &&
checkForStringAttributes(dataSet))
    {
        if(isMerged)
        {
            merger = new Merger(neuroTrainer, dataSet);
            finalDataSet = merger.combineNominalDataSet(times);
        }
        else    finalDataSet = dataSet;
    }
    else if (checkForNumericAttributes(dataSet) &&
!checkForStringAttributes(dataSet))
    {
        if (isAdjusted)
        {
            finalDataSet = adjuster.adjustNumericData(dataSet);
        }
        else    finalDataSet = dataSet;
    }
}

public Instances getFinalDataSet()
{
    return finalDataSet;
}

public Instances reuniteDataSet(Instances numericDataSet, Instances
nominalDataSet)
{
    Instances reuniteDataSet =
numericDataSet.mergeInstances(numericDataSet, nominalDataSet);
    reuniteDataSet.deleteAttributeAt(numericDataSet.numAttributes()-1);
    return reuniteDataSet;
}

```

```

public boolean checkForNumericAttributes(Instances dataSet)
{
    Instance inst = dataSet.firstInstance();
    for (int i = 0; i < inst.numAttributes()-1; i++)
    {
        if (inst.attribute(i).isNumeric())
        {
            return true;
        }
    }
    return false;
}

public boolean checkForStringAttributes(Instances dataSet)
{
    Instance inst = dataSet.firstInstance();

    for (int i = 0; i < inst.numAttributes()-1; i++)
    {
        if (inst.attribute(i).isNominal())
        {
            return true;
        }
    }
    return false;
}

/**
 * @param args
 * @throws Exception
 */
public static void main(String[] args) throws Exception {
    DataController dataController = new DataController();
    File outputfile = new File("D:\\12.csv");
    dataController.creatFileFromCsv(outputfile, "D:\\sample dataset.csv",true,
true, 2 );
}
}

```

DataPreprocessor.java

```
package preprocessing;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.File;
import java.io.IOException;

import javax.swing.JFileChooser;
import javax.swing.JOptionPane;
import javax.swing.UIManager;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

import weka.core.Instances;
import weka.core.converters.CSVLoader;

/**
 *
 * @author Xiongmin Li
 *
 * Energy Informatics Laboratory
 * Faculty of Engineering
 * University of Regina
 * http://www.uregina.ca/engg
 */

/**
 * This class provides the user interface for user.
 */

public class DataPreprocessor extends javax.swing.JFrame implements
DocumentListener {

    private final int WIDTH = 600;
    private final int HEIGHT = 400;
    private CsvChooser m_csvChooser = new CsvChooser();

    /** Creates new form NewJFrame */
    public DataPreprocessor()
```

```

{
    super(" NDT Data Preprocessor");
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    m_csvFileLabel = new javax.swing.JLabel();
    m_csvTextField = new javax.swing.JTextField();
    m_csvBrowseButton = new javax.swing.JButton();
    m_mergeCheckBox = new javax.swing.JCheckBox();
    m_adjustCheckBox = new javax.swing.JCheckBox();
    m_saveButton = new javax.swing.JButton();
    m_cancelButton = new javax.swing.JButton();
    m_mergeNumbersLabel = new javax.swing.JLabel();
    m_mergeNumTextField = new javax.swing.JTextField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    m_csvFileLabel.setText("Choose csv data file: ");

    m_csvTextField.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            csvTextFieldActionPerformed(evt);
        }
    });

    m_csvBrowseButton.setText("Browse");
    m_csvBrowseButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            csvBrowseButtonActionPerformed(evt);
        }
    });

    m_mergeCheckBox.setText(" Merge categorical data?");
    m_mergeCheckBox.setBorder(javax.swing.BorderFactory.createEmptyBorder(0, 0,
0, 0));
    m_mergeCheckBox.setMargin(new java.awt.Insets(0, 0, 0, 0));

    m_adjustCheckBox.setText(" Adjust numerical data?");

```



```

        .add(m_csvTextField,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 126,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)))
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
        .add(m_mergeNumTextField)
        .add(m_csvBrowseButton,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
        .addContainerGap(46, Short.MAX_VALUE))
        .add(layout.createSequentialGroup()
        .add(107, 107, 107)
        .add(m_saveButton)
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 193,
Short.MAX_VALUE)
        .add(m_cancelButton)
        .add(89, 89, 89))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
        .add(layout.createSequentialGroup()
        .add(66, 66, 66)
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
        .add(m_csvFileLabel)
        .add(m_csvTextField,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(m_csvBrowseButton))
        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 72,
Short.MAX_VALUE)
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
        .add(m_mergeCheckBox)
        .add(m_mergeNumbersLabel)
        .add(m_mergeNumTextField,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
        .add(49, 49, 49)
        .add(m_adjustCheckBox,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 15,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
        .add(49, 49, 49)
        .add(layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
        .add(m_saveButton)

```



```

        .add(m_cancelButton))
        .add(45, 45, 45))
    );
    pack();
} // </editor-fold>

private void csvTextFieldActionPerformed(java.awt.event.ActionEvent evt)
{
    System.out.print("run?");
    checkText();
}

private void saveButtonActionPerformed(java.awt.event.ActionEvent evt) throws
Exception {
    String text = m_csvTextField.getText();
    if ( text.length() > 0 && !text.startsWith(" "))
    {
        m_saveButton.setEnabled(true);
        saveAs();
        setVisible(false);
        dispose();
        JOptionPane.showMessageDialog(null, "The file has been saved!");
    }

    else JOptionPane.showMessageDialog(null, "Please provide the input
file!", "Alert", 0);

}

private void csvBrowseButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if (CsvChooser.APPROVE_OPTION == m_csvChooser.showOpenDialog(null))
    {
        m_csvTextField.setText(m_csvChooser.getSelectedFile().getPath());
    }
}

private void m_cancelButtonActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    dispose();
}

```

```

public void setVisible (boolean isVisible)
{
    Toolkit tk = Toolkit.getDefaultToolkit ();
    Dimension dim = tk.getScreenSize ();

    if (isVisible)
    {
        this. setLocation ((dim.width - WIDTH) / 2, (dim.height - HEIGHT) / 2);
        this. setSize (WIDTH, HEIGHT);
    }

    super.setVisible (isVisible);
}

private boolean saveAs() throws Exception
{
    if (CsvChooser.APPROVE_OPTION == m_csvChooser.showSaveDialog (this))
    {
        File saveFile = m_csvChooser.getSelectedFile();
        if (!saveFile.exists ())
        {
            saveFile(saveFile,
m_csvTextField.getText(),m_mergeCheckBox.isSelected(),
m_adjustCheckBox.isSelected(), Integer.parseInt(m_mergeNumTextField.getText()));
        }
        else
        {
            if(JOptionPane.showConfirmDialog (m_csvChooser, "Overwrite " +
saveFile.getName () + "?",
"Overwrite Confirmation", JOptionPane.YES_NO_OPTION) ==
JOptionPane.YES_OPTION)
            {
                saveFile (saveFile,
m_csvTextField.getText(),m_mergeCheckBox.isSelected(),
m_adjustCheckBox.isSelected(), Integer.parseInt(m_mergeNumTextField.getText()));
            }
            else
                return saveAs();
        }
        return true;
    }
    return false;
}

```

```

    }

    private void saveFile(File saveFile, String csvFilePathName, boolean isMerged,
        boolean isAjusted, int times) throws Exception
    {
        DataController dataController = new DataController();
        dataController.creatFileFromCsv(saveFile, csvFilePathName, isMerged,
            isAjusted, times);
    }

    public void checkText()
    {
        System.out.println("run?");
        if (m_csvTextField.getText().length() > 0)
        {
            m_saveButton.setEnabled(true);
        }
        else m_saveButton.setEnabled(false);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        // Set the look and feel.
        try
        {
            String lookAndFeel = UIManager.getSystemLookAndFeelClassName ();

            // GTK on Ubuntu uses transparency effects, which Java 5 is not capable of.
            // Java 6 and above is capable of the GTK effects.

            String javaVersion = System.getProperty ("java.version");
            String subJavaVersion = javaVersion.substring (0, 3);
            float fSubJavaVersion = Float.parseFloat (subJavaVersion);

            if (fSubJavaVersion <= 1.5 && lookAndFeel.equals
                ("com.sun.java.swing.plaf.gtk.GTKLookAndFeel"))
            {
                // Old version of Java, so use the metal look and feel.
                lookAndFeel = "javax.swing.plaf.metal.MetalLookAndFeel";
            }
        }
    }

```

```

        UIManager.setLookAndFeel (lookAndFeel);

    }
    catch (Exception e)
    {
        e.printStackTrace ();
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new DataPreprocessor().setVisible(true);
        }
    });
}

public void insertUpdate(DocumentEvent e) {
    checkText();
}

public void removeUpdate(DocumentEvent e) {
    checkText();
}

public void changedUpdate(DocumentEvent e) {
    checkText();
}

// Variables declaration - do not modify
private javax.swing.JCheckBox m_adjustCheckBox;
private javax.swing.JButton m_cancelButton;
private javax.swing.JButton m_csvBrowseButton;
private javax.swing.JLabel m_csvFileLabel;
private javax.swing.JTextField m_csvTextField;
private javax.swing.JCheckBox m_mergeCheckBox;
private javax.swing.JTextField m_mergeNumTextField;
private javax.swing.JLabel m_mergeNumbersLabel;
private javax.swing.JButton m_saveButton;

// End of variables declaration
}

```