Modeling Frameworks for Supply Chain Analytics

by

Amit Shinde

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved March 2012 by the
Graduate Supervisory Committee:

George Runger, Chair
Douglas Montgomery
Rene Villalobos
Mani Janakiram

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

Supply chains are increasingly complex as companies branch out into newer products and markets. In many cases, multiple products with moderate differences in performance and price compete for the same unit of demand. Simultaneous occurrences of multiple scenarios (competitive, disruptive, regulatory, economic, etc.), coupled with business decisions (pricing, product introduction, etc.) can drastically change demand structures within a short period of time. Furthermore, product obsolescence and cannibalization are real concerns due to short product life cycles. Analytical tools that can handle this complexity are important to quantify the impact of business scenarios/decisions on supply chain performance.

Traditional analysis methods struggle in this environment of large, complex datasets with hundreds of features becoming the norm in supply chains. We present an empirical analysis framework termed *Scenario Trees* that provides a novel representation for impulse and delayed scenario events and a direction for modeling multivariate constrained responses. Amongst potential learners, supervised learners and feature extraction strategies based on tree-based ensembles are employed to extract the most impactful scenarios and predict their outcome on metrics at different product hierarchies. These models are able to provide accurate predictions in modeling environments characterized by incomplete datasets due to product substitution, missing values, outliers, redundant features, mixed variables and nonlinear interaction effects. Graphical model summaries are generated to aid model understanding.

Models in complex environments benefit from feature selection methods that extract non-redundant feature subsets from the data. Additional model simplification can be achieved by extracting specific levels/values that contribute to variable importance. We propose and evaluate new analytical methods to address this problem of *feature value selection* and study their comparative performance using simulated datasets. We show that supply chain surveillance can be structured as a feature value selection problem.

For situations such as new product introduction, a bottom-up approach to scenario analysis is designed using an agent-based simulation and data mining framework. This simulation engine envelopes utility theory, discrete choice models and diffusion theory and acts as a test bed for enacting different business scenarios. We demonstrate the use of machine learning algorithms to analyze scenarios and generate graphical summaries to aid decision making.

DEDICATION

To my family and teachers

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. George Runger for his guidance and nurturing throughout all my years in the graduate program. I also greatly appreciate the feedback I received from Dr. Douglas Montgomery, Dr. Rene Villalobos and Dr. Janakiram at crucial stages of my research. A special thanks to Dr. Janakiram and Intel Corporation for supporting my graduate research. Finally, I would like to thank my friends and family members for being my support system

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

Supply chains are increasingly complex as companies branch out into newer products and markets to remain competitive in a global market characterized by volatile consumer and market behavior. Operations these days require more suppliers, more parts, more customers to cater to and subsequently more returns to manage. This in turn has added layers of structural, process and cultural complexities which can potentially mask problems in the supply chain. At the same time, competitive pressure coupled with the revolution of digital and process technology have significantly reduced product life cycles. In many markets multiple products with moderate differences in performance and price often compete for the same unit of demand. As a consequence, business decisions pertaining to a specific product may have an overreaching and counter productive effect on other internally competing products. The simultaneous occurrence of competitive (pricing, product introduction etc.), disruptive (severe weather events, blackouts etc.), economic (recession, unemployment etc.) scenarios further complicates the decision domain by adding uncertainties. Effective supply chain management (SCM) has therefore become a valuable dimension for gaining an advantage over competition. Given the sheer dimensionality of the problem, improving supply chain performance is in itself a complex task, one which cannot be achieved using conventional managerial wisdom. Consequently, supply chain analytics (SCA) is getting more prominence in decision support system.

Over the last decade, advancement in tracking and measuring systems have enabled the collection of vast amounts of data associated with supply chain network. Radio-frequency identification (RFID) and GPS based systems can provide accurate data regarding the exact location of each shipment thoughout the entire

supply chain. Sales history can be traced right down to the customer touch point through point-of-sales systems. Shipment specific data such as its product identification number (SKU number), weight, volume and other related characteristics can be easily traced and recorded. Enhanced tracking also enables us to measure the performance of the supply chain at individual shipment levels using metrics related to perfect order, order fulfilment lead time, cycle time, inventory level, supply chain costs etc. Additionally, network related attributes such as route capacity, as well as global and local risk factors (power outages, extreme weather events etc.) can often be layered on top of shipment related data sources. This massive data collection effort is usually undertaken with the foresight of potentially using the information hidden in it to drive supply chain network improvements. With such information rich data sources available, managers are now relying on analytical solutions that can assimilate the copious amounts of supply chain data, identify patterns in it and quantify the impact of business decisions and/or business events (such as economic, political, regulatory, competitor scenarios, amongst other) on supply chain performance.

However, non-traditional characteristics of the resultant data sets make it necessary to choose analytical models that can successfully handle the following challenges:

- *High dimensionality:* Real-world networks can involve hundreds of nodes and can carry millions of shipments per day. Dozens of variables (attributes) per shipment or node are common. Combined together, this leads to high-dimensional data sets and hence model over fitting is a strong concern. Methods are needed that enable the relevance of variables to be statistically quantified, thereby aiding model building.

- *Disparate data types:* Scenario events and business decisions are discrete in nature and are usually represented using categorical variables. These, along with other discrete attributes related to the product such as life cycle, categorization, features etc. result in data sets that have a large number of categorical variables in them. Additional information such as prices, sales volumes, capacities etc. may be present in a continuous data format. A scenario analysis method needs to have the ability to handle such disparate data types.

- *Delayed effects:* Business decisions have an inherent inertia built into them. There may be a delay in when the decision is taken and when its effect is seen. The analysis framework needs to account for these lagged effects.

- *Variable scales:* Dramatically different scales (units) might be present for numerical measurements. Traditional attribute standardization can collapse true relationships (structure) in the data and hence methods that are invariant to scale are needed.

- *Missing data and outliers:* Dirty data with extensive missing values should be expected when data sources are numerous. Also, measurement system and tracking errors can lead to outliers in the feature space as well as the output (response) space. To avoid extensive preprocessing efforts, methods that are intrinsically robust to missing values and outliers are needed.

- *Nonlinear relationships:* Due to the multivariate nature of the data, nonlinear interactions between shipment level variables and node variables are expected. For example, only shipments above a certain weight may have delay issues at certain nodes. Also, transient effects, such as supply chain problems that affect nodes during a certain time period are expected to exhibit themselves in the data set.

- *Variable masking:* A large fraction of the variables in the data set are expected to be highly correlated with each other. It is useful to identify important variables along with alternatives or replacements with similar information content.

Traditional modeling techniques for empirical scenario analysis are unable to handle such high dimensional, messy data sets. The aim of our research is to explore, extend and improve modeling frameworks that are capable of analyzing complex scenarios arising in supply chain environments.

In this work, we develop a new empirical scenario analysis framework, labeled as *Scenario Trees*, that is capable of extracting the most impactful scenarios from hundreds of potential covariates, and predict their outcome at different product hierarchies (price point level, product family, stock keeping unit etc.). Chapter 2 provides details on this method along with a case study as applied to product pricing.

Complex datasets often require complex nonlinear models for analysis. In such environments, it is important to build elements in the analysis framework that simplify the final model as much as possible. In Chapter 3 we develop a novel model simplification strategy called *feature value selection* (FVS) that extracts important variables as well as the specific levels/values that have a significant impact on the response.

Chapter 4 illustrates an application of the FVS solution towards high dimensional supply chain surveillance. We demonstrate how FVS can be used to extract "corrupt" problematic nodes in complex network (labelled Network Feature Extraction) and create rich graphical characterizations of the interactions between covariates (such as time, transactional attributes etc.).

The *Scenario Trees* framework is an example of a top-down modeling approach and it works well for analyzing scenarios where historical data is readily available. Alternatively, for certain scenarios such as new product introduction, very little (if any) historical data is available for modeling. For such situations, a bottom-up approach to scenario analysis is presented using a agent-based simulation model (Chapter 5). Additionally we integrate machine learning algorithms with the agent-based models to analyze scenarios and provide expressive graphical summaries for decision making.

1.1 Summary of contributions

A brief synopsis of our contributions are presented in this section.

Scenario Trees: An empirically driven scenario analysis framework

The price-demand relationship for consumer products has received significant focus over the last few decades. Econometric models that estimate the cross price elasticity of demand have been well researched, studied and adopted in a wide range of decision support tools. Since these models are essentially built on the regression framework, they are limited in capability to handle high-dimensional datasets with missing values, outliers, variable redundancy and complex non-linear relationships. The technology substitution process limits the data available for modeling. This sparseness of data, coupled with the large number of variables that potentially affect demand severely challenges these models.

In this study, we view any business decision, whether forced or planned as a scenario event. We propose a framework to analyze the impact of a scenario on the market share (or demand) of a particular price group. While the discussion is limited to that at the price group level, we would like to emphasize that the framework in fact has a multi-resolution property to it. We can use the same method to analyze scenario impacts at the resolution of price groups, product families or

individual stock keeping units (SKUs). One contribution of this work is a method to represent the scenario information embedded in high dimensional data sets and integrate it with other information sources. We define indicator variables to capture impulse scenario events. For example, an indicator variable is defined to represent the scenario where a product transitions between price points due to a drop in price. While some business decisions show immediate results, often, others show delayed effects. Consider the introduction of a new product. It is natural for the demand to start showing an uptake several days after the actual product has been launched. Other decisions such as price changes may force a more rapid response from the market. Existing literature does not provide a direction on how to model delayed effects. We develop a novel representation for delayed effects by creating lagged indicator variables. This has important practical implications since it allows us to not only identify high impact scenarios, but also provide insights into when the effect will be significant.

However, creating lag variables for both continuous and discrete variables in the model adds to the dimensionality of the input set. To counter this, we determined that feature selection is needed in these high-dimensional supply chain models. A second contribution is to identify the key scenarios and other variables in supply chains that have a high business impact. The proposed framework exploits state of the art tree-based feature selection methods to separate high impact scenarios (variables) from irrelevant ones. This is of great practical significance to decision makers as it focuses their decision domain, thereby leading to better decisions. Also, a reduced subset of features results in a simpler model thereby improving prediction accuracy and generalization.

Since our focus is on modeling market shares for multiple price groups, we need to ensure that the predictions for all price groups taken together sum upto

one. To address the problem of multivariate, constrained responses, we consider a multivariate logistic model within the tree-based framework. Decision tree-based ensemble methods have been shown to handle numerical and categorical inputs with complex, non-linear interactions. These methods have also proven to be robust to input-space outliers and missing data and are therefore well suited for our problem. Therefore, we promote the use of these methods as the core modeling engine.

Models of high fidelity such as the empirical tree-based ensemble approach necessitate a strategy to analyze and interpret the possible non-linear relationships amongst the various inputs to the model. Increased fidelity in the model is attenuated if the quantitative summaries used for decision making are not sufficiently expressive. One important task is to identify inputs that are important contributors to the model results and another task is to graphically summarize the effects of such contributors. We employ modern decision tree based feature selection methods to identify contributors and apply partial dependency plots to the scenario indicators for graphical summaries.

Feature value selection and its application to supply chain surveillance

Models such as those presented in the *Scenario Trees* framework can benefit from feature selection methods to extract compact, non-redundant feature subsets from the data. The importance of feature selection methods in complex modeling environments cannot be overstated. Not only does feature selection help in reducing the dimensionality of the data, but also improves the predictive performance of the supervised learners. However, in many situations, additional model simplification can be achieved by extracting specific levels/values that contribute to variable importance. This problems, of identifying the specific levels of covariates that contribute to its importance score can be referred to as *feature value selection* (FVS).

While feature selection has been a well researched area, there is currently no literature that addresses feature value selection. We contribute to this field by developing new analytical methods to address this problem of feature value selection. Our first approach leverages well researched feature selection methods by converting the original data set into a binary incidence matrix. A second strategy, one that does not require generation of the indicator incidence matrix was also developed. Here we integrated a measure of feature value importance within the decision tree induction algorithm. Yet another method is proposed that measures the influence of a feature value through its deletion influence on the prediction error. These three methods are evaluated and compared using simulated data sets and were found to produce promising results.

While the primary motivation for FVS is to simplify models, we show that supply chain surveillance can be structured as a feature value selection problem. A primary contribution in this area is the development of a novel approach to represent a supply chain as a series of transactions using case-event data, which allows us to collect a rich set of attributes related to the network and transactions (shipments). Furthermore, this lets us convert the surveillance problem into a supervised learning problem. Using a simulated data set, we provide illustrative cases of how feature value selection can be used to identify problem nodes in the network. Additionally, we combine FVS with partial dependency plots to create rich graphical views of interactions effects (of time and other transactional attributes) at these problem nodes

Scenario analysis of technology products with an agent-based simulation models and data mining framework

The weaknesses of classical models to simulate and predict simultaneous interactions of adaptive components in complex systems has led to a great interest in

agent-based simulation approaches. Complex adaptive systems provide another avenue for studying emergent phenomenon such as product diffusion. Agent-based simulation (ABM) is a method to model complex adaptive systems. Emergent phenomena, non-linear dynamics, and path-dependent behavior are some illustrations in which ABM is used to study and analyze systems instead of traditional modeling methods. ABMs enable us study interrelationships among autonomous agents and interactions between agents and their environments in evolutionary settings. In ABM we can show interaction of agents systematically by defining decision-makers (agents), set of interaction rules and processes of changing states.

In this research, we develop an agent-based model to simulate and study technology markets. In our agent-based model, agents represent potential adopters of a technology. The agents form an artificial society in which they are connected to and interact with other agents. The definitions, characteristics and attributes of the agents are in-line with those described in the theory of diffusion literature. To operationalize the agents in the model, we give a simple set of rules that they use for evaluating products. Each agent evaluates the set of available products based on a certain pre-defined preference structure. When the need for buying a new product arises, agents score the products using additive utility functions. The probability of an agent buying a product is therefore proportional to its utility score in comparison with the other products. In addition to simple discrete choice rules, agents are also allowed to communicate with each other, thereby influencing their neighbors to either buy or not buy a particular product.

Simulation models provide the users with an excellent platform for exploring business scenarios. Users can observe changes to the output(s) by changing a few parameters of the model in a real-time setting. This same process - that of changing parameter settings and observing their impact on the output(s) - can also

be carried out off-line. We collect vast amounts of data by systematically changing the parameter settings in a designed experimental mode and then representing the relationships between the input variables and the model outputs using supervised learning methods. In our prototype study, we used tree-based ensemble methods and partial dependency plots for summarizing the data generated using the simulation model.

Chapter 2

MARKET SHARE CHARACTERIZATION OF TECHNOLOGY PRODUCTS USING SCENARIO TREES

2.1 Introduction

The revolution of digital technology has been propelling the processor market to support the demand for high performance, low cost computers. In keeping up with Moore's law, the number of transistors on a chip keep doubling about every two years. Processing power, measured in millions of instructions per second (MIPS), has steadily risen because of increased transistor counts. The pursuit of Moore's law has resulted in considerable advances in silicon-based technology. Simultaneous advances in process technology have resulted in higher yields, thus making it possible to produce less expensive, more powerful processors. With each technological breakthrough, chip manufacturers are able to introduce newer, better processors at a faster rate. Consequently, product life cycles have been considerably shortened as companies engage in a constant strife for pushing technological benchmarks.

In this highly competitive market, multiple products with moderate differences in performance and price often compete for a unit of demand. As a consequence, business decisions pertaining to specific products can have overreaching and often counter productive effects on the other internally competing products. Consider product pricing for example. Price is used as a knob to boost sales for certain products. When a higher technology product is made available to the consumers at a lower price, it has the potential to cannibalize the market demand for the products already competing at that lower price group. New product introduction and termination decisions can have similar consequences. Since product life cycles are short, changes in demand structure for products can have serious implications on scheduling of supply, manufacturing and distribution capacity. The

occurrence of multiple business scenarios along with simultaneous price change decisions makes it very difficult to predict the their effect using simple intuition. As a result, models are required to capture the interaction effects of scenarios, thus aiding in quantifying their impact on the market share of certain price groups.

The complexities of the technology substitution and diffusion mechanism requires models to substantially extend traditional approaches. Data sets are sparse due to short life cycles, a problem further aggravated as the substitution of products at different points in time results in considerable missing data. It is important to include a large set of scenario type of information that is available in discrete (event data such as promotions, product introductions etc.) as well as continuous (point-of-sales, CPI index etc.) form into modern models. This input set of scenarios can often show strongly correlation with each other and can also form non-linear relationships with the product demand. Additionally, it is also important to reorganize the scenario information using lag variables to account for delayed effects (i.e. scenarios exhibiting an effect after a period of time). This can further add to the dimensionality of the input set. From a decision making point of view, it is important to extract, summarize and subsequently focus on the most relevant scenarios from a set of possibly thousands of model inputs. The combination of sparse data sets with missing values and large number of non-linear, correlated input variables severely restricts the regression based approaches to modeling.

In this study, we present an approach which can elegantly capture scenario information that is embedded in mixed and high dimensional data sets. We propose the use of modern supervised learners to learn the relationships in the data. Variable importance scores from a modern tree based data mining models are exploited to select a subset of relevant predictors. In addition to dimensionality reduction and feature selection, our modeling framework address the issue of delayed effects.

Finally, we present a realistic case study that demonstrates the ability of this framework to model changes in market shares for a group of products in response to business scenarios such as new product introduction, product discontinuation and price changes. The models and other tools developed here are envisioned to be a part of a recommender system that provides insights into the effects various pricing and other business scenarios play on shaping market shares of different price groups at the macro level as well as those of individual CPUs.

2.2 Existing Modeling Options

Different analytical and empirical models have been proposed to address specific business scenarios that impact product demand. Empirical models such as the Cross Price Elasticity (CPE) models have been incorporated in many tactical decision support systems [Deaton and Muellbauer (1980); Green and Alston (1990)]. CPE models provide a means to estimate the percentage change in quantity demanded for the first product that occurs in response to a percentage change in price of the second product. Given a group of products, CPE is a measure of the responsiveness of the demand of one product to a change in the price of another product.

$$CPE = \frac{\%\ Change\ in\ quantity\ demanded\ for\ product\ 1}{\%\ Change\ in\ price\ of\ product\ 2} \tag{2.1}$$

There are several traditional approaches to estimate CPE. One model known as the almost Ideal Demand System model has been widely used in econometric literature. The parameters of the model are obtained using constrained Iterated Seemingly Unrelated Regression as the estimation method.

The Bass (1969) diffusion model and its variants have been used for market analysis and demand forecasting of new products [Bass et al. (1994)]. The model assumes that potential adopters of an innovation are influenced by two means of communication - mass media and word-of-mouth. This model describes the process

of how a new product gets adopted as an interaction between users and potential users. Fisher and Pry (1972) extended this single product model to a two product framework that represents the process by which a new technology product replaces or substitutes an older one in the market.

Diffusion models have also been integrated with other learning algorithms to capture and analyze scenario information. For example, Yelland et al. (2010) used a combination of the Fisher and Pry and Dynamic Linear models [West and Harrison (1997)] to capture the diffusion process as well as time series and seasonal components of product demand. Meixell and Wu (2002)and Wu et al. (2006) proposed an approach to analyze demand scenarios in technology-driven markets where product demands are volatile, but follow a few identifiable life-cycle patterns. They demonstrated that products could be clustered by life-cycle patterns, and subsequently, within each cluster, identified leading indicator products that provided advanced indication of changes in demand trends. Using the Bass growth model and a Bayesian update structure, their proposed method provided a framework for scenario analysis by focusing on parametric changes of the demand growth model over time.

Kincaid and Darling (1963) first studied single-product dynamic pricing models [Gallego and Van Ryzin (1994); Zhao and Zheng (2000)]. They formulated a continuous-time stochastic dynamic program and developed properties of the revenue function. Gallego and Van Ryzin (1997) extended their single-product model to the multiple product case and demonstrated its application to network yield management. All of these continuous-time stochastic dynamic programming models assumed that the demand uncertainty was characterized by a Poisson process with a price-dependent intensity. While retaining the same assumption of Poisson demand, Bitran and Mondschein (1997) developed a discrete-time pricing model for a retail

setting . Monahan et al. (2004) also approached the problem using a a discrete-time model in which demand uncertainty could be characterized by a generic distribution.

2.3 Modeling Framework

Empirical models have traditionally focused on price-demand relationships. While price is an important driver of demand, it needs to be considered within a broader context of information. For example, economic, regulatory and socio-economic scenarios can affect the demand for the entire technology market. At the same time, competitors actions can certainly change the dynamics of the demand for the price group. Information on such scenarios is usually available in mixed (categorical and continuous) data sources. As the list of potential impact factors keeps growing, the technology substitution process and the short product life cycles keeps shrinking the data available for modeling. Traditional models such as the CPE model have limited capability in handling high-dimensional, mixed data sets with missing values. More flexible models are required that allow us to assimilate a wide range of information sources. Casting the problem within a supervised learning structure requires an alternate representation of the model inputs. We introduce one such representation in Section 2.3. Analysts are usually interested in quantifying the impact of scenarios on multiple response units (for example, different price groups). If the response of interest is market shares, we need to ensure that the predictions for each response sum upto one. In Section 2.3, we present a strategy to address multivariate, constrained responses.

Models of high fidelity necessitate a strategy to analyze and interpret the possible non-linear relationships amongst the various inputs to the model. An important requirement from the analytical model is to identify inputs that are important contributors to the model results. We employ modern decision tree based feature

selection methods to identify contributors. Such methods have been shown to handle numerical and categorical inputs, complex, interactive, and non-linear models, as well as provide robustness to input-space outliers [Tuv et al. (2009)]. Increased fidelity in the model is attenuated if the quantitative summaries used for decision making are not sufficiently expressive. Consequently, another focus of this work is to use modern methods to summarize models outputs in order to quantitatively evaluate the effects of scenario decisions. We accomplish this by using an array of dependency plots [Friedman (2001)].

Representation of Model Inputs

Let $D^j(t)$ be the total demand for all products belonging to a price group j at time t (j = 1,...,K). In order to build a supervised learning model that relates scenario events to price group demand, we need to create a framework to represent the scenario events. We denote a scenario event of type i occurring in price group j at time t as $x^{ij}(t)$.

For each scenario, we define an indicator function

$$\begin{aligned} x^{ij}(t) &= 1, \text{if scenario } i \text{ occurs in PG } j \text{ at time } t \\ &= 0, \text{otherwise.} \end{aligned} \tag{2.2}$$

The above representation is sufficient to capture impulse events i.e. events that affect the demand for the price group only in the period in which they occur. However, many events exhibit delayed effects. As an example of such an effect, consider the introduction of a new product in a price group. Depending on market conditions and other business decisions, this new product may not show an immediate uptake in sales. Also, many such events affect demand over a longer, sustained

period. To account for these unique characteristics of the problems, we modify equation 2.2 by introducing a time window of length $(t-m)$.

$$\begin{aligned} x_m^{ij}(t) &= 1, \text{if scenario } i \text{ in PG } j \text{ occurs in time } t-m \\ &= 0, \text{ otherwise.} \end{aligned} \tag{2.3}$$

Supervised Learning

When dealing with multiple response (product or price group demands in our case), it is common to build a separate model for each response variable. However, in modeling market shares (MS), we need to ensure that the sum of all price groups taken together sum to 100%. To impose this constraint in our modeling framework, we use the following approach. For a total of K price groups, we build $K-1$ models of the following form

$$\begin{aligned} ln(\frac{MS_{PG_1}|X=x}{MS_{PG_K}|X=x}) &= Model_1 \\ ln(\frac{MS_{PG_2}|X=x}{MS_{PG_K}|X=x}) &= Model_2 \\ &\vdots \\ &\vdots \\ ln(\frac{MS_{PG_{K-1}}|X=x}{MS_{PG_K}|X=x}) &= Model_{K-1} \end{aligned} \tag{2.4}$$

The choice of the denominator price group is arbitrary. By using a simple transformation, it is evident that the sum of predicted market shares for all price groups is 100%.

$$\begin{aligned}(MS_{PG_i}|X=x) &= \frac{1}{1+\sum_i^{K-1} exp(Model_i)}, \text{if } i=K \\ &= \frac{exp(Model_i)}{1+\sum_i^{K-1} exp(Model_i)}, \text{otherwise}\end{aligned} \tag{2.5}$$

Notice that this framework is similar in form to the one used in multi-class logistic regression modeling [Hosmer and Lemeshow (2000)]. This setup is extremely flexible as it lets the user choose or define the actual nature of the model. For example, in a logistic regression model, $Model_i$ is a linear regression model of the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p \tag{2.6}$$

where y is the response variable (dependent variable), and x_i are the regressor (independent) variables (for i = *0,1, ..., p*). Depending on the expected form of the relationship between the input variables and the dependent variable, this model can be substituted with other models that are more appropriate to the modeling situation. For example, in our case, we expect the input variables to be in the form of categorical as well as continuous variables. Also, with the introduction of the lag variables, the dimensions of the data set can get very large compared to the number of data points available for modeling. Hence, for this situation, we chose to use random forest (RF) [Breiman (2001)] as the model.

Random Forests is a method used for predictive modeling. A single decision trees partition rows of data successfully based on the predictor variables to achieve a consistent response value in each partition. RF combines the predictions made by

multiple, fully grown decision trees [Breiman et al. (1984)]. Each decision tree in the forest depends on the values of a random vector sampled independently and with the same distribution. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes. In a random forest each tree is grown as follows: Let N and P represent the number of cases and number of predictors, respectively, in the training data set. A random sample of size N is drawn with replacement from the original data. This is also referred to as bootstrap sample and forms the training set for growing the tree. Each decision tree is built on a separate bootstrap sample. The cases not selected in the bootstrap sample are referred to as out of bag (OOB) data. Bootstrap samples lead to correlation between trees, which, in turn, inflates the variance of the RF model. To compensate for this, RF injects additional randomness in the model-building process by randomly selecting from a smaller subset ($p < P$) of input variables at each partition, in each tree. Each tree partitions the data to a maximum depth, but the prediction is smoothed by averaging over all trees. The OOB samples can serve as a test set for the tree grown on the non-OOB data. This can be used to get an unbiased estimate of the test set classification error.

Feature Extraction and Interpretation

Predictive models benefit from a compact, non-redundant subset of features that improves interpretability and generalization. Not only do the subset of irrelevant predictors add to the computational complexity of the modeling process but also degrade the predictive capabilities of the models [Friedman and Meulman (2003)]. Modern data mining learners such as Support Vector Machines [Guyon et al. (2002)] and tree based ensemble methods [Tuv et al. (2009); Breiman (2001)] have proven to be very successful in filtering out irrelevant predictors from the input set to gen-

erate a compact subset of non-redundant features. In addition to being effective at the feature selection, support vector machines and tree based methods are very competitive at the prediction stage.

An important aspect of a successful decision support systems is the ability to provide expressive summaries of the analytical models. Model summaries help us understand the nature of the dependence of your response on the joint values of the relevant covariates [Friedman et al. (2001)]. Here, we try to identify the range or level(s) in the important predictors that drive the most significant changes in the response. For example, we may be interested in identifying the specific region in the feature space that is associated with higher adoption for a particular product. One way to accomplish this is to create a visual graph of the predicted function $f(X)$ over the entire covariate space. This provides a comprehensive summary of the the dependency of the response on the joint values of the input variables. However, such visualization is only possible for up to four dimensional views. Friedman developed a graphical summary referred to as partial dependence plots to add interpretability to any "black box" learning methods [Friedman (2001)]. Partial dependence functions represent the effect of the variable subset on the predicted response ($f(X)$) after accounting for the average effects of the other variables. Plotting the partial dependence of $f(X)$ on its most relevant covariates can reveal how the response behaves in different regions of the covariates. We will use such plots to interpret the results from the Random Forest model.

2.4 Application

Dataset Description and Visual Analysis

To demonstrate the application of the proposed modeling framework, we simulated a data set that represents a CPU market. The data set consists of price and demand information for approximately thirty seven products across five different

price groups for a period of sixty five weeks. In this time frame, three different business scenarios were embedded in the data set. Specifically, we are interested in quantifying the effect of a new product introduction, a product discontinuation and a price drop event occurring in a target or neighbouring price groups on the market share structure of the target price group. As a result of this inherent technology substitution process, the data set contains a large number of missing rows. A snapshot of the price and demand data is shown in Figures 2.1 and 2.2 respectively.

Family	Segment	SKU	Y1W1	Y1W2	Y1W3	Y1W4	Y1W5	Y1W6	Y1W7	Y1W8	Y1W9	Y1W10	Y1W11	Y1W12	Y1W13	Y1W14	Y1W15	Y1W16
PFN01	S1	SKU1	250	250	250	250	250	250	250	250	250	250	250	250	250	250	250	250
PFN01	S1	SKU2	290	290	290	290	290	290	290	290	290	290	290	290	290	290	250	250
PFN01	S1	SKU3	330	330	330	330	330	330	330	330	330	330	330	330	330	330	330	330
PFN01	S1	SKU4	370	370	370	370	370	370	370	370	370	370	370	370	370	370	370	370
PFN01	S1	SKU5	410	410	410	410	410	410	410	410	410	410	410	410	410	410	410	410
PFN01	S1	SKU6	450	450	450	450	450	450	450	450	450	450	450	450	450	450	450	450
PFN02	S1	SKU7				290	290	290	290	290	290	290	290	290	290	290	290	290
PFN02	S1	SKU8				370	370	370	370	370	370	370	370	370	370	370	370	370
PFN03	S1	SKU9	530	530	530	530	530	530	530	530	530	530	530	530	530	530	530	530
PFN04	S1	SKU10																
PFN05	S1	SKU11				450	450	450	450	450	450	450	450	450	450	450	450	450
PFN06	S1	SKU12	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1282	1282
PFN06	S2	SKU13			1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282
PFN06	S2	SKU14	1682	1682	1682	1682	1682	1682	1682	1682	1682	1682	1682	1682	1682	1682	1442	1442
PFN06	S2	SKU15			1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442	1442
PFN06	S2	SKU16												1282	1282	1282	1282	1282
PFN06	S2	SKU17	2370	2370	2370	2370	2370	2370	2370	2370	2370	2370	2370	2370	2370	2370	1682	1682
PFN06	S2	SKU18									3970	3970	3970	3970	3970	3970	2370	2370
PFN06	S2	SKU19												1282	1282	1282	1282	1282
PFN06	S2	SKU20												1442	1442	1442	1442	1442
PFN06	S2	SKU21																
PFN06	S2	SKU22																
PFN06	S2	SKU23																
PFN06	S2	SKU24	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	1282	882	882
PFN06	S2	SKU25			1042	1042	1042	1042	1042	1042	1042	1042	1042	1042	1042	1042	1042	1042
PFN06	S2	SKU26												1042	1042	1042	1042	1042
PFN06	S2	SKU27																
PFN06	S2	SKU28																
PFN06	S2	SKU29																
PFN07	S3	SKU30																7770
PFN07	S3	SKU31																
PFN07	S3	SKU32																
PFN07	S3	SKU33																
PFN07	S3	SKU34																
PFN07	S3	SKU35									7770	7770	7770	7770	7770	7770	7770	7770
PFN07	S3	SKU36	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770	7770
PFN08	S3	SKU37								6370	6370	6370	6370	6370	6370	3970	3970	3970

Figure 2.1: Example Price data

A time based graphical summary of the data helps visualize the impact that different scenarios have on the market share for a particular price group. Figure 2.3 shows the changes in market share for PG2. Time is represented on the vertical axis while changes in market share are displayed on the horizontal axis. One week, two week and four week changes are plotted on for each week. Horizontal bands indicate the presence of a scenario. This demonstrates one of the challenges with analyzing this data set. While some of the changes in market share coincide with

Family	Segment	SKU	Y1W1	Y1W2	Y1W3	Y1W4	Y1W5	Y1W6	Y1W7	Y1W8	Y1W9	Y1W10	Y1W11	Y1W12	Y1W13	Y1W14	Y1W15	Y1W16	Y1W17
PFN01	S1	SKU1	2703525	2703525	2703525	2703525	2521775	2521775	2521775	2521775	2982425	2982425	2982425	2982425	2982425	4981925	4981925	4981925	4981925
PFN01	S1	SKU2	895625	895625	895625	895625	1744675	1744675	1744675	1744675	967875	967875	967875	967875	967875	1518225	1518225	1518225	1518225
PFN01	S1	SKU3	3691575	3691575	3691575	3691575	3499075	3499075	3499075	3499075	2259475	2259475	2259475	2259475	2259475	2468175	2468175	2468175	2468175
PFN01	S1	SKU4	1710425	1710425	1710425	1710425	1851575	1851575	1851575	1851575	2785425	2785425	2785425	2785425	2785425	1172075	1172075	1172075	1172075
PFN01	S1	SKU5	875225	875225	875225	875225	1922525	1922525	1922525	1922525	3080425	3080425	3080425	3080425	3080425	1293375	1293375	1293375	1293375
PFN01	S1	SKU6	612575	612575	612575	612575	619775	619775	619775	619775	699225	699225	699225	699225	699225	141575	141575	141575	141575
PFN02	S1	SKU7																	
PFN02	S1	SKU8					734525	734525	734525	734525	829425	829425	829425	829425	829425	2503425	2503425	2503425	2503425
PFN03	S1	SKU9					361275	361275	361275	361275	863125	863125	863125	863125	863125	100675	100675	100675	100675
PFN04	S1	SKU10																	
PFN05	S1	SKU11																	
PFN06	S1	SKU12																	
PFN06	S2	SKU13														2337725	2337725	2337725	2337725
PFN06	S2	SKU14																	
PFN06	S2	SKU15														1945375	1945375	1945375	1945375
PFN06	S2	SKU16																	
PFN06	S2	SKU17	150025	150025	150025	150025	1142525	1142525	1142525	1142525	1018675	1018675	1018675	1018675	1018675	1633725	1633725	1633725	1633725
PFN06	S2	SKU18	472325	472325	472325	472325	203425	203425	203425	203425	16075	16075	16075	16075	16075	135125	135125	135125	135125
PFN06	S2	SKU19																	
PFN06	S2	SKU20																	
PFN06	S2	SKU21																	
PFN06	S2	SKU22																	
PFN06	S2	SKU23																	
PFN06	S2	SKU24	6953775	6953775	6953775	6953775	3635925	3635925	3635925	3635925	9402575	9402575	9402575	9402575	9402575	5160025	5160025	5160025	5160025
PFN06	S2	SKU25					1860775	1860775	1860775	1860775	825275	825275	825275	825275	825275	1332825	1332825	1332825	1332825
PFN06	S2	SKU26																	
PFN06	S2	SKU27																	
PFN06	S2	SKU28																	
PFN06	S2	SKU29																	
PFN07	S3	SKU30																	
PFN07	S3	SKU31																	
PFN07	S3	SKU32																	
PFN07	S3	SKU33																	
PFN07	S3	SKU34																	
PFN07	S3	SKU35	20025	20025	20025	20025	8525	8525	8525	8525	62525	62525	62525	62525	62525	31275	31275	31275	31275
PFN07	S3	SKU36																	
PFN08	S3	SKU37	52525	52525	52525	52525	3325	3325	3325	3325	25125	25125	25125	25125	25125	31275	31275	31275	31275

Figure 2.2: Example Demand data

the occurrence of scenarios, most of the changes appear to exhibit after a lag and tend to persist for a certain time window after the scenario has occurred.

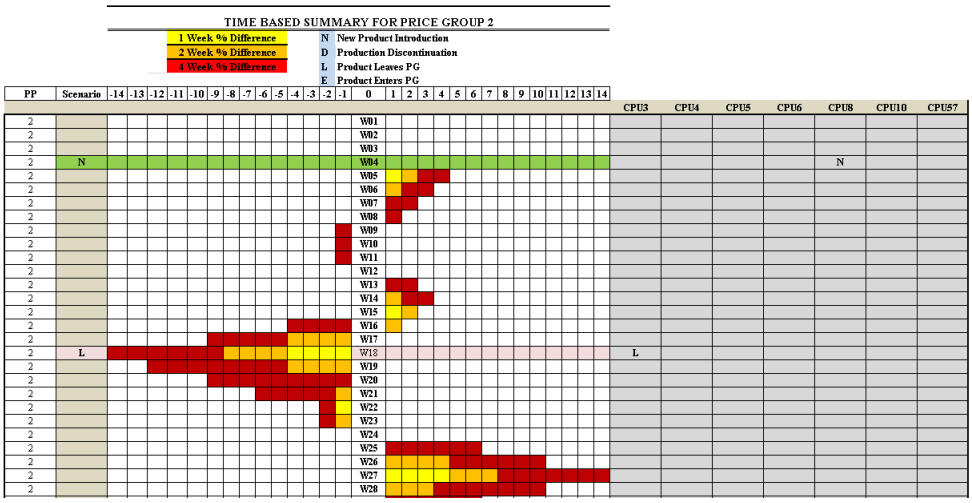

Figure 2.3: Time based summary of scenarios and market shares

Similar to the time based graphical summary, a scenario map can help track the occurrence of certain scenario events in and around a target price group. This

simple tool is a time based representation of the different scenarios that play out every week across different price groups.Figure 2.4 demonstrates the scenario map. It can be hypothesized that the impact of a particular scenario may depend largely on the popularity of the product at the time of occurrence of the the scenario. The popularity of the product can in turn be gauged by the amount of market share it captures in its current price group at the time of occurrence of the scenario. To capture this information we categorize the entering and leaving scenarios with an impact category. For example, if the product carries a market share of between $0-25\%$, we categorize it with an impact factor of 1 (Figure 2.4). To account for delayed effects as well as sustained effects, we create additional lagged features for the scenarios. In this application, we use 5 lagged variables per scenario (m = 5). These discrete lag variables behave similar to the autoregressive variables in ARIMA models.

Input variables and supervised learning

While the scenarios discussed thus far have been limited to those occurring in and around a price group, we can easily expand their scope. For example, we may consider economic, political or regulatory scenarios that have a more global affect on the business environment. At the same time, we could consider scenarios that have a more direct impact on the processor market for a particular segment (desktop, mobile or server). The current framework allows us to consider a hierarchy of input scenarios. For this current case study, we consider the gross domestic product (GDP) and the unemployment rate as an indicators of the overall economic environment and the stock price as an indicator of the company's business health. Time based effects are modelled using the month and quarter in which historic sales were realised. Thus, a rich assortment of variables can be constructed by adding more scenario variables and their indicators in this framework.

SCENARIO MAP

Week	PG1	PG2	PG3	PG4	PG5	PG6	PG7
W01							
W02							
W03						N	N
W04	N	N	N	N			
W05				L1			
W06			E1				
W07				L3			
W08			E3				
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
W56							
W57			L1				
W58		E1					
W59							
W60		D1					
W61							
W62							
W63							
W64							
W65							

LEGEND	
N	New Product Introduction
E	Product Enters PG
L	Product Leaves PG
D	Product Discontinued

IMPACT CATEGORY	
1	0 - 25%
2	25 - 50%
3	50 – 75%
4	75 – 100%

Figure 2.4: Scenario Map

Table 2.1: Error rates for RF models.

Model	*Base Error*	*Training Error*	*OOB Error*	*Error reduction*
$ln(\frac{MS-PG_2}{MS-PG_1})$	0.638	0.143	0.309	52 %
$ln(\frac{MS-PG_3}{MS-PG_1})$	0.950	0.189	0.378	60 %
$ln(\frac{MS-PG_4}{MS-PG_1})$	2.003	0.466	0.975	51 %
$ln(\frac{MS-PG_5}{MS-PG_1})$	0.906	0.248	0.528	42 %

The feature selection algorithm proposed by [Tuv et al. (2009)] was used to select a subset of input scenarios that are significantly help in predicting the market shares. The feature selection algorithm was run for all $K-1$ models and the results were summarized in a scenario impact matrix Figure 2.5. Table 2.1 summarizes the predictive capability of the models.

			Affected Model			
Scenario	Originating Price Point	Lag	ln(MS_PG2/MS_PG1)	ln(MS_PG3/MS_PG1)	ln(MS_PG4/MS_PG1)	ln(MS_PG5/MS_PG1)
Month			X	X	X	X
GDP			X	X	X	X
Stock Price			X		X	X
Unemployment rate			X			X
NPI	PG2	1				X
		2				X
		3				X
		4			X	X
EOL	PG2	1	X		X	
		2	X		X	
		3	X		X	
		4	X		X	
EOL	PG3	0				X
		1	X	X		X
		2	X	X	X	
		3	X	X	X	
		4		X	X	
NPI	PG4	1		X		
		2				
		3				X
		4				X
EOL	PG4	0		X	X	X
		1		X		X
		2				X
		3			X	X
		4		X		X
		5	X	X		X

Figure 2.5: Scenario impact matrix. ”X” indicates that the corresponding scenario is important for that model

Arranging the dependency plots for all the important scenarios and variables in a matrix such as Figure 2.6 will help us assess possible conflicting scenarios i.e. scenarios that have a positive effect on one price group and a negative impact on some other. In each dependency plot, we have highlighted the case where no scenario occurs with a box. Consider the scenario where we drop the price of a product currently in PG_4. As a consequence of this business decision, this product will leave PG_4 and enter PG_3. We can see from the dependency plots that whenever such a price drop occurs, there is a positive impact on PG_3 but a negative impact on PG_4.

2.5 Conclusion

Technology products are characterized by short life cycles with rapid obsolescence, decreasing prices, and a continuous progression of competing products both from competitors and from within the same organization. We outlined the characteristics of the technology market that make it difficult for us to model market data using traditional approaches. We demonstrated a method to shape or reorganize the sce-

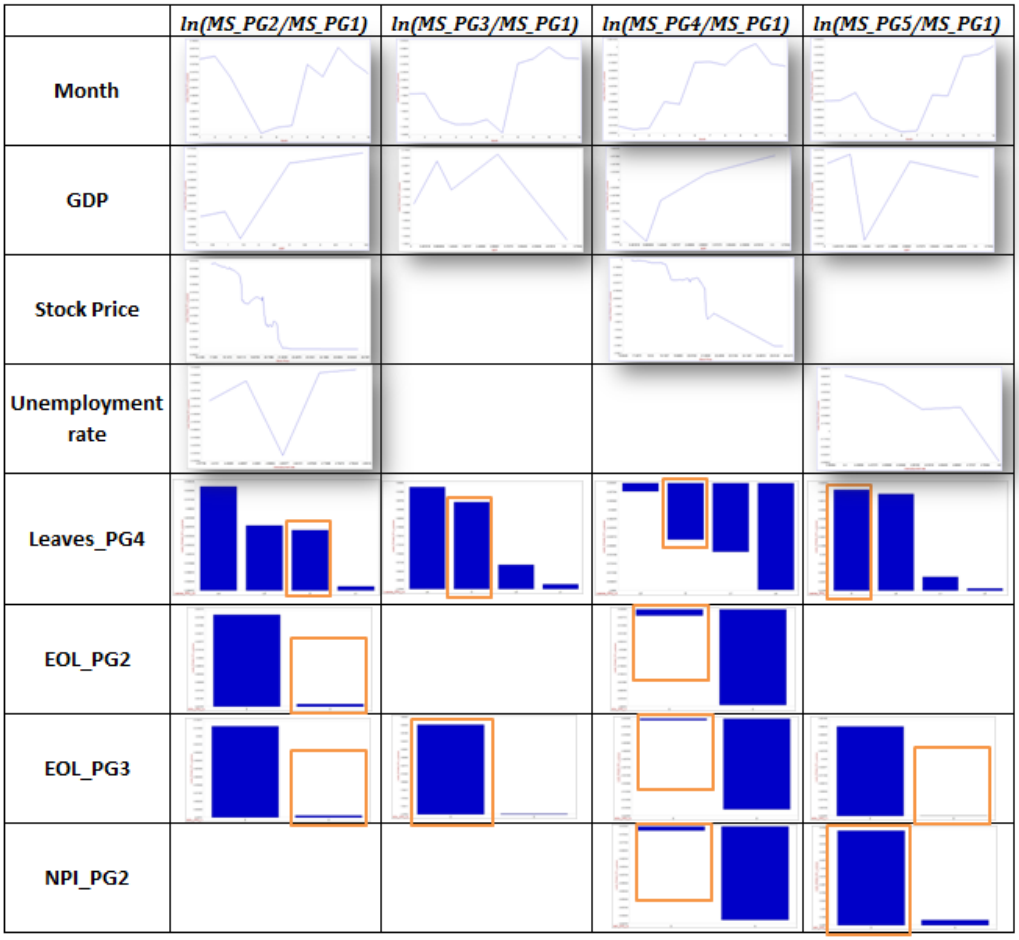

Figure 2.6: Dependency plot matrix

nario information in a manner that is amenable to use with modern data mining models. This allowed us to integrate multiple scenarios (both discrete and continuous) and accommodate for the practical issue of lagged effects in the modeling step. In markets that can potentially react to hundreds of scenarios, we provide an elegant method to extract the most relevant scenarios using variable importance scores from the tree models. Additionally, we leveraged dependency plots to characterize the nature of the impact of this reduced set of scenarios. This modeling framework is envisioned to be part of a recommender system that allows the analyst to focus and track critical scenarios and predict their outcome on more than one group of products.

Chapter 3

FEATURE VALUE SELECTION : STRATEGIES AND COMPARATIVE ANALYSIS

3.1 Introduction

In the last decade, a considerable amount of research has been focused on developing new techniques for feature selection. Good feature selection methods aim to create a compact, non-redundant subset of features that improves interpretability and generalization of models. However, in many situations, identifying important variables is not always sufficient. Often times, it is necessary to investigate each variable and identify which specific values or levels of the variable contribute to its importance. For instance, in public health surveillance, feature selection can be used to identify whether there are any disease clusters in the data. However, it is equally important to identify the specific geographical regions and/or subpopulations where these clusters may be occuring. Similarly, in complex supply chains, a model which indicates that a majority of product defects occur during the manufacturing stage is probably not going to help us solve the problem. It would be far more meaningful to localize the problem to the exact manufacturing location (node in the network). The same can be said for manufacturing processes, where it is critical to pin point which machine/tool is contributing to higher defect rates. This problems, of identifying the specific levels of covariates that contribute to its importance score can be referred to as *feature value selection* and is the focus of investigation here.

Not only is feature value selection important from a response localization point of view, it also helps reduce the complexity of the final predictive model. Take discrete Bayesian networks for instance. There are two components that add to the complexity of Bayesian networks : The number of parent variables and the number

of levels for each parent. The first problem, that of reducing the number of variables in the model is a classic subset selection problem. This can be addressed using a variety of feature selection algorithms such as the tree based ensemble methods proposed by Breiman (2001) or by Tuv et al. (2009) or the support vector based methods proposed by Guyon et al. (2002). Feature selection methods reduce the complexity of a Bayesian network by creating a compact, non-redundant subset of variables. Once this subset of important features is selected, we can further reduce the complexity of the Bayesian Network by identifying the specific levels of the features that significantly influence the response variable. Other levels that do not influence the response variable can then be grouped together in one category. This will reduce the number of parameters that need to be defined for each child node, thereby, reducing the networks complexity.

In this paper, our objective is to propose methods to address the *feature value selection* problem. We develop three different strategies to extract feature values and compare their performance using a simulation study. The first two strategies exploit traditional feature selection algorithms by using an appropriate data transformation. The third strategy extends the ensemble-based feature selection method proposed by Tuv et al. (2006, 2009) to allow for feature values to be scored.

In Section 3.2 we briefly describe some of the methods that we investigated in the simulation studies. Section 3.3 describe the three feature value selection methods and Section 3.4 describes the simulation study and performance comparison for the three methods.

3.2 Background

Chi-Square based Feature Selection

The statistical significance of the relationship between two categorical variables can be tested using the χ^2 text of independence. The test essentially finds out whether

the observed frequencies in a distribution differ significantly from the frequencies that might be expected according to the hypothesis of statistical independence between the two variables (Null hypothesis). The test statistic for testing this hypothesis is called the Chi-square statistic and is computed as -

$$\chi^2 = \sum_i \sum_j \frac{(E_{ij} - O_{ij})}{E_{ij}} \tag{3.1}$$

where, O_{ij} is the observed number of instances from Class Y_i having the j^{th} value of a given covariate (input variable), and E_{ij} is the expected number of instances if the null hypothesis (of no association between the two variables) is true.

From a feature selection point of view, we can view the value of the χ^2 as a measure of variable importance. Given that it is a statistical hypothesis test, we can also measure its statistical significance by computing its *p-value*.

Correlation-based Feature Selection

Correlation-based Feature Selection (CFS) (Hall (1999)) is a subset evaluation heuristic that takes into account the usefulness of individual features for predicting the class along with the level of inter-correlation among the subset. The underlying premise is that a good feature subset is one that contains features that have a high degree of correlation with the response variable, yet are uncorrelated with each other. The importance of merit of a feature subset S consisting of k features is given by

$$Merit_S = \frac{k\bar{r_{cf}}}{\sqrt{k + k(k-1)\bar{r_{ff}}}} \tag{3.2}$$

where $\bar{r_{cf}}$ is the average class-feature correlation, $\bar{r_{ff}}$ is the feature-feature correlation. The degree of association between two discrete features (X and Y) is estimated using the symmetrical uncertainty (SU) equation

$$SU = 2 * [\frac{H(X) + H(Y) - H(X,Y)}{H(X) + H(Y)}] \tag{3.3}$$

where $H(X)$ and $H(Y)$ is the entropy of features X and Y. The symmetrical uncertainty coefficient lies between 0 and 1. A value of 0 indicates that variables X and Y have no association between them; the value of 1 indicates that knowledge of one variable completely predicts the other. A search heuristic (hill climbing or Best First) is then applied to search the feature subset space.

The Best First strategy (Rich and Knight (1991)) starts with an empty feature set and then generates all possible single feature expansions. The subset that provides the maximum merit score is then added to the feature set. One subsequent iterations, this feature set is expanded by adding single features. If the expanding a subset does not result in an improvement in the merit score, the search drops back to the next best unexpanded subset and continues from there. To prevent the method from exploring the entire search space,it is typical to terminate the search after a certain number (say, 5) of subsets expanded result in no improvements. The best subset found is then returned when the search terminates.

Feature Selection using tree-based ensembles and artificial contrasts

Efficient mechanisms to handle mixed categorical and numerical missing-valued data make tree-based models popular for real-world applications. Ensembles of trees such as random forest (Breiman (2001)) and gradient boosted trees (Friedman (2001)) are not only accurate for supervised learning, but also provide intrinsic mechanisms to produce variable importance scores.

The random forest model constructs a set of simple decision trees, and uses their weighted outcome to predict new data. The measure of variable importance for a single decision tree (T) is given by

$$VI(X_i, T) = \sum_{n \in N_T} \triangle I(X_i, n) \tag{3.4}$$

where $\triangle I(X_i, n) = I(n) - p_L I(n_L) - p_R I(n_R)$ is the decrease in impurity because of an actual (or potential) split on variable X_i at a node n of the optimally pruned tree T. For an ensemble of M trees, Random forests simply averages the importance score in equation 4.1 over all M trees.

$$VI(X_i) = \sum_{m=1}^{M} VI(X_i, T_m)/M \tag{3.5}$$

A problem with the variables importance scores generated through tree-based methods is that there is no obvious threshold to segregate important variables from irrelevant (noise) variables. Setting this cut-off threshold based on intuition can either lead to false alarms (if set too leniently) or missed signals if the threshold is too aggressive. Tuv et al. (2006, 2009) addressed this problem by using artificially created noise variables, also referred to as contrasts to set the noise threshold in the system. Only a variable with a significant difference (as measured with a *p-value*) from the k^{th} percentile (say, 90^{th}) of artificial variables is considered to be an important feature. Therefore, this approach eliminates guess work by providing an objective measure based on *p-values* for defining significant variables and segregating them from irrelevant noisy ones.

A similar method is used to identify and remove redundant variables. A modified surrogate score, called a masking score, is computed between all pairs of variables (including contrast variables) using a gradient boosted tree (Friedman and Meulman (2003)). For a single replicate, an upper percentile (say, 75^{th}) of the masking score between the original variables and contrast variables is computed. After several replicates, a mean is calculated, which defines the threshold against which all masking scores between original variables is compared using a *t-test*. If the masking score between two original variables is statistically greater than the mean of the 75^{th} percentile, then the masking score is kept. To remove redundancy,

Table 3.1: Toy data set with categorical variables

Row	X_1	X_2	*Class*
1	A_1	B_1	0
2	A_2	B_1	1
3	A_3	B_2	1
4	A_1	B_3	1

the variable with the highest importance score is retained, and every other variable with a significant making score with it is eliminated.

Thus in summary, the ACE method generates *p-values* for : (a) identifying relevant subsets of variables that have a statistically significant impact on the response and (b) identify correlated or masked variables, thereby eliminating redundant variables.

3.3 Methodology
Indicator variables with feature selection

We are interested in identifying feature levels/values that potentially have a significant impact on the response variables. We can easily convert the feature selection problem into a feature value selection problem using a straight forward data transformation. We begin by transforming the original categorical input variables into indicator variables, one for each level of the original variable. Each covariate range and/or level in the input set becomes a column (or item) in the binary indicator/incidence matrix. Values of one in the binary incidence matrix relate to items that were included in the original row vector, while values of zero relate to the lack of items in the original row vector.A simple toy data set illustrates this procedure. The original and transformed data sets are shown in Tables 3.1 and 3.2 respectively. For example, the first row in Table 3.1 becomes $[X_1 : A_1] = 1$, $[X_2 : B_1] = 1$ and zero elsewhere in Table 3.2. The remaining rows in the incidence matrix are calculated in the same manner.

Table 3.2: Transformed toy data set with indicator variables

Row	$X_1 : A_1$	$X_1 : A_2$	$X_1 : A_3$	$X_2 : B_1$	$X_2 : B_2$	$X_2 : B_3$	*Class*
1	1	0	0	1	0	0	0
2	0	1	0	1	0	0	1
3	0	0	1	0	1	0	1
4	1	0	0	0	0	1	1

Each column of the incidence matrix corresponds to a distinct level of the original covariates. Therefore, we can use standard feature selection algorithms, such as those mentioned in 3.2, to identify important feature values.

Feature value selection using tree-based ensembles

The ACE algorithm can be easily extended to identify important feature values. To do this, we need to modify the learning algorithm to extract importance scores for individual levels within the categorical covariates. Consider a single decision tree. At each node of the tree, the splitting algorithm selects the variable (X_i) that provides the highest decrease in impurity as the variable to split the node on. Once a variable (say X_i with J_i category levels) has been chosen, we can then iterate through each category level (j, $j = 1$ to J_i) and compute its importance score, denoted as variable-value importance (VVI) as follows

$$VVI(X_{ij}, T) = \sum_{n \in N_T} Split\ weight_n * \triangle I(X_{ij}, n) \tag{3.6}$$

where: $Split\ weight_n = 1$ if the split is made on variable i, category level j at node n. Since individual category levels are evaluated sequentially, we enforce a *one-versus-rest* splitting rule at each node. This means that a split such as {a},{b,c,d} will be allowed, while, a split such as {a,b},{c,d} will not be allowed.

For an ensemble of M trees, we would simply average the importance score in equation 4.3 over all M trees.

$$VVI(X_{ij}) = \sum_{m=1}^{M} VI(X_{ij}, T_m)/M \tag{3.7}$$

Since the feature value selection algorithm is based on the ACE algorithm, it inherits all the desirable properties that tree-based ensemble methods possess - from being robust to noise and outliers, to being able to handle strong non-linear interactions. Additionally, the artificial contrasts used in the ACE algorithm allow us to test the statistical significance of each feature value using *p-values*.

Feature value selection using model prediction error

We may also investigate the influence of a feature value X_{ij} by measuring its deletion influence on the prediction (test) error. First, we split the dataset into a separate training and test set and use an appropriate supervised learner on the training set and compute the baseline prediction (test) error on the test set (TE_b). Then, we force variable level X_{ij} to missing by deleting the cells of the dataset (training and testing) that contain variable X_{ij}. Note that we only delete the cells of the data set that contain X_{ij} and not the entire row of data. We build the model again on this revised training set and again compute the test error (TE_{ij}). If variable X_{ij} is not influential in predicting the response, then the new test error TE_{ij} should be close to the baseline test error TE_b. On the other hand, if X_{ij} is an important predictor, then we should see a deterioration in the predictive performance. By this logic, we define the change in prediction error ($TE_{ij} - TE_b$) as a measure of variable value importance. For the next iteration, we again start with the original dataset and delete the next variable level. Since, we delete one variable at a time, we abbreviate this method using OFAT (One Factor and A Time).

1. Split the base dataset into a training and test set

2. Build a supervised model using the training set

3. Compute the baseline *Test Error* $(TE_b) = \frac{\textit{Number of records misclassified}}{\textit{Total no. of records}}$

4. For i = 1 to I

   For j = 1 to J_i

   Delete X_{ij} from the base dataset

   Rebuild the supervised model using this transformed dataset

   Compute the *Test Error* (TE_{ij})

   Compute the variable value importance $VVI(X_{ij}) = TE_{ij} - TE_b$

   Next j

   Next i

3.4 Simulation Study and Results

In this section, we present a detailed simulation study to demonstrate and compare the performance of the different feature value selection strategies suggested in Section 3.3. Our basic simulation is conducted using a framework of four categorical factors (factor A with 64 levels, factor B with 32 levels, C with 16 and D with 8 levels). Hence, in all there are 120 categorical levels from which to choose from. The following parameters were varied during the simulation runs to create scenarios of varying difficulty on which to test the suggested methods:

Simulation setup

- *Response model* - We wanted to generate data sets that exhibit nonlinear interaction effects between the covariates and the response variable. To do this, we used three interaction operators, *OR*, *XOR* and a combined (*OR,*

AND) to define our covariate-response relationship. Given two binary variables (X_1,X_2), the *OR* logical disjunction results in true whenever one or more of its operands (X_1, X_2) are true. The *XOR* operator on the other hand returns a value of true if exactly one of the operands has a value of true (one or the other but not both). For the *AND* response to be true, both its operands must be true. The truth tables of (X_1 *OR* X_2) and (X_1 *XOR* X_2) are shown in Table 3.3.

The combined (*OR, AND*) condition is used to create a more complex interaction between the input variable levels with the response variable and can be best explained using the toy data set in Table 3.4. Here, the *OR* function allows us to choose different levels from **within** a variable, say $\{X_1 : A_1$ *OR* $X_1 : A_2\}$ from variable X_1 and just one level $X_2 : B_2$ from variable X_2. Now, the *AND* condition can be used to create an interaction **between** the chosen variables. Hence, the final condition that defines the response variable is $\{\{X_1 : A_1$ *AND* $X_2 : B_1\}$ *OR* $\{X_1 : A_2$ *AND* $X_2 : B_1\}\}$

- *Number of active levels k* - A feature level that has a defined relationship with the response variable is deemed as being active. At each replicate of our simulation, we choose a total of k active levels out of the 120 possible variable levels. For the *OR* and *XOR* cases, the simulation setup uses three different choices for the number of active levels k: 5, 15 and 25. For the combined *OR, AND* case, defining a three or four variable interaction results in a highly unbalanced data set. Hence, we chose to draw k active levels (5 and 10) only from variables C and D. Variables A and B were still retained as inputs and acted as noise variables. The average proportion of Class 1 records in out simulations is shown in Table 3.5

Table 3.3: Truth tables for OR, XOR and AND operators

$Input_1$ X_1	$Input_2$ X_2	*OR Output* y_1	*XOR Output* y_2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Table 3.4: An example demonstrating the (OR, AND) response function

Row	$X_1 : A_1$	$X_1 : A_2$	$X_1 : A_3$	$X_2 : B_1$	$X_2 : B_2$	$X_2 : B_3$	*(OR,AND)*
1	1	0	0	1	0	0	1
2	0	1	0	1	0	0	1
3	0	0	1	0	1	0	0
4	1	0	0	0	0	1	0

- *Noise level* - To analyze the effect of noise on the performance of the feature value selection problem, we allowed the bits of the response variable to flip with varying levels of probability denoted by p_f. For the baseline case, we set this probability to zero. For the *OR* and *AND*, we set $p_f = 0.1$ for the noisy case. However, for the *OR, AND* function , at $k = 5$, the proportion of Class 1 records in the data set is as low as 4.6 %. Setting $p_f = 0.1$ therefore will result in a very low signal to noise ratio (roughly 50%). Hence, to keep noise leves proportional to signal, we set the maximum $p_f = 0.05$

Experiment Simulation

For each of 10 iterations, the following steps were carried out

1. Assign *k* active factor at random. Randomly assigning the active factors helps reduce selection bias.

2. Using the *k* active factors and the response function (*OR/XOR/ (OR,AND)*), define the response variables *y*.

Table 3.5: Summary of simulation setup and resulting data sets

Response condition	*Number of active levels k*	*Avg. proportion of Class 1*
XOR	5	0.133
	15	0.337
	25	0.436
OR	5	0.141
	15	0.397
	25	0.589
(OR, AND)	5	0.046
	10	0.163

3. After the data is generated, each row of the binary response function is allowed to flip with probability *p(f)*.

Metrics for measuring accuracy

We used the following metrics to gage the performance of the feature value selection method.

$$Sensitivity = \frac{Number\ of\ correctly\ detected\ problem\ nodes}{Total\ number\ of\ nodes} \tag{3.8}$$

$$Specificity = 1 - \frac{Number\ of\ nodes\ detected\ in\ excess}{Total\ number\ of\ nodes} \tag{3.9}$$

We expect successful methods to have large sensitivity and specificity. However, these two metrics often counteract i.e. increasing the sensitivity of a method often leads to a loss of specificity and vice-versa. Therefore, we define a derived metric that will be reported in addition to the above two metrics :

$$d = \sqrt{(1 - Specificity)^2 + (1 - Sensitivity)^2} \tag{3.10}$$

The values for sensitivity and specificity are computed for each replicate separately and are then averaged over the five replicates.

Tuning Constants

Random forest and Gradient boosted tree provide an intrinsic measure of variable importance that can be used to rank variables. However, with only variable importance scores, there is no obvious threshold to segregate important variables from a potentially large subset of covariates. The list of importance values does not come with an associated indication of which variables to include and which ones to ignore. Hence, setting the tuning constant for these methods equates to selecting a user-specified cut-off threshold for the variable importance scores. If we are too lenient in setting this threshold, we might be fitting the model to noise and thereby increase the rate of false alarms. If we are too strict with this threshold, we then run the risk of missing relevant variables. Setting the threshold value is often an ad-hoc process and the rule of thumb is to set the cut-off value between 5 % and 10 %. For our experiments, we set a 5 % threshold and results are reported at this setting. Additionally, performance curves (d v/s cut-off) are presented to better characterize the performance of these approaches over a range of cut-off values.

In comparison, the ACE method provides a sound statistical basis for picking the cut-off threshold. In addition to providing variable importance scores, the ACE method also generates *p-values* by comparing the variable importance scores of original variables and artificially generated noise (contrast) variables. The *p-value* is commonly used in statistical analysis and hence is well understood. As is typical, we set the level of significance at 0.05. Any variable whose *p-value* is less than 0.05 is retained while the others are discarded. However, note that the usual rejection rule of ($p \leq 0.05$) often leads to a higher false positive rate when multiple hypotheses are compared.

To counter this problem of multiple hypothesis testing, we use a simple Bonferroni adjustment ($p_{adj} = p/I$), where I is the number of variables) for the *p-value* threshold in the ACE method (Hochberg and Tamhane (1987)).

Each feature selection method has one or more tuning constants that balance the trade-off between model sensitivity and specificity. Since the χ^2 method of feature selection is based on a statistical hypothesis test, we can use a *p-value* approach to identify statistically significant features. We used the Bonferroni adjusted *p-value* of $0.05/I$ as the level of significance. For the CFS method, the best first search strategy was used to select feature subsets. We used the default settings and stopped the search algorithm when sequentially expanding 5 search nodes (subsets) did not yield an improvement in the merit score.

Simulation Results

Results for the XOR response simulations are shown in Table 3.6. Consider the no noise case. Here, we see that the tree-based methods (I-ACE, I-RF, I-GBT and E-ACE) perform very well. At the low (5) setting for active factors, all four methods exhibit optimal performance (sensitivity = 1 and specificity = 1). The I-χ^2 filter, which evaluates each attribute individually, yields a perfect sensitivity score with a slightly smaller but still acceptable specificity score (0.946). The CFS filter on the other hand performs poorly with a sensitivity score of only 0.66.

At the medium setting for active levels (15), all four tree-based approaches yield optimal (I-GBT, E-ACE) or near optimal performance (I-ACE, I-RF). These models still retain a perfect sensitivity score and show very little deterioration in the specificity scores, the lowest being 0.994 for I-RF. The CFS filter shows further deterioration in sensitivity (0.54). Although the I-χ^2 shows deterioration over the previous setting, it still returns respectable performance. However, at the highest

setting for active levels (25), both filters miss a lot of true positives and this can be seen in very low sensitivity scores. The tree methods on the other hand show excellent performance even at this setting with I-ACE and I-Rf still yielding perfect sensitivity scores, while E-ACE and I-GBT show slight deterioration. In general, the tree-based methods perform really well, even with high levels of active factors in the system. Although performance difference between the tree-based methods may not statistically significant, the E-ACE method ranks the best amongst its competitors, resulting in the lowest d scores in for all three settings of active factors.

When noise is added to the system ($p_f = 0.1$), at low to medium active factor levels the tree-based approaches report perfect sensitivity scores with optimal (I-GBT, E-ACE) / near optimal (I-ACE, I-RF) specificity scores. Both filters show a similar pattern to the zero noise case. The CFS method still shows poor sensitivity , while the I-χ^2 method has acceptable performance. The sensitivity measure for both filters shows deterioration at the 15 active factors setting due to noise. At the most difficult setting for the simulation (15 active levels and p_f = 0.1), the performance of both filters deteriorates considerably, with sensitivity scores of less than 46 %.The tree methods on the other hand continue to yield high sensitivity and specificity.

The OFAT approach performs quite poorly even with a gradient boosted tree model as the supervised learner. This however, is to be expected since we only evaluate one variable at a time. Since we are experimenting on the XOR function, there is a lot of variable redundancy built into the system. In other words, there are other variable levels in the data that have the same/similar information content as the one under investigation. Thus, the prediction error does not deteriorate even though we delete one of the active factors from the data.

Figures 3.1 and 3.2 show the performance curves for I-RF and I-GBT respectively for the XOR response function. The value of the cut-off threshold (shown between 0 to 0.15) is represented on the x-axis while the corresponding performance metric (d) is plotted on the y-axis. Both figures show that low cut-off values result in a high value for d (lower is better). As we increase the cut-off threshold, the value of d decreases initially, reaches a minimum and subsequently starts to increase again. This is so because, low cut-off values result in high sensitivity and low specificity. As we increase the threshold, the sensitivity drops while the specificity improves. These figures highlight a drawback of using RF or GBT or any other variable ranking method for feature selection. While thumb rules may be simple to implement, they do not always result in optimal feature subsets. Since in practise, we would never know what the true performance curve looks like, there is a real possibility of either missing an important feature or including insignificant ones by setting a wrong (sub-optimal) threshold.

Results for the OR response variable are summarized in Table 3.7. Similar to the *XOR* case, tree-based methods exhibit high sensitivity and specificity scores across all simulation settings. In fact, all tree-based approaches result in perfect performance scores when the number of active levels is held at 5 and 15, with zero noise. At the 25 active levels and zero noise setting, the I-ACE and E-ACE approaches slightly outperform the I-RF and I-GBT approaches. The I-χ^2 method yields a perfect sensitivity score at all three settings for active levels. However, this sensitivity performance comes at the price of poor specificity. The specificity score drops to as low as 0.772 at the highest active level setting. The CFS filter performs very poorly compared to any of the other approaches. A similar behavior is observed when noise is added to the system. For the low and medium active levels setting, the tree-based methods show perfect sensitivity scores whilst still

Table 3.6: Simulation results for the XOR response function

Noise	*Active Levels*	*Metrics*	*Ind ACE* / *I-ACE*	*Ind RF* / *I-RF*	*Ind GBT* / *I-GBT*	*Ind CFS* / *I-CFS*	*Ind* χ^2 / *I-*χ^2	*Embed ACE* / *E-ACE*	*Deleting OFAT GBT*
0	5	Sensitivity	1	1	1	0.66	1	1	1
0	5	Specificity	1	1	1	1	0.946	1	1
0	5	d	0	0	0	0.34	0.054	0	0
0	15	Sensitivity	1	1	1	0.54	0.973	1	0.247
0	15	Specificity	0.996	0.994	1	1	0.923	1	1
0	15	d	0.004	0.006	0	0.46	0.098	0	0.753
0	25	Sensitivity	1	1	0.956	0.488	0.572	0.992	0.08
0	25	Specificity	0.988	0.973	1	1	0.974	1	0.994
0	25	d	0.012	0.027	0.044	0.512	0.433	0.008	0.92
0.1	5	Sensitivity	1	1	1	0.66	1	1	0.98
0.1	5	Specificity	0.967	1	1	1	0.98	1	0.999
0.1	5	d	0.033	0	0	0.34	0.02	0	0.021
0.1	15	Sensitivity	1	1	1	0.533	0.927	1	0.262
0.1	15	Specificity	0.971	0.992	1	1	0.968	1	1
0.1	15	d	0.029	0.008	0	0.467	0.1	0	0.738
0.1	25	Sensitivity	0.944	1	0.988	0.432	0.452	0.96	0.084
0.1	25	Specificity	0.957	0.903	0.993	0.997	0.993	0.998	0.995
0.1	25	d	0.079	0.097	0.019	0.568	0.548	0.042	0.916

retaining high specificity. At 25 active levels, these methods continue to provide high levels of performance, with the lowest sensitivity being 0.944 for the I-RF model. Corresponding performance curves for the I-RF and I-GBT approaches are shown in Figure 3.3 and Figure 3.4 respectively.

Results for the combined (*OR,AND*) response function are presented in Table 3.8. As is evident, the I-ACE, I-GBT and E-ACE methods perform exceedingly well for the no noise case at 5 and 10 active levels respectively. The I-RF approach performs well when there are only 5 active factors in the simulation. However, at 10 active factors, this method suffers in specificity. The perfect sensitivity score of the approach is misleading since it classifies nearly all levels of factors C and D as important. By doing this, it gives us no information about which levels are in fact important. The I-χ^2 method performs even worse and is not able to distinguish

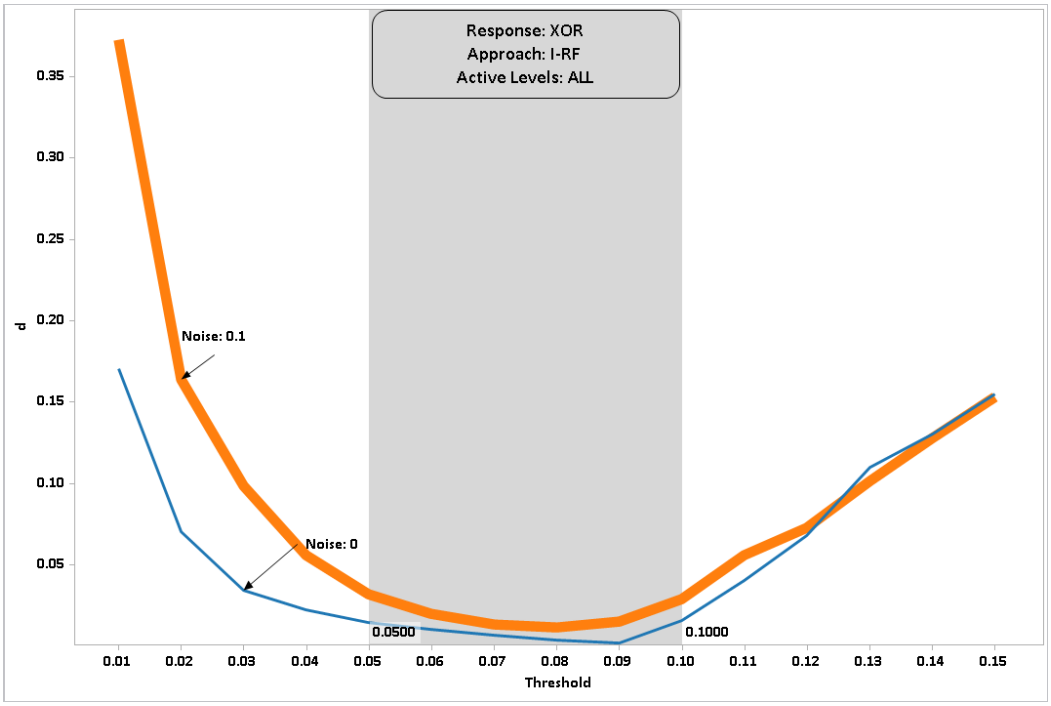

Figure 3.1: Performance curves for I-RF and XOR response. The usual threshold band between 0.05 and 0.1 is highlighted.

between the feature levels for either factors C or D. The I-CFS approach, as in the *OR* and *XOR* case continues to perform poorly with low sensitivity and specificity scores. With noise added to the system, the I-ACE, I-GBT and E-ACE approaches continue to perform very well. Similar to the no noise case, the I-RF performs well when there are only 5 active factors in the model. However, at a higher setting this approach breaks down and produces a very low sensitivity score. At the low active levels setting, the I-χ^2 approach performs poorly and is unable to distinguish between variable levels. The I-CFS again performs poorly in terms of both sensitivity and specificity.

The performance curves for the I-RF approach, shown in Figure 3.5 shows an interesting pattern. The value of d decreases rapidly between 0.01 and 0.05,

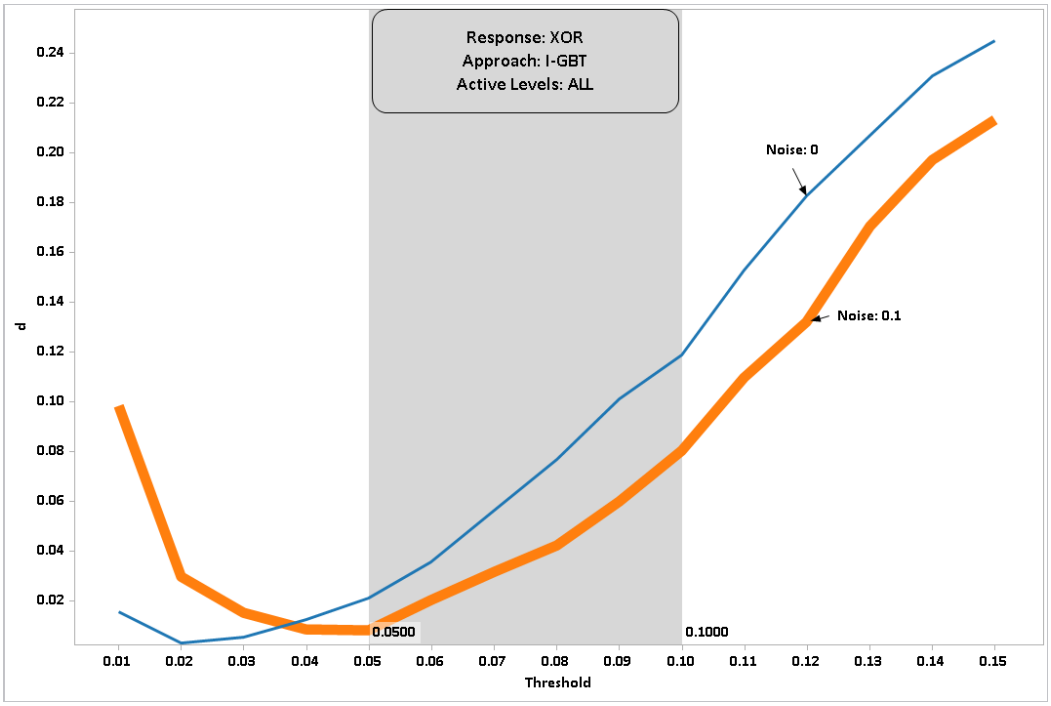

Figure 3.2: Performance curves for I-GBT and XOR response. The usual threshold band between 0.05 and 0.1 is highlighted.

stabilizes between 0.05 and 0.1 and then again decreases between 0.1 and 0.15. Thus, in theory, the I-RF results in Table 3.8 can be further improved by setting a higher value for the cut-off threshold. However, in practise, we would never know the true nature of the performance curve and therefore, for this experiment, we end up with a subset of features that contain many false positives.

3.5 Conclusion

Models in these complex environments can benefit from feature selection methods to extract compact, non-redundant feature subsets from the data. In many situations, additional model simplification can be achieved by extracting specific levels/values that contribute to variable importance. In this research, we proposed

Table 3.7: Simulation results for the OR response function

Noise	*Active Levels*	*Metrics*	*Ind ACE* / *I-ACE*	*Ind RF* / *I-RF*	*Ind GBT* / *I-GBT*	*Ind CFS* / *I-CFS*	*Ind* χ^2 / *I-*χ^2	*Embed ACE* / *E-ACE*	*Deleting OFAT GBT*
0	5	Sensitivity	1	1	1	0.66	0.94	1	0.94
0	5	Specificity	1	1	1	1	1	1	1
0	5	d	0	0	0	0.34	0.06	0	0.06
0	15	Sensitivity	1	1	1	0.573	0.967	1	0.497
0	15	Specificity	1	1	1	1	1	1	1
0	15	d	0	0	0	0.427	0.033	0	0.503
0	25	Sensitivity	1	0.936	0.956	0.344	0.928	1	0.181
0	25	Specificity	0.995	0.994	1	1	1	1	1
0	25	d	0.005	0.067	0.044	0.656	0.072	0	0.819
0.1	5	Sensitivity	1	1	1	0.7	1	1	0.94
0.1	5	Specificity	0.983	1	1	1	0.99	0.997	0.999
0.1	5	d	0.017	0	0	0.3	0.01	0.003	0.061
0.1	15	Sensitivity	1	1	1	0.52	1	1	0.422
0.1	15	Specificity	0.942	0.995	1	1	0.968	1	1
0.1	15	d	0.058	0.005	0	0.48	0.032	0	0.578
0.1	25	Sensitivity	1	0.944	0.98	0.476	0.968	0.96	0.14
0.1	25	Specificity	0.953	0.987	1	1	0.962	0.985	1
0.1	25	d	0.047	0.062	0.02	0.524	0.061	0.043	0.86

Table 3.8: Simulation results for the (OR,AND) response function

Noise	*Active Levels*	*Metrics*	*Ind ACE* / *I-ACE*	*Ind RF* / *I-RF*	*Ind GBT* / *I-GBT*	*Ind CFS* / *I-CFS*	*Ind* χ^2 / *I-*χ^2	*Embed ACE* / *E-ACE*	*Deleting OFAT GBT*
0	5	Sensitivity	1	1	1	0.92	1	1	0.6
0	5	Specificity	1	0.986	1	0.984	0.835	1	1
0	5	d	0	0.014	0	0.089	0.165	0	0.4
0	10	Sensitivity	1	1	1	0.59	1	1	0.731
0	10	Specificity	1	0.888	1	0.911	0.873	1	1
0	10	d	0	0.112	0	0.42	0.127	0	0.269
0.05	5	Sensitivity	1	1	1	0.92	1	1	0.88
0.05	5	Specificity	0.994	0.962	1	0.99	0.931	1	1
0.05	5	d	0.006	0.038	0	0.086	0.069	0	0.12
0.05	10	Sensitivity	1	1	1	0.59	1	1	0.69
0.05	10	Specificity	0.99	0.877	1	0.926	0.873	1	1
0.05	10	d	0.01	0.123	0	0.417	0.127	0	0.31

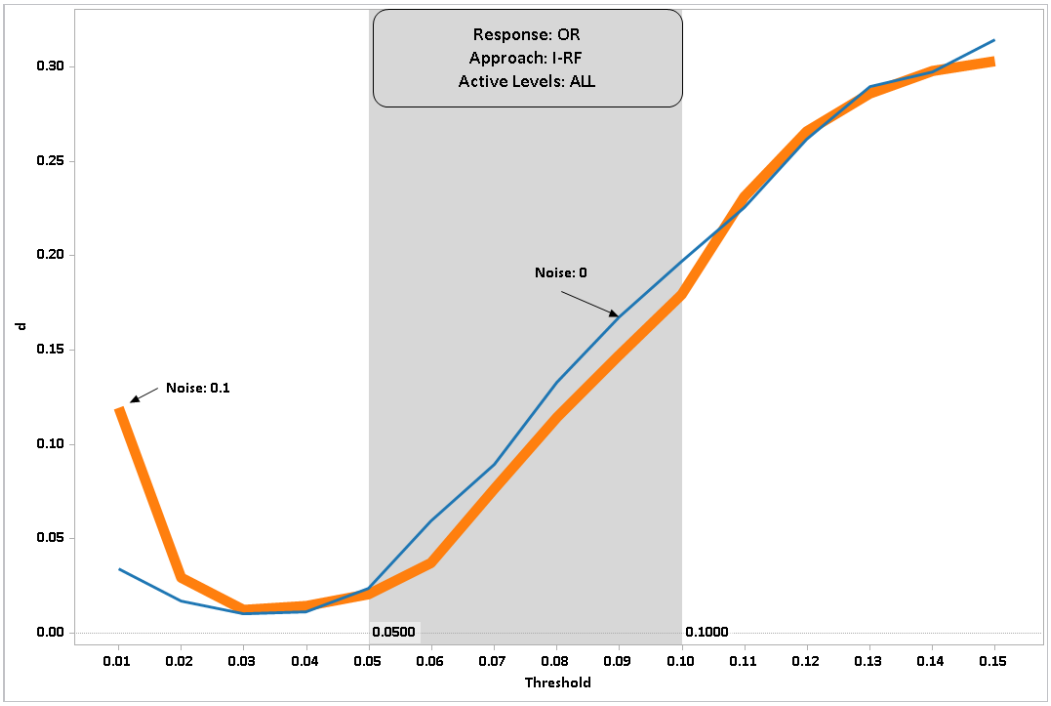

Figure 3.3: Performance curves for I-RF and OR response. The usual threshold band between 0.05 and 0.1 is highlighted.

different strategies to address this problem of *Feature Value Selection* and studied their comparative performance using simulated datasets.

Our first strategy used a simple data transformation to convert all categorical variables into a binary indicator variables. This allowed us to use standard, well researched feature selection methods on the binary incidence matrix to extract important feature levels/ values. Through simulated experiments, we showed how this approach works well, especially when used with tree-based feature selection methods. Amongst the tree-based methods, the I-ACE approach performs very well and has the added advantage since it uses a statistical hypothesis based approach to select significant feature levels. While the I-RF and I-GBT methods are competitive, their performance is sensitive to the cut-off threshold value, which often times is set

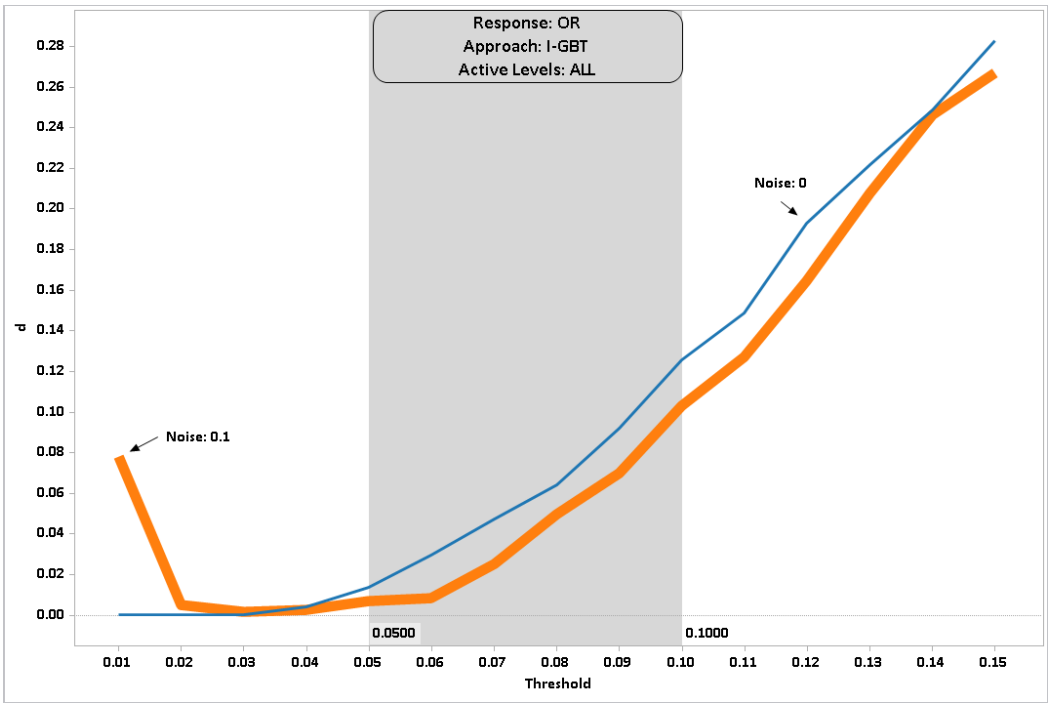

Figure 3.4: Performance curves for I-GBT and OR response. The usual threshold band between 0.05 and 0.1 is highlighted.

based on user preference. In general, this strategy is easy to implement since it can leverage existing feature selection methods and is compatible with future research as well. However, one of its drawbacks is that it needs extra storage space for the transformed incidence matrix.

To alleviate this problem, we proposed a strategy, labelled as E-ACE, that builds upon the ACE approach. A new measure for feature value importance was defined and computed as a part of the tree induction algorithm. In experimental comparisons, this method performed exceedingly well with very high sensitivity and low false alarms. Since this approach is based on the ACE algorithm, it inherits all the desirable properties that tree-based ensemble methods possess - from being robust to noise and outliers, to being able to handle strong non-linear interactions.

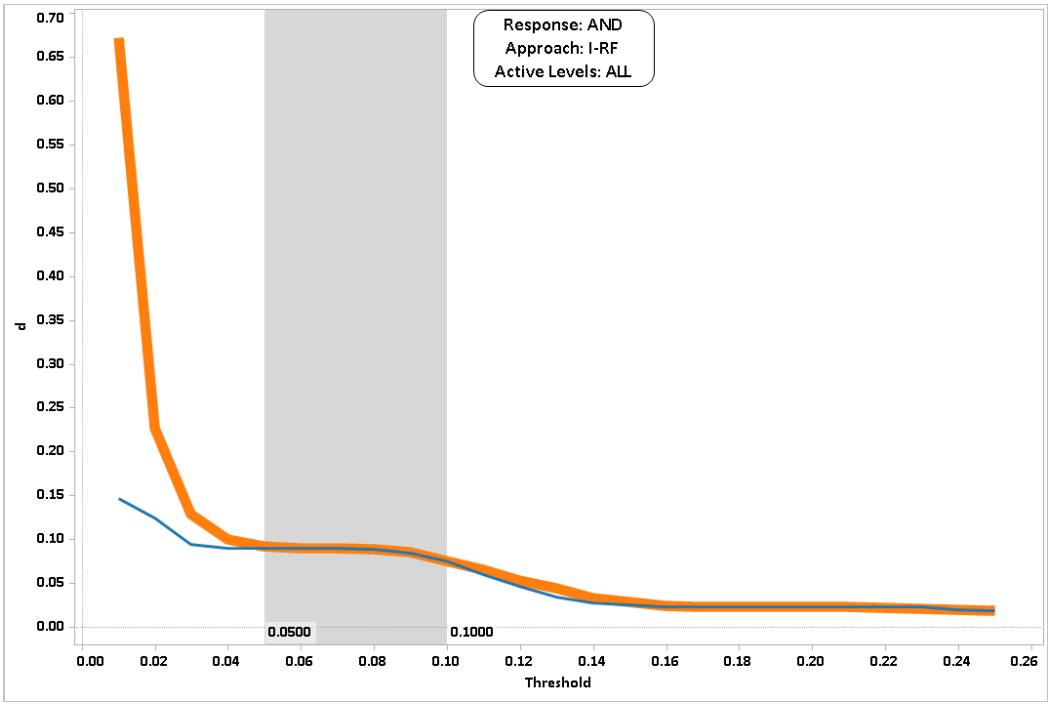

Figure 3.5: Performance curves for I-RF and AND response. The usual threshold band between 0.05 and 0.1 is highlighted.

Additionally, the artificial contrasts used in the ACE algorithm allow us to test the statistical significance of each feature value using *p-values*.

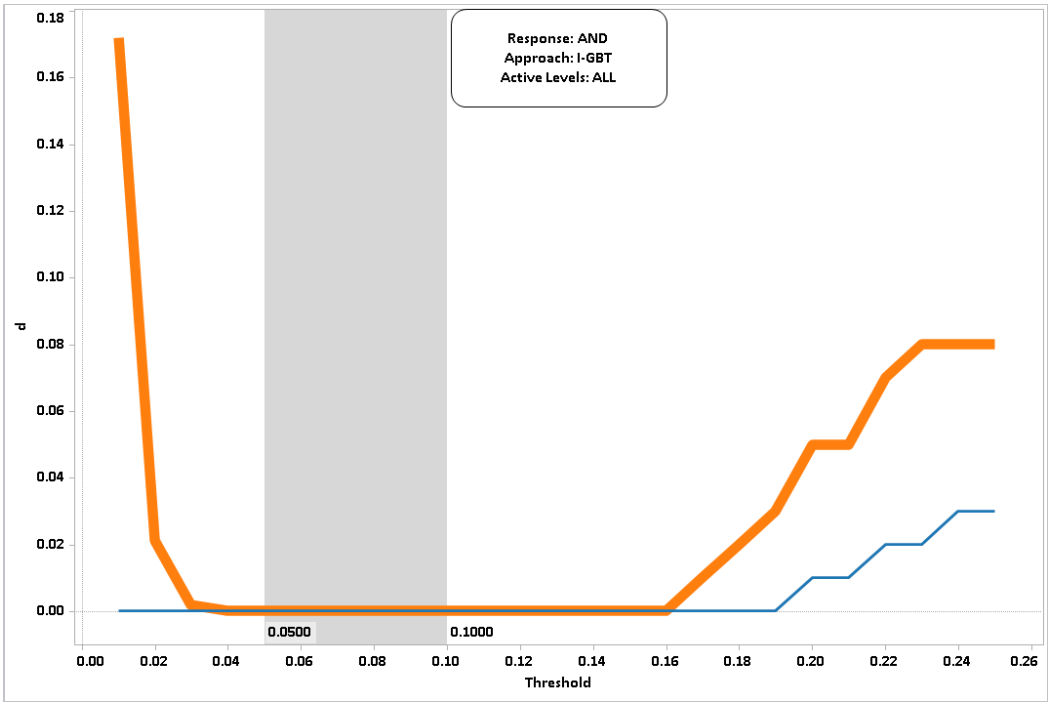

Figure 3.6: Performance curves for I-GBT and AND response. The usual threshold band between 0.05 and 0.1 is highlighted.

Chapter 4

HIGH DIMENSIONAL SUPPLY CHAIN SURVEILLANCE THROUGH NETWORK FEATURE SELECTION

4.1 Introduction

In a global market place, an effective and efficient supply chain network often provides a significant cost advantage over the competition. Rapid technological developments in product technology have significantly shortened the life cycles of products and have necessitated the need for highly agile supply chains; thus adding complexity to the entire supply chain network. This is an ever increasing problem as companies branch out into newer products and newer markets. As supply chains expand to envelope other organizations spread across the globe, there is a marked increase in the variety of risk factors they are exposed to. At the same time, the sheer number of nodes (processing / holding locations) and routes involved in catering to a wider audience has significantly reduced the ability to detect and localize elements of the network that may be contributing to poor supply chain performance.

Over the last decade, advancement in tracking and measuring systems have enabled the collection of vast amounts of data associated with supply chain network. Radio-frequency identification (RFID) and GPS based systems can provide accurate data regarding the exact location of each shipment thoughout the entire supply chain. Shipment specific data such as its product identification number (SKU number), weight, volume and other related characteristics can be easily traced and recorded. Enhanced tracking also enables us to measure the performance of the supply chain at individual shipment levels using metrics related to perfect order, order fulfilment lead time, cycle time, inventory level, supply chain costs etc. Additionally, network related attributes such as route capacity, as well as global and local risk factors (power outages, extreme weather events etc.) can often be layered

on top of shipment related data sources. With such information rich data sources, it is valuable to look for patterns in them that can help make more targeted and better-informed network improvement decisions.

Given a complex supply chain network with hundreds (even thousands) of nodes, the aim of a surveillance system is to aid in detecting problems (say, deterioration in performance metrics) and then localizing the source nodes that may be contributing to it. This is a nontrivial task given the sheer dimensionality of real-world networks, and the messy data they generate. Data sets collected from disparate sources are often riddled with outliers and missing values, exhibit strong non-linear relationships, and contain data in the form of continuous as well as categorical variables with significant redundancy. The aim of this research is to develop robust tools that aid in localizing supply chain problems to a subset of network nodes or routes while confronting the disparate, complex information associated with the network.

To accomplish these tasks, we propose to start with a transformation of the monitoring issue to a supervised learning problem and then employ robust, scalable learners and feature extraction strategies to identify and extract nodes(features) from the network - a solution we call *network feature extraction* (NFE). The solution generated here is envisioned to be a component of a surveillance systems which aims to improve supply chain visibility. The unique aspects of the proposed method are 1) capability to detect anomalies, not only in the graph structure, but within localized regions of this high-dimensional space of network attributes. 2) the capability to integrate disparate data from multiple sources which exhibit all the nuances of ”messy” datasets such as missing values, non-linear structures, and mixed variable types.

Although supply chain networks are a primary focus of this research and our discussion therein, we believe that the same set of tools and methodologies can be applied to a wide variety of applications. For example, air traffic networks could use these tools to identify which multi-stop routes or airports lead to high rates of lost baggages.While not a network in the traditional sense, manufacturing processes could use these methods to identify which tool combinations lead to poor yield.

4.2 Literature review

One approach for problem localization in large networks relies on determining the importance of an individual node or route within the overall chain by using node specific indices. In transportation networks, critical highway segments are identified and monitored using indices such as the volume / capacity (V/C) ratio (Dheenadayalu et al. (2004)) or the network robustness index (Scott et al. (2006)). While this detection and monitoring approach is simple to implement, creating separate monitors for hundred to thousands of nodes / routes can potentially increase false alarms.

Another widely used approach is to determine the importance of the node based on its location in the network. Measures based on node degree have previously been used by Bavelas (1948) and Freeman (1979). Borgatti (2006) and Arulselvan et al. (2009) proposed measuring node importance by computing the maximum network fragmentation that occurs as a result of deleting those nodes from the network. These approaches rely entirely on the structure of the network and are therefore, more useful for comparing and determining the robustness of different network during the planning stage. However, they are unable to assimilate the rich sources of data that are often found in real-world networks. Not only is it important to identify critical nodes in the network, but also to determine which covariates are involved in the failure at these nodes.

4.3 Background

Given a complex supply chain network with potentially thousands of nodes and routes, our goal in using a feature selection strategy is to find a compact, non-redundant subset of the network nodes for further investigation. Complex network designs, along with detailed information regarding the state of each shipment and node in the network can lead to information rich data sets. However, non-traditional characteristics of the resultant data sets make it necessary to choose a feature selection algorithm that can successfully handle the following challenges:

- *High dimensionality:* Real-world networks involve hundreds (even thousands) of nodes and can carry millions of shipments per day. Dozens of variables (attributes) per shipment or node are common. Combined together, this leads to really high dimensions for the data set and hence model over fitting is a strong concern. Methods are needed that enable the relevance of variables to be statistically quantified, thereby aiding model building.

- *Disparate data types:* Categorical variables, such as node and route identifiers, as well as continuous variables such as shipment weight, volume etc. are expected in such data. Dramatically different scales (units) might be present for numerical measurements. Traditional attribute standardization can collapse true relationships (structure) in the data (Friedman et al. (2001)) and hence methods that are invariant to scale are needed.

- *Missing data and outliers:* Dirty data with extensive missing values should be expected when data sources are numerous. Also, measurement system and tracking errors can lead to outliers in the feature space as well as the output

(response) space. To avoid extensive preprocessing efforts, methods that are intrinsically robust to missing values and outliers are needed.

- *Nonlinear relationships:* Due to the multivariate nature of the data, nonlinear interactions between shipment level variables and node variables are expected. For example, only shipments above a certain weight may have delay issues at certain nodes. Also, transient effects, such as supply chain problems that affect nodes during a certain time period are expected to exhibit themselves in the data set.

- *Variable masking:* A large fraction of the variables in the data set are expected to be highly correlated with each other. It is useful to identify important variables along with alternatives or replacements with similar information content.

The artificial contrasts with tree-based ensembles (ACE) method suggested by Tuv et al. (2006, 2009) has been successfully used in a wide range of applications to extract compact, non-redundant sets of variables from complex data sets. This feature selection method is based on ensembles of decision trees which can easily deal with thousands of variables (predictors) and can be applied to mixed variable types. Since these methods are invariant to scale, it is not necessary to extensively preprocess (standardize) the data. This also makes these methods robust to outliers [Breiman (2001)]. Additionally, tree models can intrinsically handle missing values through mechanisms such as surrogate splits [Breiman et al. (1984)]. More importantly, tree-based ensembles provide an embedded measure of variable importance that aids in selecting the most relevant features.

Since our proposed method - *network feature extraction* - is based on the ACE algorithm, we begin by providing an overview of the ACE algorithm (Section 4.3), followed by details of how it can be extended to extract feature values/levels (Section 4.3).

Feature selection using artificial contrasts

Tree based ensemble models, especially the random forest model used in our proposed method construct a set of simple decision trees, and use their weighted outcome to predict new data. The measure of variable importance for a single decision tree (T) is given by

$$VI(X_i, T) = \sum_{n \in N_T} \triangle I(X_i, n) \tag{4.1}$$

where $\triangle I(X_i, n) = I(n) - p_L I(n_L) - p_R I(n_R)$ is the decrease in impurity because of an actual (or potential) split on variable X_i at a node n of the optimally pruned tree T. For an ensemble of M trees, Random forests simply averages the importance score in equation 4.1 over all M trees.

$$VI(X_i) = \sum_{m=1}^{M} VI(X_i, T_m)/M \tag{4.2}$$

However, this only solves a part of the problem. With only variable importance scores, there is no obvious threshold to segregate important variables from a potentially large subset of covariates. The list of importance values does not come with an associated indication of which variables to include and which ones to ignore. Randomly setting this noise threshold often carries two types of risks with it. If we are too lenient in setting this threshold, we might be fitting the model to noise and thereby increase the rate of false alarms. If we are too strict with this threshold, we then run the risk of missing relevant signals.

To remedy this, the ACE method uses artificially created noise variables, also referred to as contrasts to set the noise threshold in the system. The logic be-

hind their method is quite elegant : The variable importance score from the ensembles is based on the relevance of the variable with regards to the target or response variables. For the variable to be relevant or significant, this score should be higher than the score for an artificially created variable (that is generated to be irrelevant to the target). That is, a higher variable importance score is expected out of a true relevant variable than from an artificially created noise (contrast) variable. Only a variable with a significant difference (as measured with a *p-value*) from the k^{th} percentile (say, 90^{th}) of artificial variables is considered to be an important feature. Therefore, this approach eliminates guess work by providing an objective measure based on *p-values* for defining significant variables and segregating them from irrelevant noisy ones.

A similar method is used to identify and remove redundant variables. A modified surrogate score, called a masking score, is computed between all pairs of variables (including contrast variables) using a gradient boosted tree (Friedman and Meulman (2003)). For a single replicate, an upper percentile (say, 75^{th}) of the masking score between the original variables and contrast variables is computed. After several replicates, a mean is calculated, which defines the threshold against which all masking scores between original variables is compared using a *t-test*. If the masking score between two original variables is statistically greater than the mean of the 75^{th} percentile, then the masking score is kept. To remove redundancy, the variable with the highest importance score is retained, and every other variable with a significant making score with it is eliminated.

Thus in summary, the ACE method generates *p-values* for : (a) identifying relevant subsets of variables that have a statistically significant impact on the response and (b) identify correlated or masked variables, thereby eliminating redundant variables.

Feature Value Extraction

Feature selection techniques such as the one described above enable us to identify the most important set of variables that impact the response. However, in many situations, identifying important variables is not always sufficient. Often times, it is necessary to investigate each variable and identify which specific values or levels of the variable contribute to its importance. For instance, identifying that shipment weight contributes to delays is not very informative. It is far more productive to figure out the exact weight values (say, heavy items > 50 pounds) that result in delays. Similarly, a model which indicates that a bulk of product defects occur during the manufacturing stage is probably not going to help us solve the problem. It would be far more meaningful to localize the problem to the exact manufacturing location (node in the network). This problem can be referred to as *feature value selection*, where one identifies the specific levels and/or ranges of the important covariates that are leading to relevant changes in the data. Dávila (2010) applied feature value selection using rule-based methods for public health surveillance. Subsequently, Shinde et al. (2012), expanded on this work and compared the performance of different implementation strategies for feature value selection.

The ACE algorithm can be easily extended to identify important feature values. To do this, we need to modify the learning algorithm to extract importance scores for individual levels within the categorical covariates. Consider a single decision tree. At each node of the tree, the splitting algorithm selects the variable (X_i) that provides the highest decrease in impurity as the variable to split the node on. Once a variable (say X_i with J_i category levels) has been chosen, we can then iterate through each category level ($j, j = 1$ to J_i) and compute its importance score, denoted as variable-value importance (VVI) as follows

$$VVI(X_{ij},T) = \sum_{n \in N_T} Split\ weight_n * \triangle I(X_{ij},n) \tag{4.3}$$

where: $Split\ weight_n = 1$ if the split is made on variable i, category level j at node n. Since individual category levels are evaluated sequentially, we enforce a *one-versus-rest* splitting rule at each node. This means that a split such as {a},{b,c,d} will be allowed, while, a split such as {a,b},{c,d} will not be allowed.

For an ensemble of M trees, we would simply average the importance score in equation 4.3 over all M trees.

$$VVI(X_{ij}) = \sum_{m=1}^{M} VI(X_{ij},T_m)/M \tag{4.4}$$

Since the feature value selection algorithm is based on the ACE algorithm, it inherits all the desirable properties that tree-based ensemble methods possess - from being robust to noise and outliers, to being able to handle strong non-linear interactions. Additionally, the artificial contrasts used in the ACE algorithm allow us to test the statistical significance of each feature value using *p-values*.

4.4 Methodology

Network transformation for supervised learning

From a graph-theoretic perspective, a supply chain network can be viewed as a directed acyclic attributed graph $G = (V,E)$ with nodes $V = \{v_1, v_2, \ldots, v_V\}$ and directed links E. A node in the network denotes a candidate location for holding inventory for the stock keeping unit (SKU) and the arcs represent transportation routes between the nodes. The nodes in the network are assumed to follow a structure defined in terms of functional/processing stages such as part procurement, manufacturing or assembly, transportation, distribution and retail. Thus each stage i ($i = 1\ to\ I$), denoted using variable X_i, represents a processing activity of a stock keeping unit (SKU) while each node j ($j = 1\ to\ J_i$) belonging to stage i and denoted using

variable X_{ij} represent the actual location at which the processing is carried out. A representative network is shown in Figure 4.1.

Using case-event data, a supply chain network can be transformed into a supervised learning problems in a simple, yet effective manner. To do this, the network is characterized as a series of transactions. We view each transaction T_n ($n = 1$ *to* N) that traverses through the network as being attributed. Each transaction can encode information that is abstracted from the network, the path traversed over the network, or from the transaction itself. For each transaction, we record the stages X_i as well as the actual nodes (X_{ij}*'s*) through which the shipment traversed across the network. This transaction framework allows for a flexible collection of disparate attributes (transaction, time or network related) to be combined from multiple sources. For example, we could record transactional attributes such as product price, package size, weight, life cycle stage etc. Additionally, macro-economic variables around the time of the shipment may also be included. Attributes associated with the network itself can be appended to the data. Examples of network related attributes could include the presence or absence of a variety of disruptive events (such as extreme weather or power outages etc.). Note that attributes associated with individual nodes in the network can also be added to the data set. However, a split based on a stage variable (X_i) will inherently consider all partitions of the nodes. In other words, this partition implicitly detects if node attributes are important to the target or not.

Let U_a ($a = 1$ *to* A) denote the a^{th} attribute and let U_{na} denotes its value associated with transaction T_n. Similarly, let Y_n denote the variable that measures the supply chain performance metric at the transaction level. For example, one possible definition of Y_n could be the turn around time for the shipment or we could

use an indicator for whether the shipment was on time or late. Thus, the complete set of variables that characterizes a transaction is represented by $\{X_{ij}, U, Y\}$

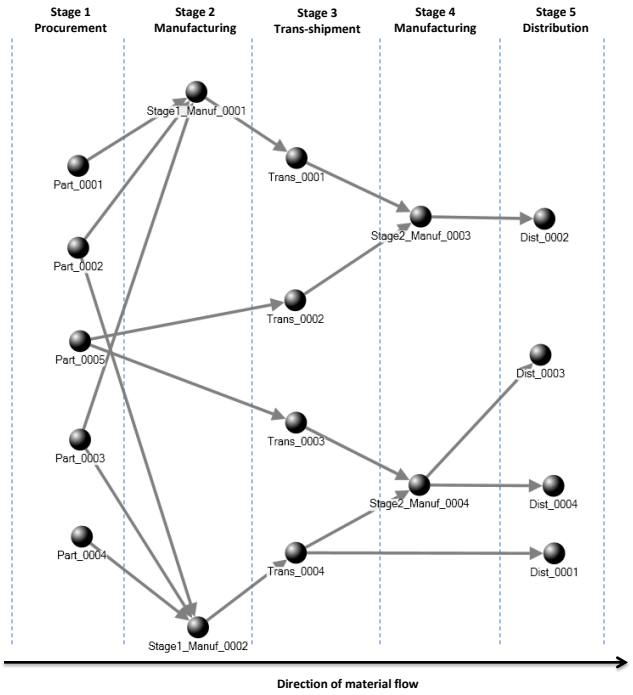

Figure 4.1: Example of a supply chain network with five stages : Stage 1 (procurement) with five facilities, Stage 2 (manufacturing) with two facilities, Stage 3 (trans-shipment) with four facilities, Stage 4 (manufacturing) followed by a distribution stage with four locations

A sample representative data table for the network transformation with regards to Figure 4.1 is shown in Table 4.1. Notice that the indicator vectors *weather* (U_1) and *power outage* (U_2) denote the presence (1) or absence (0) of disruptive events on the path that the shipment traversed and therefore represent network-level attributes. The column vectors *weight* (U_3) and *volume* (U_4) represent transaction-level attributes. For each shipment the node through which it passed (X_{ij}) is recorded in the appropriate stage that contains the node. The response vector (Y_n) denotes whether the transaction arrived on time or not.

Given this set up, we are interested in identifying a compact, non-redundant set of variables - stages (X_i) and network/transactional attributes(U_a) - that have a statistically significant impact on he supply chain performance measure of interest (Y). The current data table format enables us to address this problem through the use of a wide range of feature selection techniques [Kohavi and John (1997); Breiman (2001); Tuv et al. (2009); Guyon et al. (2002)]. Feature selection techniques will enable us to identify which stages in the supply chain process provide us with information regarding the response variable. However, to provide more visibility, we need to drill this problem down to the node level (X_{ij}) and identify the exact set of nodes that interact to produce the observed effects. To do this, we use the feature value selection technique described in section 4.3. Because the feature value selection method is applied to extract network features, we call this solution the *network feature extraction* (NFE) method.

Since the NFE method is based on the ACE algorithm, it generates *p-values* for each variable (U, V, X_{ij}) by comparing the its importance score with that of artificially generated noise variables. This provides us with a strong statistical basis for segregating variables that have a statistically significant impact on the output measure (Y) from insignificant variables. Also, the *p-value* threshold can be adjusted to balance the true and false positive rates. The usual rejection rule of ($p \leq 0.05$) often leads to a higher false positive rate when multiple hypotheses are compared. It is easy to incorporate a simple Bonferroni adjustment for the *p-value* threshold in the NFE method to counter this problem of multiple hypothesis testing (Hochberg and Tamhane (1987)).

Table 4.1: Example of a network transformation into a data table

Id	*Stage 1*	*Stage 2*	*Stage 3*	*Stage 4*	*Stage 5*	*Weather Disruptions*	*Power Outages*	*Weight*	*Volume*	*On-time?*
	X_1	X_2	X_3	X_4	X_5	U_1	U_2	U_3	U_4	Y
1	Part-0001	Stage1-Manuf-0001	Trans-0001	Stage2-Manuf-0003	Dist-0002	No	No	3.1	05	Yes
2	Part-0002	Stage1-Manuf-0002	Trans-0004	Stage2-Manuf-0004	Dist-0004	No	No	2.8	10	Yes
3	Part-0002	Stage1-Manuf-0002	Trans-0004		Dist-0001	Yes	No	3.8	20	No
4	Part-0005		Trans-0003	Stage2-Manuf-0004	Dist-0003	Yes	Yes	3.5	35	Yes

Feature visualization using partial dependency plots

After important variables and their associated variable levels have been identified, an important step is to understand the nature of the dependency of the output on these the relevant covariates [Friedman et al. (2001)]. Visualization of the predicted function $f(x)$ over the input space defined by the sub-network and its attributes is a powerful tool to interpret the model. However, direct visualization is limited by the number of input variables that can be handled successfully. Instead, we apply the graphical summary referred to as partial dependence plots to add interpretability to our model [Friedman (2001)]. Partial dependence functions represent the effect of the covariate on the output after accounting for the average effects of other covariates. Plotting the partial dependence of $f(x)$ can help us gain a deeper understanding of the nature of the interaction between the problem nodes and the attributes associated with them.

4.5 Simulation studies and results

In this section, we present simulation studies to demonstrate how the network feature extraction method can be used to detect problems in the supply chain and localize it to the source nodes (Section 4.5). We also present an example of how partial dependency plots can be combined with NFE to visualize the interaction between network nodes (X_{ij}'s) and transactional attributes (U_a's) (4.5). We end this section with an illustration of how the NFE method can potentially be used to provide a minimalistic summary of the network when entire routes are affected (4.5).

Problem localization through network feature extraction

Consider the network shown in Figure 4.2. This network consists of 60 nodes spread across four stages and is unique in that any downstream node (successor) can be reached by any of its upstream (predecessor) nodes. For this study our learning objective is to identify nodes in this network that result in some form of deteriorated supply chain performance (say, shipment delays).

Simulation setup

We simulated data to represent the flow of a product over this network in accordance to the flow relationships defined by the nodes and directed arcs of the network. To begin, we randomly choose k nodes out of 60. A binary response variables Y is defined such that if the shipment passes through any of these k selected nodes, then it is flagged as delayed (denoted as class 1). If the shipment bypasses these nodes, then it arrives on time (denoted as class 0). To make the experiments more realistic, we introduce another parameter, *probability of delay* (p_d), that defines the probability that a shipment passing over any of the k active nodes gets delayed. A value of $p_d = 1$ would imply that all shipments passing through an active node are delayed. A value of $p_d = 0.1$ indicates that there is only a 10 % chance that the

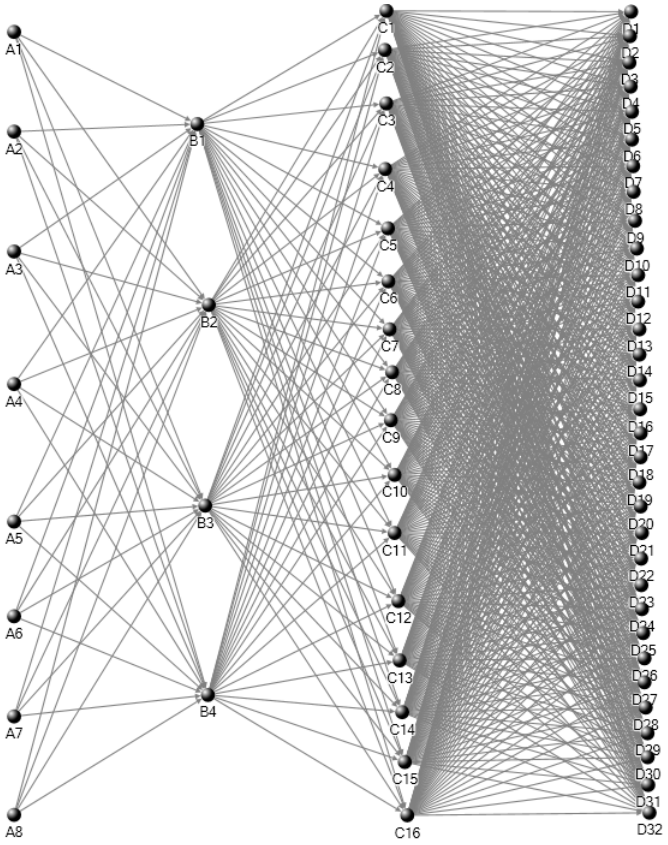

Figure 4.2: Simulated network design for experimentation

shipment will get delayed at each active node. For our experiments, we used three levels for k : 1, 6 and 12 and two levels for p_d : 0.25 and 0.10. Since the flow of shipments over this network is stochastic in nature, we ran five replicates at each experimental setting. Note that at each replicate, we start by randomly selecting k new nodes.

Metrics for measuring accuracy

We used the following metrics to gage the performance of the network feature extraction method.

$$Sensitivity = \frac{Number\ of\ correctly\ detected\ problem\ nodes}{Total\ number\ of\ nodes} \tag{4.5}$$

$$Specificity = 1 - \frac{Number\ of\ nodes\ detected\ in\ excess}{Total\ number\ of\ nodes} \tag{4.6}$$

We expect successful methods to have large sensitivity and specificity. The values for sensitivity and specificity are computed for each replicate separately and are then averaged over the five replicates.

Results

The result of applying NFE on the simulated experimental data is summarized in Table 4.2. In addition to reporting the experimental setting and the corresponding accuracy metrics, we also report the average proportion (over five replicates) of the data set that was classified as delayed (class 1). This metric is a measure of the unbalance in the data set (class 1 versus class 0) and indicates the rarity of delays (problems) in the network. Also note that we used a simple Bonferroni adjustment for the *p-value* threshold in the NFE method i.e. $p_{adj} = p/n$, where n is the number of nodes in the network.

With just one active node, the proposed method has a perfect score for sensitivity (true positive rate) and specificity (false positive rate). This is important for surveillance and monitoring models since a lot of time and effort is often wasted in investigating false alarms. This is encouraging, especially since the data set is highly unbalanced i.e. only a very rare fraction of the data set contains delayed shipments (class 1).

For six active factors, the dataset is again unbalanced with approximately 6 % (at $p_d = 0.1$) and 14 % (at $p_d = 0.25$) of the entire shipments data being flagged as delayed. In spite of such rare proportions, the NFE method provides ideal results by exhibiting perfect scores for sensitivity and specificity. In other words, none of the significant signals from the network are missed, and again, no false signals are detected. As we increase the number of active factors, the problem becomes much more difficult to handle as more signals need to be detected. The sensitivity drops

Table 4.2: Mean accuracy metrics for NFE method after five replicates

k	P_d	*Sensitivity*	*Specificity*	*% of shipments with delays*
1	0.25	1	1	2.87 %
6	0.25	1	1	14.30 %
12	0.25	0.96	1	20.23 %
1	0.1	1	1	1.16%
6	0.1	1	1	6.1 %
12	0.1	0.83	1	9.2 %

to 0.83 when the probability of delay is further reduced to 0.1. At $p_d = 0.25$, we see only a slight deterioration in performance. Sensitivity at this setting is fairly high at 0.96 with perfect specificity scores.

Figure 4.3 graphically illustrates how the method is used to summarize the network from one of the replicates (12 active nodes, $p_d = 0.25$ from this method. The active nodes identified by the proposed method are highlighted in bold along with the paths that join them. It is evident from the figure that the NFE method can be used to extract a sub-network of nodes and arcs that contain the maximum information regarding the performance measures for the entire chain. This is clearly an advantage since further investigations can be focused on this sub-network rather than the entire network of 60 nodes.

Detecting interactions between nodes and transactional attributes

Localizing supply chain problems to a subset of nodes is only a starting point towards network improvements. Many a times, performance deterioration at a node can be further explained by the attributes of the shipments that flow through those nodes. For example, a large proportion of delays may be linked to certain types of shipments such as bulkier items. Similarly, performance of the supply chain can be linked to node or route specific attributes such as blackouts, extreme weather con-

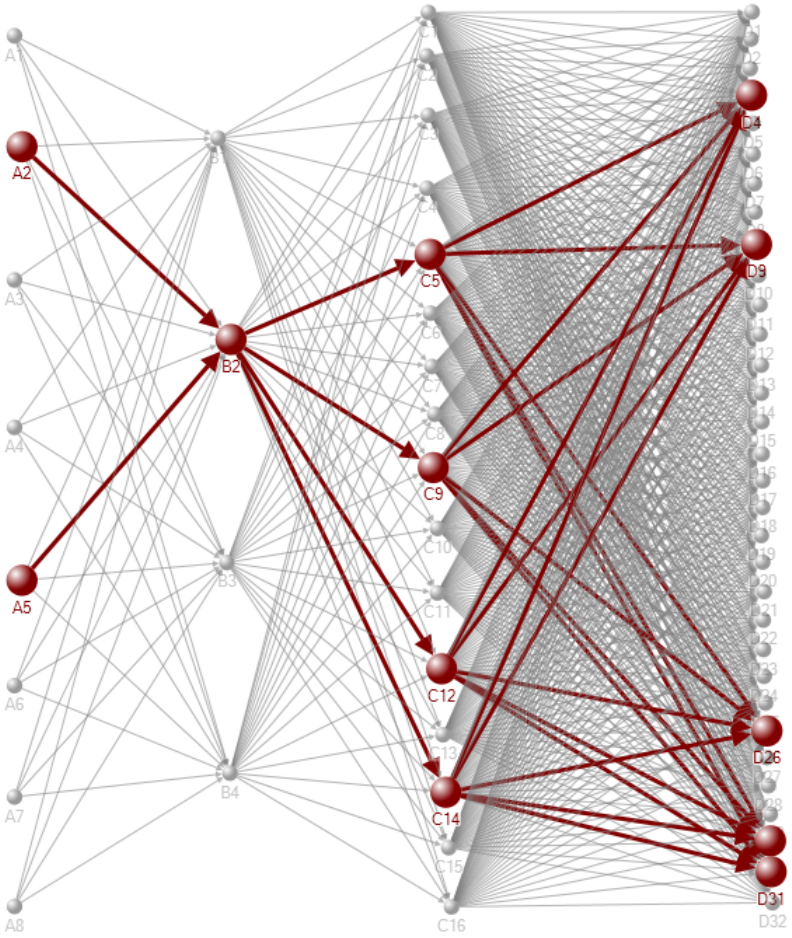

Figure 4.3: Nodes identified using NFE as being important in predicting shipment delays.

ditions etc. Thus drilling down the problem from a node level to a shipment/node attribute level can often provide critical improvement insights. One advantage of our proposed method is its ability to assimilate a wide variety of shipment and/or network related information.

We modify our previous experiment to incorporate shipment level data to illustrate the usefulness of our method in detecting node-shipment attribute interactions. Two new factors, shipment weight (pounds) and volume (m^3) are recorded for each shipment. The response variable is defined such that at the active nodes,

shipments over 70 pounds get delayed. The probability of delay (p_d) at the active nodes is held at 0.25 and we used 6 active nodes in the network.

Results of the experiments are summarized in Table 4.3. As in the previous experiment, the NFE method does an excellent job of identifying the problem nodes in the network. Additionally, *weight* is correctly identified as an important variable in predicting delays. Partial dependency plots between *weight* and each stage of the network for one of the replicates (replicate 5) are shown in Figure 4.4. Visually, it is evident that for each stage, the behavior of the partial dependency function is distinctly different for the active nodes as compared to the inactive nodes. Also, there is a marked difference for shipment weight more than 70 pounds. Thus, combining partial dependency plots with the NFE method can significantly help in understanding the nature of the interaction by allowing analysts to localize and visualize fault patterns in the network.

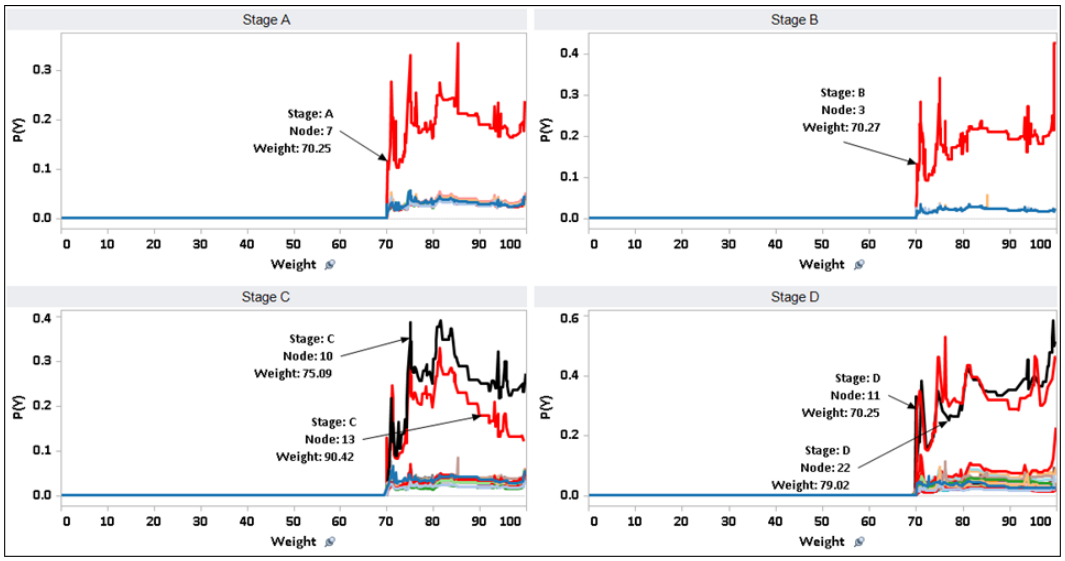

Figure 4.4: Partial dependency plot illustrating node-weight interaction for replicate 5.

Table 4.3: Accuracy metrics for NFE method with extraneous attributes

Replicate	*Active Factors*	*Factors Detected*	*Sensitivity*	*Specificity*	*% of shipments with delays*
1	{A6, B2, C2, C4, D4, D21}, Weight	{A6, B2, C2, C4, D4, D21}, Weight	1	1	4.13%
2	{A4, B1, C10, C13, D28, D30}, Weight	{A4, B1, C10, C13, D28, **D27**, D30}, Weight	1	0.983	4.33%
3	{A2, B2, C10, C16, D14, D28}, Weight	{A2, B2, C10, C16, D14, D28}, Weight	1	1	4.01%
4	{A4, B1, C11, C15, D12, D20}, Weight	{A4, B1, C11, C15, D12, D20}, Weight	1	1	4.05%
5	{A7, B3, C10, C13, D11, D22}, Weight	{A7, B3, C10, C13, D11, D22}, Weight	1	1	3.76%

Flow over structured real-world networks

The previous example demonstrated the usefulness of our proposed method over a densely connected network. However, geographical feasibility, delivery and manufacturing costs, risk strategies and other such factors influence the design of the network. Most real-world networks therefore, exhibit local structures. For example, a manufacturing plant may use only a subset of suppliers and in turn may only deliver products to a few distribution centres. In this section, we demonstrate the

applicability of our method to one such supply chain network. The chain described in this section is an actual supply chain model that has been implemented in practice and therefore exhibits complexities that are associated with real-world supply chains. For a complete description of this model, please refer Willems (2008).

Figure 4.5 is an example of a supply chain used in the delivery of computer peripheral equipment. The network allows for three possible shipment routings: 1) *Procurement* $\rightarrow$ *Stage 1 - Manufacturing* $\rightarrow$ *Transshipment* $\rightarrow$ *Stage2 - Manufacturing* $\rightarrow$ *Distribution*, 2) *Procurement* $\rightarrow$ *Transshipment* $\rightarrow$ *Stage2 - Manufacturing*$\rightarrow$ *Distribution* and 3) *Procurement* $\rightarrow$ *Stage 1 - Manufacturing* $\rightarrow$ *Transshipment* $\rightarrow$ *Distribution*.

Consider the route *Part-0001* $\rightarrow$ *Stage 1 - Manuf-0001* $\rightarrow$ *Trans-0001* $\rightarrow$ *Stage2 - Manuf-0003* $\rightarrow$ *Dist-0002*. It is evident from the figure that the entire information content of the route is contained in a single node - *Part-0001*. Put another way, in the context of this study, *Part-0001* represents the minimal node set that contains all necessary information regarding the route. Similarly, the route *Part-0005* $\rightarrow$ *Trans-0003* $\rightarrow$ *Stage2 - Manuf-0004* $\rightarrow$ *Dist-0003* only requires one of two pairs - $\{$*Part-0005, Dist-0003*$\}$ or $\{$*Trans-0003, Dist-0003*$\}$ - to uniquely identify it.

We used the NFE method on data simulated over this chain. The response variable was created such that all shipment that are routed through either of the above two routes were reported as delayed. It can be seen from Table 4.4 that the NFE with it's redundant feature elimination algorithm, is correctly able to detect unique route identifiers. Therefore, for networks with local structures, the NFE method can potentially provide a minimalistic set of nodes for monitoring and further investigation.

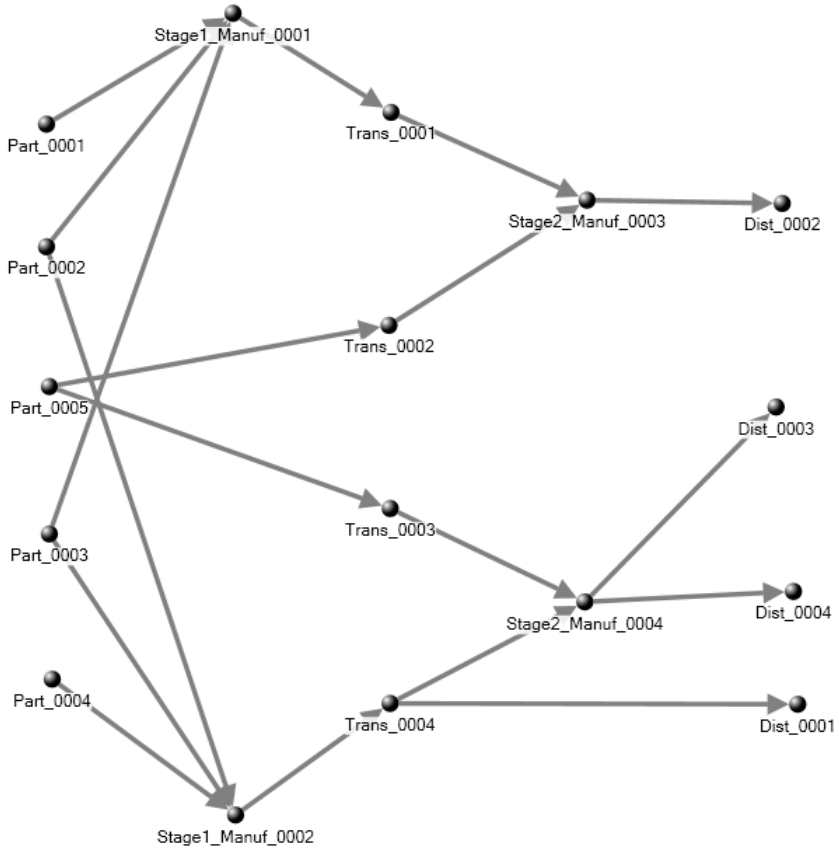

Figure 4.5: A supply chain network for computer peripheral equipment. Willems (2008)

Table 4.4: NFE on computer peripheral equipment chain

Path	*Unique path identifiers*	*Nodes identified using NFE*
{*Part-0005 → Trans-0003 → Stage2 - Manuf-0004 → Dist-0003*}	{*Part-0005, Dist-0003*} or {*Trans-0003, Dist-0003*}	{*Part-0005, Dist-0003*}
{*Part-0001 → Stage 1 - Manuf-0001 → Trans-0001 → Stage2 - Manuf-0003 → Dist-0002*}	{*Part-0001*}	{*Part-0001*}

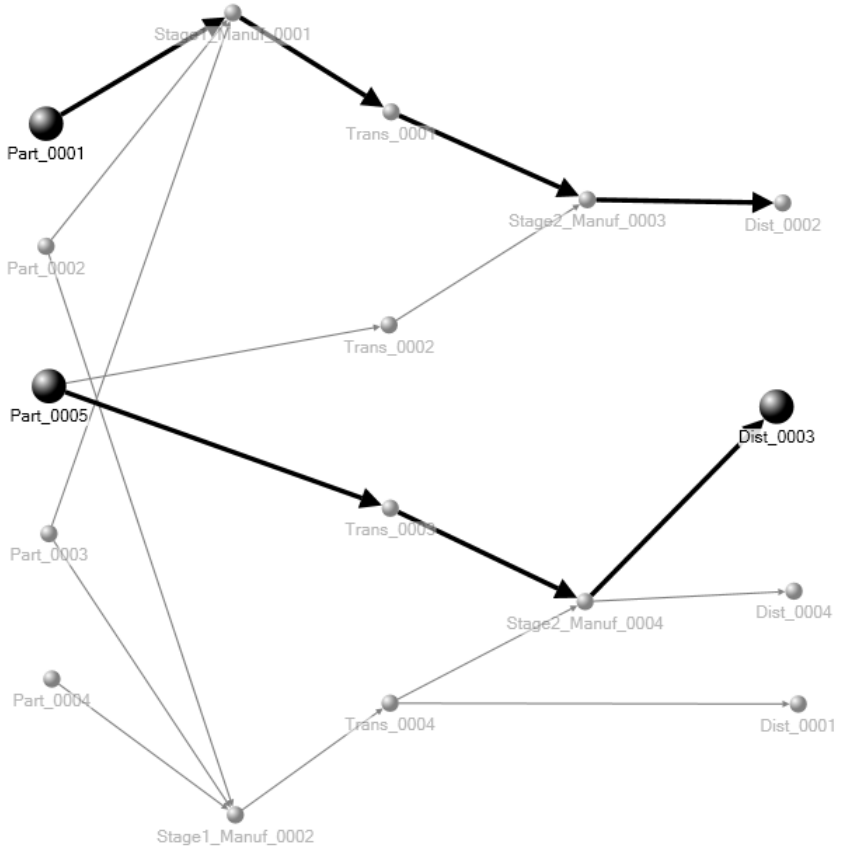

Figure 4.6: Network feature extraction applied to a supply chain network for computer peripheral equipment. Willems (2008)

4.6 Conclusion

With supply chains exhibiting increasing levels of complexity in design and information sources, traditional methods for surveillance are no longer sufficient. Using this paper, we showed how modern, state of the art machine learning algorithms can be used to provide a robust solution to the surveillance problem. We started by showing how a network surveillance problem can be converted into a feature value selection problem using a simple, yet efficient transformation of the network to a transactional data format. Applying feature selection methods for tracing supply chain problems to their root nodes needed an extension : a mapping from feature

selection to feature value selection. We recommended an extension of the tree-based feature selection method (Tuv et al. (2009)) that allowed us to investigate each stage of the supply chain so as to localize the problem at the node level. This solution that we call *network feature extraction* along with the transactional view of the supply chain network, enables us to assimilate a wide variety of information sources. Also, given that it is based on the ACE algorithm, it inherits all the properties that are necessary of good learners - robustness to outliers, ability to handle missing values, and detect strong nonlinear interactions.

Using simulated data, we showed that the NFE method possesses good sensitivity and low false alarm rates even when applied to networks that exhibit infrequent problems. Not only does the NFE method enable us to identify problem nodes in the network, but also allows us to detect complex interactions between problem nodes and their corresponding transactional attributes. We showed how partial dependency plots combined with the NFE method can be used to gain significant insights and creating rich visualizations into such interactions.

Finally, we illustrated how the NFE method can potentially be used to provide a minimalistic summary of the network when entire routes are affected (4.5). Initial results are encouraging and we plan to investigate this property further in subsequent research.

Chapter 5

SCENARIO ANALYSIS OF TECHNOLOGY PRODUCTS WITH AN AGENT-BASED SIMULATION AND DATA MINING FRAMEWORK

5.1 Introduction

The revolution of digital technology, coupled with competitive pressures has drastically reduced the life cycle of technology products. The semiconductor processor market is a fitting example. The pursuit of Moore's law has resulted in considerable advances in silicon-based technology. Processing power, measured in millions of instructions per second (MIPS), has steadily risen because of increased transistor counts. Simultaneous advances in process technology have resulted in higher yields, thus making it possible to produce less expensive, more powerful processors. With each technological breakthrough, chip manufacturers are able to introduce newer, better processors at a faster rate.

In this highly competitive market, multiple products with moderate differences in performance and price are often introduced simultaneously, and therefore, compete for the same unit of demand. As a consequence, the adoption success of a newly launched product is dependent on its value proposition compared to that of its competitors. It is also important to be cognizant of the fact that most business decisions pertaining to specific products can have overreaching and often counter productive effects on other internally competing products. For example, when a higher-technology product is made available to consumers at a lower price, it has the potential to cannibalize the market demand for the products already competing at that lower price point. Because product life cycles are short, changes in demand structure for products have serious implications on scheduling of supply, manufacturing and distribution capacity. Along with simultaneous price change decisions, the occurrences of additional business scenarios (competitor strategies, discounts,

product switching costs, etc.) make it very difficult to predict effects using simple intuition. As a result, models are required to capture the interaction effects of scenarios thereby aiding in quantifying their impact on the market share of certain price groups.

Econometric models such as the regression based elasticity models proposed by Deaton and Muellbauer (1980) have been incorporated in many tactical decision support systems. The theory of diffusion of innovations has been an area of ongoing research since the seminal work of Bass (1969). Over the years there has been a proliferation of mathematical models that are extensions of the Bass diffusion model. Often, these studies have focused on building market response models that explain the aggregate dynamics of new product entries, from their introduction to their complete penetration into their potential markets. Concurrently, considerable work has been done on identifying the characteristics of individual consumers and their motivation to adopt new products [Rogers (1995)]. In their review of diffusion research, Gatignon and Robertson (1985) refer to the mathematical modeling type of diffusion research as diffusion modeling research, and the behavioral studies as consumer diffusion research, and suggest that ”an integration of the behavioral and modeling literatures on diffusion could be beneficial to both constituencies”. However the complexities of the technology substitution and diffusion mechanism for high-technology products requires modeling approaches that substantially extend these traditional models.

Data sets are sparse due to short life cycles (and aggravated as the substitution of products at different points in time results in considerable missing data) and the simultaneous effects of multiple scenarios. The weaknesses of classical models to simulate and predict simultaneous interactions of adaptive components in complex systems has led to a great interest in agent-based simulation (ABM) approaches. Complex adaptive systems provide another avenue for studying emergent phenomenon such as product diffusion. Emergent phenomena, non-linear dynamics, and path-dependent behavior are some illustrations in which ABMs are used to study and analyze systems instead of traditional modeling methods. ABMs enable us study interrelationships among autonomous agents and interactions between agents and their environments in evolutionary settings. In ABM we can show interaction of agents systematically by defining decision-makers (agents), set of interaction rules and processes of changing states.

In this study, our aim is to create a framework that allows us to simulate and analyze the effect of multiple business scenarios on the adoption behavior of a group of technology products. We view diffusion as an emergent phenomenon that results from the interaction of consumers. To this end, we present the use of an ABM in which potential adopters of technology product are allowed to be influenced by their local interactions within the social network. Along with social influence, the effect of product features is an important consideration for technology products. In this research, we incorporate feature sensing attributes to the consumer agents along with sensitivities to social influence. The model encompasses utility theory and discrete choice models in the decision making process for the consumers.

Models of high fidelity such as ABMs necessitate a strategy to analyze and interpret the possible non-linear relationships amongst the various parameters of the simulation model. Increased fidelity in the model is attenuated if the quantitative

summaries used for decision making are not sufficiently expressive. To that end, one contribution of this paper is the use of modern data mining approaches to derive actionable knowledge from the agent-based simulation models. These methods allow us to summarize models outputs in order to quantitatively evaluate the effects of scenario decisions (such as prices). One important task is to identify inputs that are important contributors to the model results and another task is to graphically summarize the effects of such contributors. We employ decision tree based feature selection methods to identify contributors. Such methods have been shown to handle numerical and categorical inputs, complex, interactive, and non-linear models, as well as provide robustness to input-space outliers [Tuv et al. (2009)].

Finally, we present a realistic case study that demonstrates the ability of this framework to model changes in market shares for a group of products in response to business scenarios such as new product introduction, product discontinuation under pricing strategies. The models and other tools developed here are envisioned to be a part of a recommender system that provides insights into the effects of various business scenarios on shaping market shares of different product groups.

Section 5.2 provides a literature review. Sections 5.3, 5.4, and 5.5 describe our agent-based simulation model. Section 5.6 describes our use of an expressive machine learning model to identify important factors and graphically describe effects. Section 5.7 presents a representation of a high-technology example and Section 5.8 provides conclusions.

5.2 Literature Review

Different analytical and empirical models have been proposed to address specific business scenarios that impact product demand. Empirical models such as the Cross Price Elasticity models [Deaton and Muellbauer (1980),Green and Alston (1990)] have traditionally focused on modeling price demand relationships. These regres-

sion based approaches are suitable for products with longer life cycles. However, the technology substitution process poses unique challenges for modeling. Data sets are sparse due to short life cycles; a problem that is further aggravated as the substitution of products at different points in time results in considerable missing data.

The Bass diffusion model and its variants have been used for market analysis and demand forecasting of new products [Bass (1969)]. The model assumes that potential adopters of an innovation are influenced by two means of communication - mass media and word-of-mouth. Innovators tend to adopt a new technology as a consequence of external influences, whereas imitators are influenced by those who have already adopted. This model describes the process of how a new product gets adopted as an interaction between users and potential users. Fisher and Pry (1972) extended this single product model to a two product framework that represents the process by which a new technology product replaces or substitutes an older one in the market. A drawback of these models lies in the underlying assumption of a homogeneous population and perfect mixing amongst individuals of the population [Tanny and Derzko (1988)]. However, it has been shown that many real world social networks represents a set of individuals with some pattern of interaction or ties between them.

Diffusion models have also been integrated with other learning algorithms to capture and analyze scenario information. For example, Yelland et al. (2010) used a combination of the Fisher and Pry models and Dynamic Linear Models [West and Harrison (1997)] to capture the diffusion process as well as time series and seasonal components of product demand. Meixell and Wu (2002) and Wu et al. (2006) proposed an approach to analyze demand scenarios in technology-driven markets where product demands are volatile, but follow a few identifiable life-cycle patterns.

They demonstrated that products could be clustered by life-cycle patterns, and subsequently, within each cluster, identified leading indicator products that provided advanced indication of changes in demand trends. Using the Bass growth model and a Bayesian update structure, their proposed method provided a framework for scenario analysis by focusing on parametric changes of the demand growth model over time.

Several studies can be found on the use of ABM for technology diffusion. Jager (2006) presented a comprehensive survey of ABM applications for studying consumer behavior. Garcia (2005) provided a simple ABM to demonstrate its usefulness in a competitive environment. Ma and Nakamori (2005) built a multi-agent model to simulate the process of technological innovation as an evolutionary process with two types of agents : producers and consumers. Using their model agent-based model, they showed how consumers incomplete information and diversity of consumers demand could prevent producers from monopolizing the market by means of technological innovation. Delre et al. (2007) modeled diffusion dynamics of consumer agents under the effect of social influences. They showed the effects of heterogeneity and degree of network randomness on the speed of diffusion. An agent-based model was presented by Delre et al. (2007) to study the effects of different timing and targeting strategies for promoting new products. North et al. (2010) presented a multi-scale agent-based consumer market model and a structure to calibrate, verify, and validate the it.

Recently, agent-based models have been used to study the potential behavior of new electricity technologies. Hamilton et al. (2009) used an agent-based model of technology diffusion where bounded rational agents are faced with uncertainty about the performance of the new technology versus the old technology as well as spatial externalities such as fashion effects. Athanasiadis et al. (2005) used agent-

based models to control consumer demand by supporting interaction between consumers in a diffusion mechanism. In another paper, Ma and Nakamori (2009) compared the advantages and disadvantages of optimization models and agent-based modeling for modeling technological change in different energy systems.

The impact of the structure of a social network on the spread of innovations has been an actively researched issue. Montanari and Saberi (2010) demonstrated a model where under competing alternatives agents have the ability to adopts a new behavior based on its neighbors. Also, in their model the pay off for agents increases as more neighbors adopt the same choice. Guardiola et al. (2002) considered upgrade costs in modeling diffusion of innovations on a social network. Speed and other properties of diffusion are affected by network structure. Bohlmann et al. (2010) analyzed network topologies and communication links between innovator and follower market segments. Rahmandad and Sterman (2007) compared agent-based and differential equation models and analyzed the effect of individual heterogeneity and different network topologies.

5.3 An Agent-based model of processor markets

Purpose

Technology products are characterized by short life cycles and as a consequence, the adoption success of one product is correlated to that of its competitors. The purpose of this model is to study the adoption behavior of one or more technology products under different introductory market scenarios. We plan to use the agent-based framework to launch products with varied technological capabilities and pricing levels in an artificial society of technology consumer and study the conditions that can lead to successful adoption for target products. Subsequently, the model will also be used as an experimental test bed and outputs of the simulation will be analyzed using data mining models to map out optimal penetration strategies.

Environment

In order to model and investigate the adoption characteristics of a system involving multiple competing technology products within an agent-based framework, we consider a social network populated with technology consumers. The social network is represented using a scale-free network [Barabási and Bonabeau (2003)]. The nodes of the network represent individual consumers while the links between them symbolize communication channels. We adopt a single fixed network structure because we are not interested in quantifying the effect of network structure on adoption characteristics. The size of the social network can be defined by the user using the *population* variable. A representative social network is shown in Figure 5.1 while state variables are defined in Table 5.1

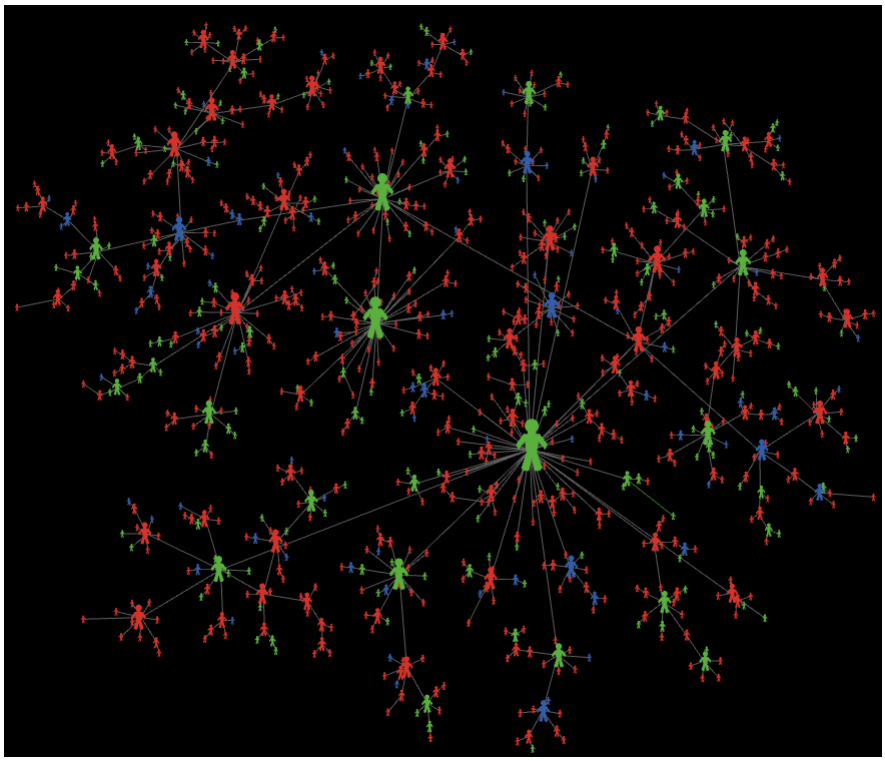

Figure 5.1: Example of a scale-free network. Each node represents a consumer that owns product A (red), product B (green) or product C (blue)

Table 5.1: State variables for modeling environment

Variable	Range of parameter	Units
population	2000	No. of consumers
B-initial and C-initial	0 to 1	Percentage of population

Table 5.2: State variables for technology products

Variable	Range of parameter	Units
price	80-110	$
speed	2.33-3.33	Ghz
cache	{2,4,6}	M
switching cost	0-100	$

Agents

Technology products are characterized by n attributes (n = *1,2,..*). For semiconductor processors, features could include *speed* (Ghz), *cache* (MB) etc. Agents adopting newer product may need to bear additional upgrade costs in addition to the *price* of the processor. For example, newer processors may require a new socket type, and therefore, a new motherboard. Such upgrade costs are set by the variable *switching cost*. At initialization, the proportion of innovators that adopt the second generation of products can be controlled using the variables *B-initial* and *C-initial*. A summary of the state variables for technology products is given in Table 5.2.

Each agent in our model represents an autonomous decision making entity and hence, its correct definition plays a key role in defining the model. Since agents represent adopters of products, we use the theory of diffusion to as a guide for designing their characteristics Rogers (1995). The theory of diffusion classifies consumer groups into five categories : innovators, early adopters, early majority, late majority and laggards [Rogers (1995)]. A summary of the attributes and characteristics possessed by different consumer groups is provided in Table 5.3. To

Table 5.3: Characteristics of adopters

Adopter categories	% of population	Order of adoption	Risk Threshold	Financial Lucidity	Social interactions	Opinion leadership
Innovators	2.5 %	First	Highest	Highest	Closest contact with other innovators	Low
Early adopters	13.5 %	Second	Less than innovators	Lower than innovators	Close contact with innovators and early majority	Highest
Early majority	34 %	Third	Less than early adopters	Lower than early adopters	Contact with early adopters	Low
Late majority	34 %	Fourth	Low	Low	Contact with others in late majority and early majority	Low
Laggards	16 %	Last	Lowest	Lowest	Contact with only family and close friends	Negligible

provide more concrete tractability in the model, the agents in our model can be operationalized with these categories and their corresponding attributes.

The preference structure of the agents defines their minimum (L_n) and desired (U_n) levels of each product attribute. Based on their preference structure,

Table 5.4: State variables for Agents

Variable	Distribution	Units
utility threshold	$\sim$Normal(μ_1, σ_1)	Ratio
γ_n	$\sim$Normal (μ_2, σ_2)	-
W_n	$\sim$Normal ((μ_3, σ_3)	-
change event	$\sim$Poisson(λ)	Time

agents have the ability to score how attractive a product attribute is using desirability functions [Derringer and Suich (1980)]. The coefficients for the desirability functions for each attribute(γ_n) are also included in the preference structure of the agents. Additionally, it also defines how much weight (W_n)the agent associates with individual product attributes as well as social influence. For instance, some agents may consider price as being the most important decision making factor, whereas, others may base their decisions entirely on the number of neighbors that have adopted the product. Assigning different weights to these agents results in a heterogeneous population of consumers.

Each agent has the ability to gather and assimilate information about product attributes as well as the extent of proliferation of a product in its neighborhood. Agents are given behavioral rules that gives them the ability to generate a utility score for each product.The higher the utility score, the more probable the agent is of choosing that product. Since agents are connected on a virtual social network, they continuously communicate their utility scores to their network neighbors. This mimics a feedback mechanism which allows agents to persuade other agents to adopt a product other than the one they bought. Agent *i* adopts the new product the utility score of product *k* is higher than its pre-assigned threshold (*utility threshold*) and if it has sufficient *budget* to buy the product.

Timescale

For this study, we are interested in technology switching from a first generation product to a second generation product. As such, we assume that the agents only switch once during this time window. Furthermore, we assume that the rate at which customers consider switching to the second generation of products is constant over the time window under consideration. This assumption lets us define change events using a Poisson distribution with a user defined parameter (*change event*). Agents evaluate products and compute utility scores on a continuous, asynchronous basis.

5.4 Process overview and scheduling

A brief outline of the process is given below. Specific sub-processes are explained in more detail later:

1. At initialization, we assume that a majority of the population owns product A. A small proportion of the population can be seeded with products B and C. These products are assigned to those agents with the lowest utility threshold.

2. At each time increment agents compute utility scores for each product B and C.

3. A change event is generated for agents according to the internal time clock operationalized by the Poisson distribution.

4. If the ratio of utility for a product to current technology is greater than the agents threshold and the new product fits the agents budget, the agent adopts the new product with probability proportional to the utility score of the product.

5. At each time interval, each agent's budget increases by a random amount if the agent does not buy a new technology. Once the agent buys the new technology, its budget is set to zero.

5.5 Decision Engine

Here, we discuss three sub-processes in more detail. Specifically, we will focus on the elements of the model that allow the agents to gauge social influence, evaluate products using utility scores and finally the adopt a product. Together, these processes make up the decision making engine of the agents.

1. Computing social influence: The decision to buy a particular technology product can be influenced by the agent's neighbors. Here, neighbors are defined as directly connected agents. We assume that agents are more likely to buy a particular technology if a large proportion of their neighbors have already adopted it. We allow all agents to update their valuation of all products even though they may have adopted a particular product. This is important since pricing decisions and other external stimuli can make certain products more attractive at a later point in time. Hence, even though an agent may have adopted one product, it may in fact recommend some other product to its neighbor. Social influence index, defined for each agent, is computed as the average utility for technology K in its neighborhood times the proportion of neighbors that have already adopted technology K.

2. Utility score: The process of computing product utility scores starts with evaluating the desirability of each product attribute with regards to the agent's preference structure. Then, the utility score of the product is computed by taking the weighted average of each desirability index and the social influence index. Desirability indices are computed as follows: Agents score each

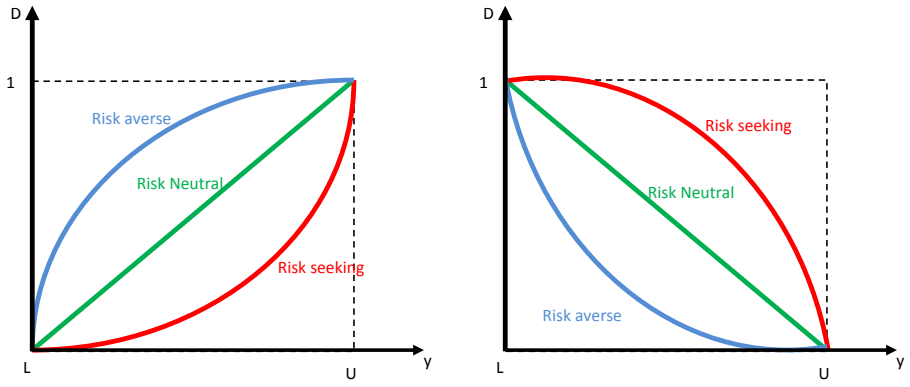

Figure 5.2: Desirability functions showing (left) risk-averse agents ($\gamma < 1$) and risk-seeking agents ($\gamma > 1$) for targeting maximum, (right) risk-averse agents ($\gamma > 1$) and risk-seeking agents ($\gamma < 1$) for targeting minimum

product attribute between 0 - 1 using non-linear desirability functions. For attributes where higher value makes the product more desirable (speed, cache), agents use a function of the form given in Equation 5.1, whereas when lower values of an attribute makes the product more desirable to the agent, Equation 5.2 is used.

$$D_n^k = \left(\frac{Y_n^k - L_n}{U_n - L_n}\right)^{\gamma^n}, \tag{5.1}$$

$$D_n^k = \left(\frac{U_n - Y_n^k}{U_n - L_n}\right)^{\gamma^n}, \tag{5.2}$$

where, D_n^k is the desirability index for attribute n of product k, Y_n^k is the current value of attribute n of product k and U_n and L_n define the upper and lower preference values for attribute n. The γ parameter, also known as the desirability coefficient defines the profile of the function. Figure 5.2 shows the influence of the γ parameter on the shape of the desirability function.

For each agent, the utility of technology K is computed as the weighted average of the desirabilities of the product attributes and its local-attractiveness as shown in Equation 5.3. In addition to having different desirability functions,

Table 5.5: Discrete choice element of the decision engine. Cells represent adoption probability for technologies (B,C).

	$\frac{U_B}{U_A}$ >threshold & $\frac{U_C}{U_A}$ <threshold	$\frac{U_C}{U_A}$ >threshold & $\frac{U_B}{U_A}$ <threshold	$\frac{U_B}{U_A}$ >threshold & $\frac{U_C}{U_A}$ >threshold	
Budget>Price-B & Budget<Price-C	$\frac{e^{\beta U_B}}{e^{\beta U_B}+e^{\beta U_C}}$, 0	0,0	0,0	
Budget>Price-C & Budget<Price-B	0,0	0, $\frac{e^{\beta U_C}}{e^{\beta U_B}+e^{\beta U_C}}$	0,0	
Budget>Price-B & Budget > Price-C	0,0	0,0	$\frac{e^{\beta U_B}}{e^{\beta U_B}+e^{\beta U_C}}$, 1 $Prob_B$	–

agents can assign different importance (weight, W_n) to different attributes that make up the utility function. Thus, we assume an additive utility function.

$$U_K = \frac{\sum_n W_n D_{k_n}}{\sum_n W_n}, \tag{5.3}$$

3. Technology switching: Each agent is assigned a different value for its utility threshold. The utility threshold is a measure of how much better the newer technology should be compared to its existing technology for the agent to consider switching. After a change event is triggered, the agent checks its ratio of utility for technology K to that of its currently adopted technology. If this ratio exceeds the utility threshold, and the agent has enough money to buy the new technology, then it considers buying the new technology. For illustrative purposes, let us consider that all agents currently own technology A and they have a choice to switch to either technology B or C. A set of different decision environments arise at this point. Each cell of Table 5.5 shows the probability with which an agent will adopt technology (B,C).

   In general, the probability of adopting a technology K is given by

$$Prob_k = \frac{e^{\beta U_k}}{e^{\beta U_k} + e^{\beta U_{k'}}}, \tag{5.4}$$

where,k' represents the set of competing products except product k and $beta$ represents how sensitive the agents are to differences in the product utility scores. The higher the β, the more sensitive agents to the dynamics of technology attributes.

5.6 Data Mining Framework

An advantage of ABMs is the capability to explore complex scenarios that may consist of a large number of attributes. Furthermore, many attributes of mixed type (numerical and categorical), with potentially different distributions, units and/or scales of measurements can be expected. The relationships between model outputs and inputs might be nonlinear, with interaction effects, and inputs may differ substantially in their impact on outputs. In addition, models developed from actual data may need to handle outliers and missing values in the inputs. Consequently, these characteristics are challenges to an analysis method used to summarize the results of an ABM. Although an analysis method might be selected for a specific ABM and application, we use an analysis framework that can handle the complexity of the ABM scenarios more generally and produce important summaries to interpret the ABM.

In our framework, summaries are generated from predictive models that learn the relationship between an output and inputs from the data generated by the ABM. Therefore, good predictive performance, in spite of the challenging characteristics of the data, is important. For this role we use decision-tree ensembles [Breiman (2001), Tuv et al. (2009)]. A decision tree applies a recursive partitioning to the rows of data in order to obtain subsets in which the outputs for the rows in a subset are similar [Quinlan (1993)]. Similarity (referred to as purity) can be measured with a Gini score [Breiman et al. (1984)] for categorical outputs or with squared error for numerical outputs. The subset models are expressive and handle

nonlinearities and interactive effects. The partitioning can easily handle numerical or categorical inputs, the model is invariant to attribute scales or units and it is insensitive to outliers and missing values. Furthermore, the purity improvement from an attribute used to create a partition provides an intrinsic measure of attribute importance to the output [Breiman et al. (1984)].

However, a single tree is constructed with a greedy algorithm that can lead to an unstable model (small changes to the data can alter the model substantially). Also, predictions can change abruptly with small changes to the input (as an input changes from one subset to another). Ensembles generate a large number of trees (our results use hundreds) and average the predictions to provide more stable, smoother predictions. Different types of tree-based ensembles can be generated, but we use a simple method known as random forest (RF) [Breiman (2001)]. The trees are grown from random samples (with replacement) from the original training data. Also, to reduce the correlation between trees (and thereby decrease the variance of the model) only a random subset of attributes is considered for each partition in each tree. The size of this subset is really the only parameter in a RF model and we used the common default setting equal to the square root of the total number of attributes. The ensemble average can substantially improve the predictive performance and many more partitions from numerous trees improve the attribute importance measures.

Because random samples in RF are selected with replacement, some data rows are typically omitted from each sample. These rows are referred to as out-of-bag (OOB) data and such data is useful to evaluate the performance of a model. An estimate of model error from OOB data approximates how the model will perform on new data that is not used for training [Breiman (2001)]. A model for analysis that can handle the challenging characteristics of data from an ABM is not ex-

pected to be directly interpretable. However, the tree ensembles provide a number of tools that are particularly useful for an ABM study. With a large number of inputs, critical tasks are to identify inputs that are important contributors to the model output and to graphically summarize the effects of such contributors. The general problem to identify important contributors to a model is called feature selection and an overview was provided by Guyon and Elisseeff (2003). For our application regarding the adoption behavior of new technologies under different competitive environments, an important goal of the ABM is to learn what input attributes are most important to product success in the market. Consequently, we want to summarize and quantify the impact of input attributes. Feature selection techniques are based on the idea that information content in high-dimensional data is often contained in a small subset of relevant attributes. Modern learners such as tree-based ensembles have proven to be very successful in filtering out irrelevant attributes while preserving the relevant ones [Tuv et al. (2009)]. Consequently, the tree-based ensembles provide robust framework with few parameters, and with useful tools (such as feature selection, intrinsic error estimates) that can handle the complexity of data generated from ABMs.

After important attributes are identified, an important step is to understand the nature of the dependence of the output on these the relevant attributes [Friedman et al. (2001)]. Here, our goal is to identify the values (or range) of the important attributes that drive the most significant changes in the output. For example, we are interested in attribute values that are associated with a higher adoption of a particular product. Visualization of the predicted function $f(x)$ over the input space is a powerful tool to interpret the model as it provides a summary of the dependency of the output on the values of the inputs. However, direct visualization is limited by the number of attributes that can be handled successfully. Instead, we apply the

graphical summary referred to as partial dependence plots to add interpretability to our model [Friedman (2001)]. Partial dependence functions represent the effect of the attribute on the output after accounting for the average effects of other attributes. Plotting the partial dependence of $f(x)$ on its most relevant attributes can reveal how the output behaves in different regions. We use such plots to interpret the results from our tree ensembles.

5.7 Scenario Analysis Examples

Product substitutions are fairly frequent in technology markets. Since the life cycle of these products are relatively short (6 - 18 months), substitution decisions need to be made every quarter. In making product substitution decisions, it is important for analyst to understand how various product attributes such as price, switching cost, speed, cache, etc. influence the adoption success of each product in the market. In the following sections we demonstrate how the ABM and data mining framework can be utilized to gain insight into the adoption process by using some representative scenarios.

Effect of Price

Consider the following representative product substitution scenario: A semiconductor chip manufacturer currently has two technology products in the 80 $ - 110 $ price range. Product A, which is the older of the two products, currently dominates the market share in this price group. At the beginning of the simulation, a small percentage of innovators and early adopters own product B. The company now decides to terminate the production of product A and launch a new product - product C. Furthermore, products A and B are compatible with the same socket type, i.e., they can be interchanged on the same motherboard. However, product C requires a different socket type. Hence, a switching cost, is associated with buying product

Table 5.6: Simulated market conditions

Factor	Levels
Speed-A	2.66 Ghz
Speed-B	2.93 Ghz
Speed-C	3.13 Ghz
Cache-A	2 M
Cache-B	2 M
Cache-C	4 M

Table 5.7: Parameter setting for analyzing the effect of product pricing

Category	Variable	Value
Environment	Population	1500
Environment	B-initial	20 %
Environment	C-initial	5 %
Agent	utility threshold	∼Normal(1.3, 0.26)
Agent	γ_n	∼Normal (1, 0.2)
Agent	W_n	∼Normal (1, 0.2)
Agent	change event	∼Poisson(50)
Agents	Initial budget	60 $

C. Customers who currently own product A, are now faced with three options: Stay with product A, switch to product B or switch to product C.

Since price is one of the most important drivers of product demand, there is strong interest in understanding its effect on the adoption of product C. To analyze the effect of introductory prices, we assumed that the technological attributes of the three products were fixed. The speed and cache of the products were fixed as per the settings summarized in Table 5.6. Other simulation parameters that were held constant during the runs are summarized in Table 5.7. The price for each product was varied starting from 80 $ to 110 $ in increments of 5 $. A total of five replicates were run at each combination of the prices. After each run, we noted the maximum share of the market (as a proportion) captured by product C. This was the output of interest. A RF model was built to analyze the effects of pricing.

The RF model reduced the prediction error from a base error (that used the mean output as the model) from 0.219 to 0.029. We note that this estimate is based

on OOB data, and as discussed previously, this generally provides a good estimate of generalization error. To interpret the results, we discuss the dependency plots of the price attributes. Figure 5.3 shows the dependency of the adoption success of product C as a function of the prices on two products, after accounting for the price of the third product. The top row shows surface plots and the vertical axis is the proportion of the market share captured by product C. The bottom row of figures show the corresponding contour plots.

The figures on the left shows the price effects of A and C on the market share of C. Note that the share of C is relatively insensitive to the price of A, especially over the range from approximately $ 85 to $ 100. Instead, the price of C itself has a much greater impact on its share. Although the effect of the price of C might be expected, the limited sensitivity to As price over a wide range is interesting.

The middle figures show the price effects of B and C on the market share of C. Here Bs price modifies the effect of Cs price. With a low price for B of $ 85, C cannot obtain a market share greater than approximately 20%. However, when B is at a higher price, say $ 100, the effect of Cs price is much more pronounced. At this price for B, the surface plot for C raises quickly as Cs price is reduced. This allows for some interesting demand shaping. For example, with B priced at $ 100 the company can substantially change the market share of C through its price, possibly to meet sales targets or adjust to inventory status.

The figures on the right show the price effects of A and B on the market share of C. Similar to the middle figure, when the price of B is low, say $ 85, it is difficult for C to obtain substantial market share. When B's price is higher A's price has a much greater effect. From the contour plot, even for B as high as $ 90 the effect of A's price on C's share is small. It is only when B's price is approximately $ 100 that we notice a strong effect of A's price on C's share. Otherwise the effect

of A's price is modest. As expected, with both A and B priced high, a substantial increase in C's share is observed.

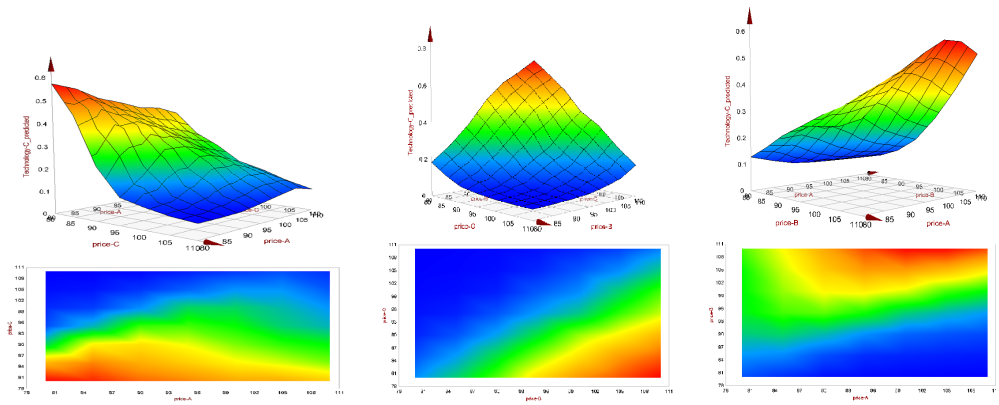

Figure 5.3: Dependency plot showing the relative price effects on the adoption of product C. (Left) Price-C vs Price A, (Middle) Price-C vs Price-A, (Right) Price-B vs Price A.

Effect of simultaneous new product introductions

In the first scenario we focused on the price of the three products. While price is a primary driver of demand, other attributes such as the speed and cache of the processor can influence the adoption success of competing products. For a more comprehensive analysis, we expand the scope of the first scenario by varying the technological attributes along with product prices. The objective here is to not only quantify how price influences buying decisions, but also to include product attributes in the decision domain.

Consider a scenario where product A is already in the market priced at $ 95, with 3.0 Ghz speed and 4 M cache. Using the agent based model, we simulate situations in which two new products, B and C, are simultaneously introduced in the market. Each run of the simulation corresponds to a unique product mix obtained by changing product attributes of products B and C (Table 5.8). A full factorial design of these different product settings leads to 1458 runs per replicate. We ran five

Table 5.8: Experimental conditions under which new products were introduced

Factor	Levels
Price-B and Price-C	$ {80, 95, 110}
Speed-B and Speed-C	{2.6,3.0,3.4} Ghz
Cache-B and Cache-C	{3,4,5} M
Switching cost-C	$ {10,30}

replicates of this experiment and recorded the maximum market share that products A, B and C achieved at steady state.

Given the results from the experiment, an RF model is built for the market share of each product separately. The models are used to interpret what-if scenarios. We discuss some of the common questions related to product introduction and how they can be answered using partial dependency plots.

The first question relates to the incumbent product. Getting a better understanding of how much of the market will be retained by the incumbent product under different mixes of the new products is critical because it drives inventory decisions. Companies usually struggle with product phase-outs since it is not always clear how much stock to hold for the incumbent product during its transitional phase. Therefore, we would like to understand scenarios under which product A continues to hold a sizeable share of the market even after the introduction of the new products. On the flip side, we would also like to understand what situations lead to a near complete take over of the market by the new products. These insights are important to improve inventory management and demand shaping point of view.

The RF model built for product A yields a 78% reduction in base error (OOB). The variable importance scores obtained from the feature selection algorithm are shown in Table 5.9. From this, it is evident that over the range of experimentation, different factors affect the predictive power of the model with varying scales of importance. For example, speed of C is much more of a defining factor

Table 5.9: Variable importance scores for Product A

Factor	Importance (% of highest)
Speed-B	100 %
Price-B	77.54 %
Speed-C	39.50 %
Cache-B	36.73 %
Price-C	32.19 %
Cache-C	18.77 %
Cache-C	9.42 %

compared to its switching cost. Consequently, over the range of interest studied, switching cost is very much a secondary factor and models can effectively focus of the other attributes.

The partial dependency plots generated using this model are shown in Figure 5.4. The plot on the left is for a situation where products B and C have the most advantage over product A. That is, for the new products B and C, price and switching cost are at the lowest setting, and speed and cache are at the highest setting, respectively. The figure provides the market share of product A as a function of the prices of B and C. Since pricing decisions are often the last product related decisions to be made, we display these price variables for the two new products on the two axes of the plot. The first observation is that B's price has a more pronounced effect than that of C's. For example, if the price of C is $100 the effect of a change of B;s price is greater than the the other case (when the price of B is $100 and C's price is changed . This is consistent with the inference drawn from the variable importance scores above and is possibly because C has an additional switching cost associated with it. When either new product is priced at its lowest setting ($80), we see that product A retains only a small portion of the market (17%). That is, either new product is capable of capturing substantial market at $80. Interestingly, when one product is priced at $ 110 and the other is at $ 80, we can see that product A retains a relatively low percentage of the market (less than 25%). Only when both

new products are priced at the highest setting ($ 110), can A retain a majority of the market (approximately 68%).

The plot on the right of Figure 5.4 represents yet another product mix where the technological advantage of the new products is not as lopsided. For product B, the speed and cache are held at 3.4 Ghz and 4 M, respectively, That is, B has higher speed, but the same cache as product A. For product C, the speed, cache and switching cost are held at 3.0 Ghz, 5 M and $10, respectively, That is, it has the same speed as product A, but higher cache. It can be seen that for the two new products to occupy more than 30 % of the market, it is necessary to price product B lower than $ 85. Interestingly, product A is relatively insensitive to the price of C once the price of B is set. That is, little interaction between the prices of B and C are observed. Any other point on these graphs represent an alternative pricing solution, and thus a way to effectively shape demand according to production schedules and build plans.

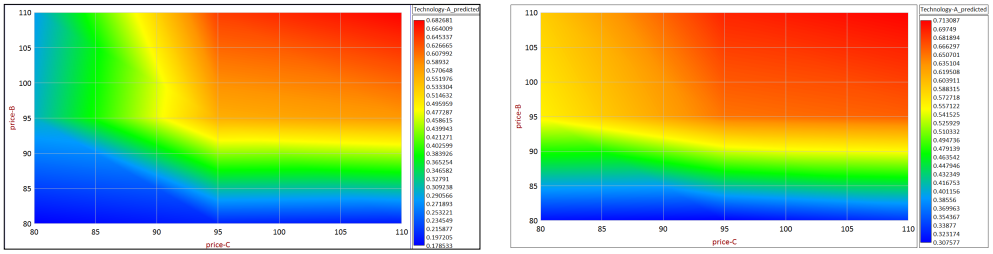

Figure 5.4: Dependency plot showing the effect of new product introduction on the adoption of product A.

We can use the same strategy to gain more insight into the adoption of each of the two new products. To do this, we can build a separate RF model for product C and draw inferences from the variable importance scores and partial dependency

Table 5.10: Variable importance scores for Product C

Factor	Importance (% of highest)
Price-C	100 %
Speed-C	70.50 %
Cache-C	30.17 %
Price-B	22.27 %
Switching cost-C	22.13 %
Speed-B	8.18 %
Cache-B	2.93 %

plots for fixed settings of product B. To demonstrate, let us fix the price, speed and cache of product B at $ 90, 3.4 Ghz and 4 M respectively. For product C, we fix its switching cost at $ 10 and cache at 5 M. The RF model reduces the base error by 76 % (OOB). We can see from the variable importance scores that speed and price of product C are the two most important variables (Table 5.10). Hence, in Figure 5.5 we show the dependency plot with these two variables as the axes. It is evident that above $ 95, product C achieves very little penetration in the market. Similarly, irrespectively of its cost, a product with less than 2.8 Ghz captures very little market share. The lower right corner of the plot, represented by high speed and low price for C, is the region of high penetration. When C is priced at 80 $ and has a speed of 3.4 Ghz, we see that it can potentially dominate the market with greater than 60% of the share. Thus, by using Figures 5.4 and 5.5 in tandem, one can explore different product mix strategies that are consistent with inventory and sales targets.

5.8 Conclusion

Analyzing the effect of price and product attributes for technology products with empirical models has been a challenge due to their short life cycles. The limited amount of time that each product remains relevant in the market reduces the available data. But it is still important to gain a deeper understanding of what competitive scenarios lead to the success or failure of product diffusion. The interplay of

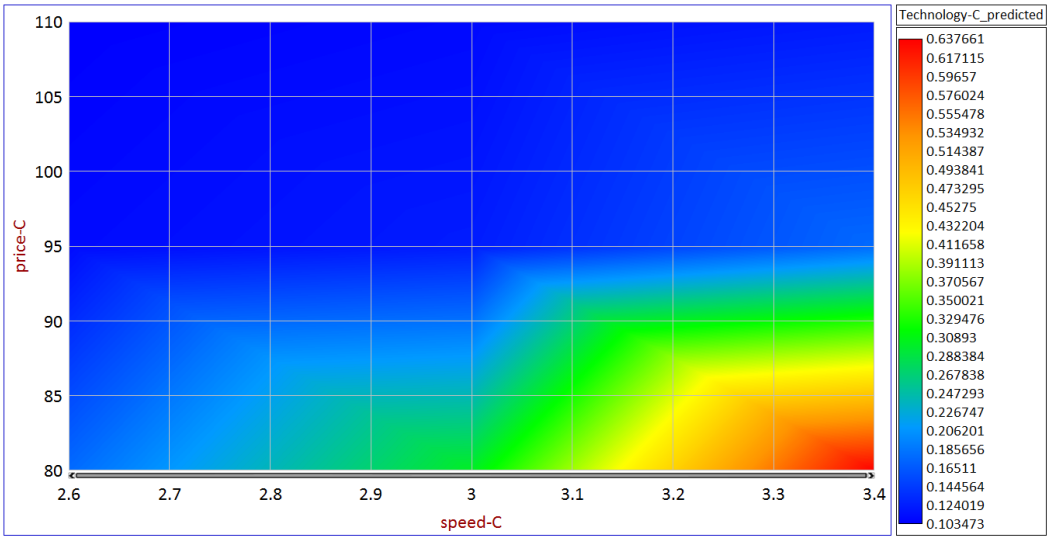

Figure 5.5: Effect of price and speed of product C on its adoption success

a number of different design attributes of existing and new products, along with prices, determine the adoption success (or failure) within a short time horizon.

Here we present an agent-based model of product diffusion as an evolutionary process. The simulation model was built to study how interactions between consumers and competition from other products can lead to different adoption characteristics. Due to the flexibility offered by ABMs, we were able to combine and integrate the theory of diffusion, utility theory and discrete choice models into the behavioral engine of the agents. Furthermore, we demonstrated how the agent-based models can be used as a test bed for simulating different market situations and therefore form the core of a scenario analysis platform.

We used modern data mining methods to extract the information hidden in the data collected from multiple runs and replicates of the simulation engine for specific scenario settings. The data obtained from ABMs in usually non-linear in nature and we showed how decision-tree ensemble models can be used to successfully capture these complex relationships. Furthermore, we were able to score the impact of each product attribute on the adoption behavior of the competing products using feature selection methods from the ensemble methods. Through partial dependency plots, we demonstrated how alternate pricing and product attribute strategies could be explored to achieve market share targets for each product. This is a valuable tool since it summarizes a complex framework of ABM and data mining models through simple, intuitive and visual graphs that analysts can interpret.

With short life cycles, many products may be introduced into a complex mix of existing products within a single quarter. Furthermore, rapid technology changes can change the product attributes substantially. Consequently, it is difficult to use traditional models with historical data to select attributes and prices of new products (or adjust the prices of the existing products). The essentially new product environment encountered for most new product introductions challenges the fidelity of previously built models, and limits the generalization of previously learned guidelines for attributes or pricing. Consequently, rather than a traditional model, it is useful to have a methodology that is based on consumer tendencies, but allows for the current (possibly complex) product environment. We showed how an ABM provides the capability to handle such complexities, and that our analysis framework can be a valuable addition for decision makers to interpret and act upon the results from the ABM.

Chapter 6

CONCLUSIONS AND FUTURE WORK

Analytical tools have become an integral part of modern supply chain management practices. With advances in measurement, tracking and storage technologies, large, complex data sets are becoming the norm in modern supply chain operations. Traditional supply chain health metrics are supplemented with detailed transactional information (such as product life cycle, pricing, rebates, shipment route, sales patterns etc.) as well as information on economic, regulatory, environmental and competitive scenarios. Additionally, decision events such as those pertaining to product pricing, new product introduction, discontinuation etc., are also captured in these extensive data sets. Given such rich history, it is prudent to look for patterns and relationships hidden in these data sets to enable better decisions.

Traditional methods for scenario analysis that were designed for data sets with modest dimensions struggle in the new ”messy” data environment. Modern data sets are fraught with missing values, outliers and exhibit strong nonlinear relationships. Additionally, the product diffusion and substitution processes result in incomplete, non-homogeneous data sets. As a result, analytical toolkits should be capable of comprehending the complexities of disparate data sets and providing accurate qualitative as well as quantitative feedback on the impact of business scenarios. In this research, we emphasize the need for end-to-end modeling systems that capture the entire life-cycle of the analytical process; from knowledge representation to model summaries. Towards that end, we explored, extended and improved modeling techniques that are more tuned for modern supply chain analytics.

We proposed a new scenario analysis framework, labelled as *Scenario Trees*. This framework included a data shaping element that enables us to represent dis-

crete scenario information and past decision events as well as continuous predictors in a form that is amenable for modern machine learning algorithms. Delayed effects were captured using lagged variables. Relationships between multiple responses (such as the requirements that market shares sums to one) was incorporated through a structure analogous to logisitic regression. With hundreds of predictors, we used a modern feature selection strategy that enables us to extract a non-redundant, compact subset of features from the data. This is of great practical importance as it allows decision makers to focus attention on a smaller scenario set. We used robust tree-based, supervised learners to characterize the relationship between the predictors and the supply chain response (demand / market share in this case). These methods provide excellent prediction accuracy and are capable of comprehending mixed variables without needing extensive scaling. Also, tree-based ensemble models are robust to outliers and have built in mechanisms to handle missing data. All of these characteristics make them well suited to handle the complexities presented by modern supply chain data. Finally, to aid model understanding and create actionable insights, we summarized the models using partial dependency plots.

While the feature selection methods used in the *Scenario Trees* approach are successful at limiting the decision domain by extracting important predictors, further model simplifications can be achieved by investigating each variable and identifying the actual settings / levels / values for the variable that contribute to importance. Beneficial insights into the model can be gained by grouping variable levels that are insignificant and highlighting important ones. We approached this problem from a feature extraction point of view. One proposed strategy used a simple transformation of the dataset into a binary incidence matrix. This transformation allowed us to leverage existing feature selection methods to extract important feature values from the data. This method was shown to be effective in a simulated ex-

perimental setting. However, it does require additional data storage capacity for the incidence matrix. To alleviate this problem, we used an embedded feature value importance metric within the tree induction algorithm. This new method also showed promising results in a simulated environment, and since it is based on the tree-based ensembles method, it inherits all of its desirable properties.

While the primary motivation behind developing the feature value selection algorithms was to simplify complex models, it is more generally useful for supply chain analysis. High-dimensional supply chain surveillance has traditionally been restricted to a series of control charts that monitor network segments (nodes) using multiple health metrics. In our research, we proposed an approach to identify problems nodes in the network by transforming the surveillance problem to a feature value selection problem, a solution we called *Network Feature Extraction*. We demonstrated how a rich set of transactional and network related attributes could be represented using a case-event view of the supply chain. We then applied the embedded feature value selection method to extract important nodes and important attributes from the network. Furthermore, partial dependency plots were combined with the network feature extraction method to create graphical summaries of these complex interactions.

Empirical scenario analysis works well in situations where there is an abundance of historical data from which patterns can be learnt. However, scenarios such as those related to new product introduction have little to no historical information to learn from. For such situations, we presented an agent-based model to simulate multiple product diffusion scenarios. Given its flexibility, we demonstrated how various different theories such as the theory of diffusion, utility theory and discrete choice models could be integrated into the simulation model. Additionally, we used expressive machine learning algorithms to handle complex, non-linear, and

interactive effects to identify important inputs that contribute to the model and to graphically summarize their effects. Finally, using realistic case studies, we illustrated how our analysis framework can be a valuable addition for decision makers to interpret and act upon the results from the agent-based models.

Future Work

Expert-based systems for scenario analysis tend to rely almost entirely on experts to generate decision rules. Future work should focus on developing a hybrid approach that integrates empirical learners such as the tree models with expert generated rules. For successful implementation of any scenario analysis method, it is important to incorporate expert judgement into the framework. Expert inputs, especially for scenarios that are hard to quantify and measure, often add value to the analysis.

A sensitivity analysis for the Scenario Trees is required to demonstrate its efficacy in modeling effects of varying magnitude. This should ideally be done in a simulated environment where the scenario effects can be constructed and built into the simulation design. We can then assess the performance of the analysis framework against known ground truth. Furthermore, this simulation setup can be used to compare the proposed method with existing alternatives for empirical scenario analysis.

Chapter 3 provides a preliminary sensitivity analysis for the three feature value selection strategies. This work could benefit from a more extensive sensitivity analysis by varying the simulation parameters over a wider range.

Supply chain surveillance is another rich area for research. This work has primarily focused on converting the surveillance problem into a feature value selection problem. For this approach to gain acceptance, it is important to compare its

performance with existing monitoring approaches such as those from the process control literature.

Identification and localization of faults in the network using an approach such as the one presented in Chapter 4 is just one component of a larger monitoring system. We envision a monitoring system that comprises of a rich set of monitoring rules that can be learnt by integrating the NFE approach with rule-based classifiers. The rules can act as probes that can be used to examine the status of the network and furthermore signal the occurrence of specific scenario events. A scenario playbook which documents what steps should be taken given certain signals from the monitoring system can then be developed to help practitioners take appropriate actions.

REFERENCES

Arulselvan, A., C. Commander, L. Elefteriadou, and P. Pardalos (2009). Detecting critical nodes in sparse graphs. *Computers & Operations Research 36*(7), 2193–2200.

Athanasiadis, I., A. Mentes, P. Mitkas, and Y. Mylopoulos (2005). A hybrid agent-based model for estimating residential water demand. *Simulation 81*(3), 175.

Barabasi, A. and E. Bonabeau (2003). Scale-free networks. *Scientific American 288*(5), 50–59.

Bass, F. (1969). A new product growth for model consumer durables. *Management Science 12*(5), 215–227.

Bass, F., T. Krishnan, and D. Jain (1994). Why the Bass model fits without decision variables. *Marketing Science 13*(3), 203–223.

Bavelas, A. (1948). A mathematical model for group structures. *Human organization 7*(3), 16–30.

Bitran, G. and S. Mondschein (1997). Periodic pricing of seasonal products in retailing. *Management Science 43*(1), 64–79.

Bohlmann, J., R. Calantone, and M. Zhao (2010). The Effects of Market Network Heterogeneity on Innovation Diffusion: An Agent-Based Modeling Approach. *Journal of Product Innovation Management 27*(5), 741–760.

Borgatti, S. (2006). Identifying sets of key players in a social network. *Computational & Mathematical Organization Theory 12*(1), 21–34.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Breiman, L., J. Friedman, R. Olshen, and C. Stone (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Davila, S. (2010). *Public Health Surveillance in High-Dimensions with Supervised Learning*. Ph. D. thesis, Arizona State University.

Deaton, A. and J. Muellbauer (1980). An almost ideal demand system. *American Economic Review 70*(3), 312–26.

Delre, S.,W. Jager, T. Bijmolt, and M. Janssen (2007). Targeting and timing promotional activities: An agent-based model for the takeoff of new products. *Journal of Business Research 60*(8), 826–835.

Delre, S., W. Jager, and M. Janssen (2007). Diffusion dynamics in small-world networks with heterogeneous consumers. *Computational & Mathematical Organization Theory 13*(2), 185–202.

Derringer, G. and R. Suich (1980). Simultaneous optimization of several response variables. *Journal of quality technology 12*(4), 214–219.

Dheenadayalu, Y., B. Wolshon, and C. Wilmot (2004). Analysis of link capacity estimation methods for urban planning models. *Journal of transportation engineering 130*, 268–275.

Fisher, J. and R. Pry (1972). A simple substitution model of technological change. *Technological forecasting and social change 3*, 75–88.

Freeman, L. (1979). Centrality in social networks conceptual clarification. *Social networks 1*(3), 215–239.

Friedman, J. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J. and J. Meulman (2003). Multiple additive regression trees with application in epidemiology. *Statistics in medicine 22*(9), 1365–1381.

Friedman, J., R. Tibshirani, and T. Hastie (2001). *The elements of statistical learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Gallego, G. and G. Van Ryzin (1994). Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science 40*(8), 999–1020.

Gallego, G. and G. Van Ryzin (1997). A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research 45*(1), 24–41.

Garcia, R. (2005). Uses of Agent-Based Modeling in Innovation/New Product Development Research. *Journal of Product Innovation Management 22*(5), 380–398.

Gatignon, H. and T. S. Robertson (1985). A propositional inventory for new diffusion research. *The Journal of Consumer Research 11*(4), 849–867.

Green, R. and J. Alston (1990). Elasticities in aids models. *American Journal of Agricultural Economics 72*(2), 442–445.

Guardiola, X., A. Diaz-Guilera, C. Perez, A. Arenas, and M. Llas (2002). Modeling diffusion of innovations in a social network. *Physical Review E 66*(2), 26121.

Guyon, I. and A. Elisseeff (2003, March). An introduction to variable and feature selection. *Journal of Machine Learning Research 3*, 1157–1182.

Guyon, I., J. Weston, S. Barnhill, and V. Vapnik (2002). Gene selection for cancer classification using support vector machines. *Machine learning 46*(1), 389–422.

Hall, M. (1999). Correlation-*based feature selection for machine learning*. Ph. D. thesis, The University of Waikato.

Hamilton, D., F. Roques, and W. Nuttall (2009). Agent Based Simulation of Technology Adoption.

Hochberg, Y. and A. Tamhane (1987). *Multiple comparison procedures*, Volume 82. Wiley Online Library.

Hosmer, D. and S. Lemeshow (2000). *Applied logistic regression*. New York: Wiley-Interscience.

Jager, W. (2006). Simulating consumer behaviour: A perspective. *Environmental Policy and Modeling in Evolutionary Economics; Faber, A.; Frenken, K.; Idenburg, AM, Eds.; Netherlands Environmental Assessment Agency*, 1–27.

Kincaid, W. and D. Darling (1963). An inventory pricing problem. *Journal of Mathematical Analysis and Applications 7*(2), 183–208.

Kohavi, R. and G. John (1997). Wrappers for feature subset selection. *Artificial intelligence 97*(1-2), 273–324.

Ma, T. and Y. Nakamori (2005). Agent-based modeling on technological innovation as an evolutionary process. *European Journal of Operational Research 166*(3), 741–755.

Ma, T. and Y. Nakamori (2009). Modeling technological change in energy systems-From optimization to agent-based modeling. *Energy 34*(7), 873–879.

Meixell, M. and S. Wu (2002). Scenario analysis of demand in a technology market using leading indicators. *Semiconductor Manufacturing, IEEE Transactions on 14*(1), 65–75.

Monahan, G., N. Petruzzi, andW. Zhao (2004). The dynamic pricing problem from a newsvendor's perspective. *Manufacturing and Service Operations Management 6*(1), 73—91.

Montanari, A. and A. Saberi (2010). The spread of innovations in social networks. *Proceedings of the National Academy of Sciences 107*(47), 20196.

North, M., C. Macal, J. Aubin, P. Thimmapuram, M. Bragen, J. Hahn, J. Karr, N. Brigham, M. Lacy, and D. Hampton (2010). Multiscale agent-based consumer market modeling. *Complexity 15*(5), 37–47.

Quinlan, J. (1993). *C4. 5: Programs for machine learning*. Morgan Kaufmann.

Rahmandad, H. and J. Sterman (2007). Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. MIT Sloan School of Management,Forthcoming Management Science.

Rich, E. and K. Knight (1991). *Introduction to artificial intelligence*. McGraw-Hill.

Rogers, E. (1995). *Diffusion of innovations*. New York: Free Press.

Scott, D., D. Novak, L. Aultman-Hall, and F. Guo (2006). Network robustness index: A new method for identifying critical links and evaluating the performance of transportation networks. *Journal of Transport Geography 14*(3), 215–227.

Shinde, A., and G. Runger (2012). Performance characterization of different feature value selection strategies. Discussion paper, Arizona State University.

Tanny, S. and N. Derzko (1988). Innovators and imitators in innovation diffusion modelling. *Journal of Forecasting 7*(4), 225–234.

Tuv, E., A. Borisov, G. Runger, and K. Torkkola (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *The Journal of Machine Learning Research 10*, 1341–1366.

Tuv, E., A. Borisov, and K. Torkkola (2006). Feature selection using ensemble based ranking against artificial contrasts. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, 2181–2186. IEEE.

West, M. and J. Harrison (1997). *Bayesian forecasting and dynamic models*. New York, NY, USA: Springer-Verlag New York, Inc.

Willems, S. P. (2008). Data Set - Real-World Multiechelon Supply Chains Used for Inventory Optimization. *Manufacturing & Service Operations Management 10*(1), 19–23.

Wu, S., B. Aytac, R. Berger, and C. Armbruster (2006). Managing short life-cycle technology products for agere systems. *Interfaces 36*(3), 234—247.

Yelland, P., S. Kim, and R. Stratulate (2010). A bayesian model for sales forecasting at sun microsystems. *Interfaces* 40(2), 118–129.

Zhao, W. and Y. Zheng (2000). Optimal dynamic pricing for perishable assets with non-homogeneous demand. *Management Science 46*(3), 375–388.