2012

A Comparative Analysis of Machine Learning Techniques For Foreclosure Prediction

Dexter Randell Brown
Nova Southeastern University, msndex@msn.com

A Comparative Analysis of Machine Learning Techniques For Foreclosure Prediction

By

Dexter R. Brown

A dissertation report paper submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
In
Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University
2012

We hereby certify that this dissertation, submitted by Dexter R. Brown, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Sumitra Mukherjee, Ph.D. Chairperson of Dissertation Committee	Date
Mike Laszlo, Ph.D. Dissertation Committee Member	Date
Junping Sun, Ph.D. Dissertation Committee Member	Date

Approved:

Eric Ackerman, Ph.D. Dean, Graduate School of Computer and Information Sciences	Date

Graduate School of Computer and Information Sciences
Nova Southeastern University
2012

Abstract

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

**A Comparative Analysis of Machine Learning Techniques
For Foreclosure Prediction**

By
Dexter R. Brown

January 2012

The current decline in the U.S. economy was accompanied by an increase in foreclosure rates starting in 2007. Though the earliest figures for 2009 - 2010 indicate a significant decrease, foreclosure of homes in the U.S. is still at an alarming level (Gutierrez, 2009a). Recent research at the University of Michigan suggested that many foreclosures could have been averted had there been a predictive system that did not only rely on credit scores and loan-to-value ratios (DeGroat, 2009). Furthermore, Grover, Smith & Todd (2008) contend that foreclosure prediction can enhance the efficiency of foreclosure mitigation by facilitating the allocation of resources to areas where predicted foreclosure rates will be high.

The primary goal of this dissertation was to develop a foreclosure prediction model that builds upon established bankruptcy and credit scoring models. The study utilized and compared the predictive accuracy of three supervised machine learning (ML) techniques when applied to mortgage data. The selected ML techniques were:

- ML1. Classification Trees
- ML2. Support Vector Machines (SVM)
- ML3. Genetic Programming

The data used for the study is comprised of mortgage data, demographic metrics and certain macro-economic indicators that are available at the time of the inception of the loan.

The hypothesis of the study was based on the assumption that foreclosure rates, and associated actions, are dependent on critical demographic (age, gender), economic (per capita income, inflation) and regional variables (predatory lending, unemployment index). The task of the machine learning techniques was to identify a function that well approximates the relationship between these explanatory variables and the binary outcome of interest (mortgage status in +3 years from inception).

The predictive accuracy of ML1 through ML3 was significantly better than expected given the size of the recordset (1000) and the number of input variables (~110). Each ML technique achieved classification accuracy better than 75%, with ML3 scoring in the upper 90s. Given such high scores, it was concluded that the hypothesis was satisfied and that ML techniques are suitable for prediction tasks in this problem domain.

Acknowledgments

Thanks to the almighty for granting me the physical and mental fortitude to see this task to completion. All things are possible with His guidance, and the love and support of family. I owe everything that I am to the discipline and tenacity imparted to me by my dear mother Grace; the world of thanks to her. Much appreciation to my brothers Martin and Raymond and sister Denise, for all the support and encouragement they gave me since childhood.

Very special thanks to Dr. Sumitra Mukherjee who made this journey an enjoyable and stimulating exercise. If I had to do this once more, Dr. Mukherjee will again be my choice for dissertation chair. I must also acknowledge my wonderful friends Ryan ‘Piccard’ Russon and Duane ‘DeGlove’ Dudley for all the advice they shared with me. Finally, thanks to the other members of my dissertation committee, Dr. Sun and Dr. Laszlo, and all the faculty and staff at the GSCIS with whom I interacted over the past four years.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Introduction

This study focused on building on the existing literature in order to develop an improved method for predicting the performance of residential mortgages within a period of three years from contract inception. The prediction task was treated as a binary classification problem where mortgage performance was limited to 'Status Quo' or 'Foreclosure'. Performance indicators such as 'Refinance' and 'Sell with Profit' are considered for future work. The analysis period was limited to three years because of the dependence on macroeconomic forecasts, which are generally less accurate as the projection point increases. The prediction was based on data acquired from a specialized data vendor.

Problem Statement and Goal

A mortgage is a legal instrument which conveys a lien against property in exchange for securing a loan to purchase said property (Pritchard, 2009). Mortgages are the principal means by which homes are purchased by American families and individuals. The term 'foreclosure' is officially defined by Merriam-Webster as "a legal proceeding that bars or extinguishes a mortgagor's right of redeeming a mortgaged estate". In addition to the social and economic hardships experienced by those foreclosed upon, foreclosure also has a negative effect on surrounding homes by reducing the value of nearby properties (Schuetz, Been & Ellen, 2008). According to Schuetz, Been & Ellen, foreclosure also has the potential to reduce local governments' tax bases.

The current decline in the U.S. economy was validated by an increase in foreclosure rates starting in 2007. Approximately one million homes were lost to foreclosure in 2008, up by nearly 63.5% from the 2007 national foreclosure index (Gutierrez, 2009a; Gores, 2009a). Though the earliest figures for 2009 indicate a decrease by approximately 25%, foreclosure of homes in the U.S. is still at an alarming level (Gutierrez, 2009b). The wealthy were not immune to the foreclosure crisis, as even homes valued at a million dollars or more saw double digit foreclosure rate increases in cities such as Ft. Worth, Texas (Brown, 2009). Recent research at the University of Michigan suggested that many foreclosures could have been averted had there been a predictive system that did not only rely on credit scores and loan-to-value ratios (DeGroat, 2009). Also, in recognition of the need for mortgage performance prediction systems, ForeclosureU.com introduced the LoanMod Creator system (ForeclosureU.com, 2009). LoanMod Creator automatically underwrites mortgage modifications based on affordability equations and computes real time success probabilities (ForeclosureU.com). Furthermore, Grover, Smith & Todd (2008) contend that foreclosure prediction can enhance the efficiency of foreclosure mitigation by facilitating the allocation of resources to areas where predicted foreclosure rates will be high.

The primary goal of this dissertation was to develop a foreclosure prediction model that:

1. Builds upon established bankruptcy and credit scoring models.
2. Based the prediction on data that is available at the time of loan inception.
3. Employed supervised machine learning techniques.

A secondary goal was to investigate the relative merits of alternate supervised machine learning techniques for this prediction task. Three supervised machine learning (ML) techniques were contrasted to determine the most accurate predictor. The selected ML techniques are

ML1. Classification Trees

ML2. Support Vector Machines (SVM)

ML3. Genetic Programming.

The following highlights the reasoning behind the choice of the genesis models and technologies:

1. *Bankruptcy Prediction's* primary objective is to identify the variables of importance which can be used to forecast the financial failure of a commercial organization (Altman, 1984). If a homeowner unit can be viewed upon as a financial entity, similar to a commercial organization or going concern (Lensberg, Eilifsen & McKee, 2006), then bankruptcy prediction models may be adaptable at this level as indicators of financial distress. Since book losses usually precede insolvency (Mora et al, 2008), it may be theorized that homeowner financial distress is a potential precursor to foreclosure. Accurate prediction of financial distress can afford homeowners the time to find and implement corrective measures before foreclosure occurs.
2. *Credit Scoring Models* have been the staple of loan determination for several decades. Fair Isaac Corporation is one of the US's leading developers of credit scoring systems (myFICO, 2009). Their numeric

ranking system is referred to as FICO and, like other mainstream models, is based on accounting ratios and regression analysis (Finlay, 2009). Recent research has seen a shift towards the application of ML techniques in credit scoring models (Lee, 2007; Bellotti & Crook, 2009; Abdou, 2009). This shift is recognition that the existing models are inaccurate predictors of borrower default (Finlay). Since credit scoring is an integral part of the mortgage process that is unlikely to change, a cutting edge foreclosure prediction model should include elements of a forward-looking credit scoring system.

3. *ML Techniques* have evolved into the most commonly used analytical and predictive methods utilized in bankruptcy and credit scoring models (Odeh, Koduru, Das, Featherstone & Welch, 2007; Tsai & Wu, 2008; Yu, Wang & Lai, 2009). This move is in recognition that the traditional accounting and statistical methods have proven less reliable in their predictive power (Zhang, Hu, Patuwo & Indro, 1999; Gao, Cui & Po, 2008). In addition, ML approaches have been found to perform well in domains where there is a large amount of data but limited supporting theory (Tan & Gilbert, 2003). The general learning algorithms employed by ML techniques have the ability to assemble classifiers or hypotheses that can proffer an explanation relevant to the complex inter-relationships within domain datasets (Tan & Gilbert).

The classification accuracy of ML1 - ML3 was measured by comparing their predicted output versus historical data for foreclosures in the South Florida (Miami-Dade, Broward, and Palm Beach) area. Data was acquired from Dextec Systems for all identified input and output variables for the last three years. A suitable subset of data was used to train ML1 - ML3, while the remaining subset of data was used to test ML1 - ML3's predictive power.

The outcome variable of interest was whether a mortgage resulted in foreclosure within a specified period of time (three years) of its inception. The input variables used as predictors are restricted to data available at the time of the inception of the loan and may be grouped as follows:

- Variables that characterize the mortgage parameters
- Variables that characterize the borrower
- Macroeconomic indicators
- Other indicators specific to the location of the property under consideration.

Relevance and Significance of Study

The contribution of this study to the body of IS research is to demonstrate the suitability and value of ML techniques when applied to the foreclosure prediction problem. A general search for literature specifically targeting 'Foreclosure Prediction' results in numerous articles which regurgitate numbers supplied by industry sources and organizations (Olick, 2010; Brown, 2009; Johnson, 2009). A distinct methodology for deriving said numbers is seldom supplied, and tends to be more of an account of total regional foreclosures within a past or current period rather than a prediction. Some articles present economic indicators in support of stated forecast, while others merely comment on perceived trends (Silva, 2009). Of these sources, the Mortgage Bankers

Association (MBA) stands out as an organization that attempts to collate and present legitimate metrics related to foreclosures (2008). Given the above, this study will be among the first to develop a foreclosure prediction model based on ML techniques.

Barriers and Issues

The primary obstacle that this dissertation project encountered was that certain independent variables were not available because of issues pertaining to the Privacy Act (see Definition of Terms) and/or difficulty in consistent measurement.

Hypothesis

The comparison of machine learning techniques was based on the hypothesis that foreclosure rates, and associated actions, are dependent on critical demographic (age, gender), economic (per capita income, inflation) and regional variables (predatory lending, unemployment index). The task of the machine learning techniques was to identify a function that well approximates the relationship between these explanatory variables and the binary outcome of interest -- whether foreclosure occurs within three years of a loan's inception. This study was a binary classification problem, and the stated goal was accomplished by designing the study as per the framework illustrated in Figure 1.

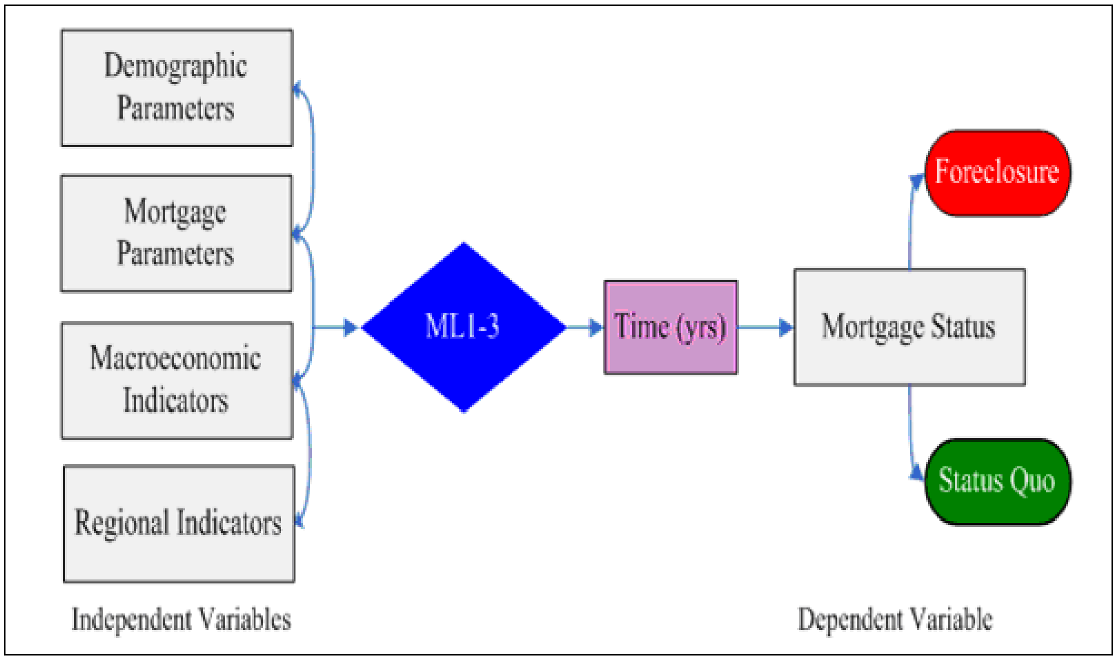


Figure 1 - Theoretical Framework

Definition of Terms

This section provides brief definitions for key terms that are used in chapters 2, 3 and 4.

Table 1 - Definition of Terms

Term	Definition
API	Application Programming Interface.
AppFabric	Microsoft's distributed memory caching technology used primarily by cloud based applications.
Azure	Azure is Microsoft's cloud computing infrastructure.
Cloud Computing	Cloud Computing is a software development approach that allows developers to consume data and computational services without significant concern for the operational status of the environment.

Term	**Definition**
CIL/MSIL	Object oriented assembly like code that Microsoft.Net compatible languages' source code is compiled into. Common Intermediate Language (CIL) was formerly called Microsoft Intermediate Language or MSIL.
IKVM	IKVM is a freely available open source implementation of Java for Microsoft.NET. It facilitates calls to Java classes directly from .NET code, and provides a .NET version of the Java Virtual Machine.
IY	Inception Year.
J#.Net	Microsoft's implementation of Java for the .Net framework 2.0. J#.Net supports Java code up to JDK 1.1.6.
JDK	Java Development Kit.
Lien	A form of security interest granted over an item of property to secure the payment of a debt or performance of some other obligation.
Linear Model	A mathematical model in which linear equations connect the random variables and parameters.
Macroeconomics	Branch of economics which studies the overall level of economic activity (Bowden, 1992, p. 98). Macroeconomic indicators are monetary figures that interact to influence the flow of money through an economy (Qi, 2001).
Managed Code	Microsoft code that strictly adheres to the data types defined by the Common Type System (CTS) and runs in the context of the Common Language Runtime (CLR).

Term	**Definition**
MemCached	A free open source, high-performance, generic, distributed memory object caching system for use in speeding up dynamic applications by alleviating database load and/or API calls.
OO	Object Oriented. A programming paradigm that attempts to decompose the behavior and properties of real world entities into representative templates that form the foundation upon which instances of the entity are created in a virtual work (Kay, 1996; Cho & Kim; 2001).
Overfitting	Overfitting generally occurs when a statistical model is excessively complex relative to the number of observations. Overfitting leads to poor predictive accuracy.
Plug-In	A software component which can be dynamically loaded and early bound through the implementation of a known interface(s). Usually extends or modifies the functionality of the parent software application.
Privacy Act	The Privacy Act of 1974 establishes a code of fair information practice that governs the collection, maintenance, use, and dissemination of personally identifiable information relevant to individuals.
Q*x*	*x* Quarters where *x* Ɛ {1,2,3,4,5,6,7,8,9,10,11,12}.
Reflection	A programming language’s ability to do type introspection during run-time.
SDK	Software Development Kit.

Term	Definition
SOA	Service Oriented Architecture. A software development architecture that stresses upon the decoupling of core components via the use of secure, distributed, consumable and platform neutral informational services.
SQL Azure	Microsoft's cloud implementation of its SQL Server database.
Tournament Selection	Tournament selection is commonly used in genetic algorithms to select an individual from a population of individuals. The selection process involves running several "tournaments" among a group of random individuals from the population. The individual with the best fitness score is then selected for crossover.
Use Case	A thorough definition of a system's behavior in direct response to a particular external request from another system or actor.
WCF	Windows Communication Foundation. Microsoft's current distributed component technology.
Worker Thread	A thread is the smallest unit of execution within a Windows process space and executes asynchronously to its parent. Worker threads are commonly used to handle background tasks that would otherwise put an application in a wait/busy state.

Summary

One of the many objectives of Information Systems (IS) research is to advance knowledge that encourages dynamic applications of Information Technology (IT) towards solving tangible problems in human organizations (Hevner, March, Park & Ram, 2004). Foreclosure is a significant problem that can threaten the stability of an economy (Calhoun, 2010; Durbin, 2010). As such, any predictive model that can accurately anticipate foreclosures, with a reasonable degree of accuracy, will automatically gain significant societal value. Therefore, this research attempted to develop a foreclosure prediction model based on ML techniques which, with confidence, will stimulate additional examinations of the topic.

Chapter 2

Literature Review

Introduction

This chapter provides a more in-depth examination of the genesis models and ML technologies used in this study. With regard to the ML technologies, the seminal papers and authors thereof are identified and discussed. Said discussions will lead into explorations of the foundation algorithms and or mathematical derivations for each ML type. The genesis models were examined from an evolutionary perspective, starting with statistical methods and proceeding to current research of ML techniques in the development of new models. Finally, this chapter will conclude with a summary of some advantages and disadvantages for each ML technique.

Machine Learning

ML is a sub-field of Artificial Intelligence (AI) that focuses on the development of computational algorithms that allow computers to induce rules and patterns from empirical data (Langley & Simon, 1995). ML is an interdisciplinary field which draws knowledge from mathematics and statistics, computer science, engineering, cognitive science, optimization theory and other scientific and mathematical disciplines (Ghahramani, 2004). In ML methods, the input values and related output values are used to algorithmically deduce an assumed (but unknown) functional relationship among variable types that can be applied to predict outputs for new input values (Steinwart & Christmann, 2008, p.2). ML methods *generally* fall into three main categories (Russel & Norvig, 2003, p.650):

- Supervised learning methods are based on the existence of a priori data knowledge whereby a sub-set of the input(s) and associated output(s) can be used by computational algorithms to classify and cluster the input data (Tan & Gilbert, 2003). In this learning method, the input observations are known to cause the output observations, therefore, the inputs are at the beginning and the outputs are at the end of the causal chain (Tan & Gilbert).
- Unsupervised learning methods do not depend on the existence of a priori data knowledge in performing classification and clustering tasks (Tan & Gilbert). In unsupervised learning, all the observations are assumed to be caused by latent variables at the end of the causal chain.
- Reinforcement learning methods are based on psychology's reinforcement theory which attempts to shape behavior by controlling the consequences of said behavior (Russel & Norvig, 2003, p.650). Reinforcement learning agents do not depend solely on inputs from the controller, but also rely on feedback provided from the execution environment to alter or adjust their behavior accordingly. Continuous positive or negative feedback allows the agent to acquire reinforced knowledge of the environment (Ghahramani, 2004).

The following sections (ML1 - ML3) discuss the supervised ML techniques used in this study.

ML1: Classification Trees

A classification tree is a decision tree with discrete output values as opposed to continuous values in the case of regression trees (Russel & Norvig, 2003, p.653; Abu-Nimeh, Nappa, Wang & Nair, 2007). As decision trees, classification trees are an induced collection of decision branches, leafs and nodes that classify observations dependent on input values (Cielen, Peeters & Vanhoof, 2004). Each node in a decision tree represents a test of a property value, whiles the branches represent the possible values of the test (Russel & Norvig).

Classification trees classify instances into the categories of the dependent attribute (*Y*) by using the values of the independent (*X*) attributes (Morasca, 2002). The classification process starts with the association of the dependent variable with a probability distribution for random selection of a binary (0, 1) entity (Morasca). The probability distribution does not use the independent variables, thus the selection probability $p(y)$ is unconditional. As the process progresses, the conditional probability $p(y|x)$ is used. As such, each independent attribute will have varying degrees of usefulness for classifying instances as either 0 or 1. An attribute *X* is considered "best" based on the maximization of the information gain $H(Y) – H(Y|X)$, where

$H(Y) = - \sum_y p(y) \log p(y)$ and $H(Y|X) = - \sum_x p(x) \sum_y p(y|x) \log p(y|x)$ (Morasca; Russel & Norvig, 2003, p.659).

Several inductive algorithms exist for the generation of classification trees. Quinlan's (1986) ID3 and (1993) C4.5, Breiman, Friedman, Olshen & Stone's (1984) CART are examples of commonly used induction algorithms for classification trees (Esmeir & Markovitch, 2004). Many classification tree algorithms are greedy because they induce from the top-down, making best possible decisions at each node (Esmeir & Markovitch). Additionally, *Ockham's Razor* (the least complex explanation for a given phenomenon is most likely the correct one) is drawn upon to choose from among equally competing hypotheses (Russel & Norvig, 2003, p.659; Murphy & Pazzani, 1994).

The Recursive Partitioning Algorithm (RPA) is a foundation algorithm for many classification tree techniques (Ravi-Kumar & Ravi, 2007). RPA is a non-parametric classification technique based on pattern recognition. Quinlan's C4.5 is an RPA that extends ID3 (Quinlan, 1986) for use with continuous variables (Morasca, 2002). Baesens, Van Gestel, Stepanova, Suykens & Vanthienen (2003) applied C4.5 to credit scoring classification. The following is a pseudo-code representation of an RPA adapted from Russel & Norvig (2003, p.658):

Function 1 - Classification Tree Learning Algorithm

```
Begin Function TreeLearning (examples, attributes, default) returns Tree
    if (examples.count==0) return default;
    if (examples.all.output==classification) return classification;
    if (attributes.count==0) return MaxClass(examples);
    declare best, tree, m;
    best = ChooseAttribute(examples, attributes);
    tree = new Tree(best); //root node of new tree is best
    m = MaxClass(examples);
    for each $v_i$ in best
        $examples_i$ = examples.Find(where best = $v_i$);
        //recursive function call
        declare subtree = TreeLearning ($examples_i$, attributes - best, m);
        tree.AddBranch($v_i$, subtree);
    next $v_i$
    return tree;
End function
```

```
Begin Function ChooseAttribute(examples, attributes)
    declare dictionary = new Dictionary<key, value>();
    for each attribute in attributes
        dictionary.Add(attribute.InformationGain(examples), attribute);
    next attribute
    dictionary.Keys.Sort;
    return dictionary[dictionary.Keys[dictionary.Count-1]];
End function

Begin Function MaxClass (examples) returns classification
    declare list = new List<classification>;
    for each example in examples
        list.Add(example.classification);
    next example
    list.Sort;
    return list[list.Count-1];
End function
```

ML2: Support Vector Machines

SVM is a kernel machine learning method that performs classification tasks by constructing maximal margin hyperplanes in a multidimensional space in order to separate cases of different class labels (Moore, 2003). Maximal margin hyperplanes provide the greatest separation between class boundaries with the training point nearest to the hyperplane acting as support vectors (Min & Lee, 2005; Russel & Norvig, 2003, p.751).

The genesis of SVM can be traced back to the work of Boser, Guyon & Vapnik (1992) which drew upon the *Generalized Portrait Algorithm* (GPA) by Vapnik and Lerner (Steinwart & Christmann, 2008, p.13). Boser, Guyon & Vapnik's work was originally called "Maximal Margin Classifier" and later "Hard Margin SVM" (Steinwart & Christmann, p.14). The GPA is based on Vapnik and Chervonenkis' (1971) Structural

Risk Minimization (SRM) principle from computational learning theory (Steinwart & Christmann). SRM is an inductive principle in machine learning designed to address the problem of overfitting when a generalized model is selected from a finite data set (Vapnik & Chervonenkis, 1971).

From Boser, Guyon & Vapnik's (1992) original work, for a linearly separable training set ($i=1,\ldots.,N$), the SVM hyperplane satisfies the inequality –

(1) $y_i\ (w\bullet x_i + b) \geq \forall i \in \{1,\ldots,N\}$ where w is a normal and b is a bias (Gao, Cui & Po, 2008; Min & Lee, 2005). Furthermore, $y_i \in \{-1, +1\}$, $x_i \in R^d$ is a case of the training set where d is the dimension of input space and $w\bullet x_i$ is the dot product of the normal and x_i (Gao, Cui & Po). The dot product is an operation which takes two vectors and returns a real-valued scalar quantity. The dot product of two vectors $a = [a_1, a_2, \ldots, a_n]$ and $b = [b_1, b_2, \ldots, b_n]$ is therefore defined as: $\sum_{i=1}^{n} (a_i\, b_i)$ (William et al., 1998). Under the constraint specified in (1), the optimal hyperplane is equivalent to minimizing $||w||^2$ (Min & Lee).

For non-linear surfaces a set of slack variables, $e_{i\ldots.n}$ and a penalization variable C for misclassification are introduced in order to relax the optimization problem (Gao, Cui & Po, 2008; Min & Lee, 2005). The optimal hyperplane is, therefore, now achieved by minimizing (2) $\left[0.5||w||^2 + C\sum_{i=1}^{n}(e_i)\right]$ with respects to w,b,e under the constraint (3)

$y_i\ (w \bullet x_i + b) \geq 1 - e_i,\ e_i \geq 0,\ \forall i \in \{1,\ldots,N\}$ (Gao, Cui & Po; Steinwart & Christmann, 2008, p.15). Finally, a Lagrange multiplier α is applied to each constraint in order to present the maxima of the linear problem. Lagrange multipliers are used to find the extrema of a function that is subject to fixed outside conditions or constraints. As such, the objective function with respect to α is: (4) $\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \bullet x_j)$; under the constraints: $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C, \forall i \in \{1, \ldots, N\}$ (Gao, Cui & Po; Steinwart & Christmann). SVM in a non-linear space is thus a quadratic programming optimization problem (Russel & Norvig, 2003, p.749).

Kernel Methods (KMs) are pattern analysis algorithms which discover relation types such as clusters, rankings and classifications in general types of data (Moschitti, 2008). A kernel function *k* is a mapping function which performs a non-linear map to a higher dimensional feature space (Russel & Norvig, 2003, p.751; Wu, Tzeng, Goo & Fang, 2007). Kernel functions are usually represented as $K\ (x_i, x_j)$ and replace the inner products of equation (4) in non-linear SVM such that eq. (4) becomes $\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i \bullet x_j)$ (Russel & Norvig; Gao, Cui & Po). The selected kernel function is dependent on the classification task and the desired level of accuracy. The RBF (Gaussian) kernel has been used in many bankruptcy prediction and credit scoring studies (Lee, 2007; Wu, Tzeng, Goo & Fang; Min & Lee, 2005; Fan &

Palaniswami, 2000; Schebesch & Stecking, 2005). The RBF kernel is defined as follows:

$k(x', x'') = \exp\left(-\frac{\|x' - x''\|^2}{\sigma^2}\right)$ (Steinwart & Christmann, 2008, p.116).

Optimal choice of SVM kernel parameters is critical to classification accuracy and stability (Wu, Tzeng, Goo & Fang, 2007; Min & Lee, 2005). The penalization variable *C* and the bandwidth of the RBF kernel σ^2 (sigma squared) must be cautiously predetermined. Exponentially growing sequences of *C* (e.g. $C^{-5},\ldots,C^{5}$) and σ (e.g. $\sigma^{-10},\ldots,\sigma^{5}$) is an acceptable method for pre-selecting SVM parameters, but is not without fault (Min & Lee, 2005).

Wu, Tzeng, Goo & Fang (2007) proposed a GA-SVM model for determining the optimal choices for these parameters relevant to bankruptcy prediction. Their approach is based on using a Real Valued Genetic Algorithm (RGA) to optimize the parameters of the SVM. Wu, Tzeng, Goo & Fang encoded a chromosome *X* as $\{p_1, p_2\}$ where $p_1 = C$ and $p_2 = \sigma$. The hit ratio is used as the fitness function whereby the GA-SVM's performance is compared against other models such as traditional SVM, logit and Neural Network (NN). Wu, Tzeng, Goo & Fang concluded that prediction accuracy was drastically improved by using the GA to seed the SVM.

ML3: Genetic Programming

Genetic programming (GP) is an AI programming technique based on natural selection (Lensberg, Eilifsen & McKee, 2006). Genetic programming is founded upon genetic algorithms (GA), which are implemented using coded bit strings commonly referred to as chromosomes (Russel & Norvig, 2003, p.133). Each gene in a chromosome, therefore, represents a specific behavioral condition or state within the problem space (Lensberg, Eilifsen & McKee). In genetic algorithms, the chromosomes are evolved through generations via a process of mating, mutation and tournament selection based on suitability to a defined objective function or fitness function as in the case of GPs (Russel & Norvig). The parameters which control mating and mutation are referred to as the Genetic Operators.

GPs differ from GAs in that the mutated elements are executable structures, often represented in the form of LISP expression trees, Java, or machine code programs for stack based machines, as opposed to bit strings (Russel & Norvig, 2003; Riolo, Worzel & Soule, 2009). As such, GPs use a subset of a suitable programming language to represent the individual behavior rules (Lensberg). In GP, new generations of programs are evolved through a process of mating of the top two selected programs. Primarily, tournament selection is used to randomly select *n* number of programs from the GP population. The top two programs are then determined by rank according to the values returned from execution of their fitness function. These programs are mated based on the genetic operators (crossover point & mutation factor) and their offspring replace the least fit programs in the population. This concept is illustrated in Table 2.

Table 2 - Example of GP Program Selection

Randomly Selected Programs	**Fitness Score**	**Rank**
11	0.81	1
27	0.65	3
35	0.57	4
n	0.78	2

Program 11 & *n* will be selected for mating.

The crossover point is a point between 1 and the number of points in a program tree. During mating, this point is randomly generated for each of the programs involved in the mating process. The sub-trees rooted at the two picked points are then used in a recombination process to produce offspring. In mutation, a single program is randomly selected and a point in the program's sub-tree is deleted. A new sub-tree is then grown at the mutation point thus creating a new program.

GPs have had successful applications in areas such as automated combination of analog electrical circuits (Koza et al, 1999), automatic creation of computer programs (Bruce, 1995) and solving complex state-space search problems (Russel & Norvig). The upsurge in interest of GPs is attributed to John Koza's 1992 publication titled 'Genetic Programming: On the Programming of Computers by Means of Natural Selection'. In this work, Koza introduces four examples of GPs and discusses several evolutionary concepts such as evolution of emergent behavior, evolution of subsumption, entropy-driven evolution, evolution of strategy, and symbolic regression.

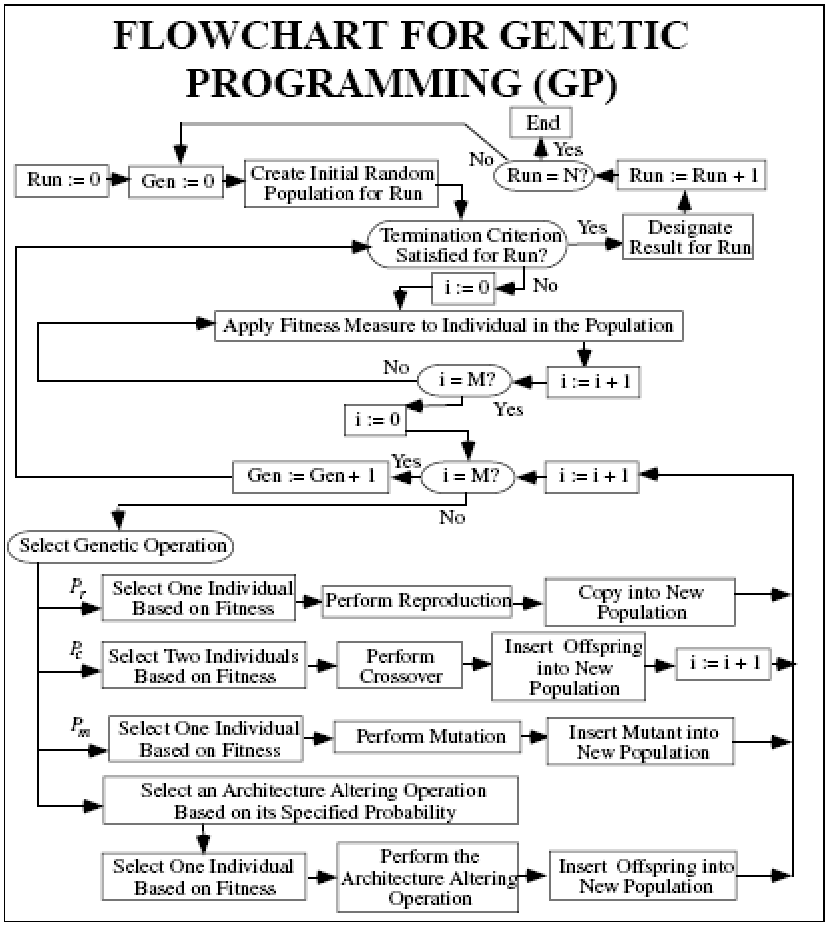
Symbolic regression is a GP technique for the search of a satisfactory mathematical expression that fits a set of data points, in a specific domain, from a constrained space of possible functions and terminal conditions (Koza, 1992, p.162). Simply stated, symbolic regression, also known as symbolic function identification, derives an equation from a given set of data points. In symbolic regression, pre-determination of the relationship type is minimized by a chosen set of standard mathematical and logical operators known as the instruction set (Koza, 1992, p.81). A simple instruction set *F*, can be such that *F = {+, -, *, /, and, or, not, conditional (if-then-*

else), loop, recursion}. Set *F* is generally sufficient to account for most linear and polynomial relationships (Koza, p.163).

Symbolic regression uses Koza's (1992) Automatically Defined Functions (ADFs). ADFs are programs that consist of a function defining branch that can potentially utilize subroutines, loops, recursion and internal storage to promote the reuse of code, and a result producing branch (Koza, 2008, p.81). Through the evolutionary process, the main program branch is free to decide how to use the ADFs to find a solution within the constrained space of possible functions (Langdon & Poli, 2002, p.11). Symbolic regression has been applied to the bankruptcy problem by Lensberg, Eilifsen, and McKee (2006) with favorable results, and has also been applied to the credit scoring problem (Abdou, 2009).

The following is a pseudo-code representation of Koza's (2008) symbolic regression GP algorithm illustrated in Figure 2.

```
1. Generate population of n randomly composed programs that comprise an
   instruction set F.
2. Set termination condition and max generations.
3. Loop until termination condition is met or max generations reached
    a. Calculate fitness score for each program in current generation i
    b. Randomly select genetic operation
        i. Case reproduce
            x1. Select programs for mating.
            x2. Determine crossover points
            x3. Create offspring and into new (i+1) population
     ii.  Case mutate
            x1. Select one program based on fitness
            x2. Mutate program
            x3. Insert mutant into new (i+1) population
    iii.  Case architecture alteration
            x1. Select one program based on fitness
            x2. Perform architecture altering operation
            x3. Insert offspring into new (i+1) population
    c.  End select
    d.  Increment generation counter (i++)
4.  End loop
5.  Output program designation.
```


FLOWCHART FOR GENETIC PROGRAMMING (GP)
End
Yes
No
Run := 0
Gen := 0
Create Initial Random Population for Run
Run = N?
Run := Run + 1
Termination Criterion Satisfied for Run?
Yes
Designate Result for Run
i := 0
No
Apply Fitness Measure to Individual in the Population
No
i = M?
i := i + 1
Yes
i := 0
Yes
Gen := Gen + 1
i = M?
i := i + 1
No
Select Genetic Operation
P_r
Select One Individual Based on Fitness
Perform Reproduction
Copy into New Population
P_c
Select Two Individuals Based on Fitness
Perform Crossover
Insert Offspring into New Population
i := i + 1
P_m
Select One Individual Based on Fitness
Perform Mutation
Insert Mutant into New Population
Select an Architecture Altering Operation Based on its Specified Probability
Select One Individual Based on Fitness
Perform the Architecture Altering Operation
Insert Offspring into New Population
Figure 2 - Symbolic regression GP flow chart. (Koza, 2008)

Bankruptcy Prediction

Beginning with the seminal paper on financial failure prediction by Beaver (1966), work on bankruptcy prediction logically progresses from purely an accounting practice to applications of ML techniques. ML approaches such as Neural Networks (Perez, 2006), Genetic Programming (Abdelwahed & Amir, 2005; McKee & Lensberg, 2002) and Support Vector Machines (Shin, Lee & Kim, 2005; Min & Lee, 2005) have been applied to the bankruptcy problem. The bankruptcy problem is considered difficult because of the number of variables and the complexity of their relationships (Ohlson, 1980; Altman, 1984; Keasey & Watson, 2005; Ward, 2006). The application of ML techniques to the bankruptcy problem has generally indicated better results when compared to purely statistical approaches (Laitinen & Laitinen, 2000; Charalambous, Charitou & Kaourou, 2000). In their review paper titled 'Bankruptcy prediction in banks and firms via statistical and intelligent techniques', Ravi Kumar & Ravi (2007) concluded that stand-alone statistical techniques are no longer fashionable in bankruptcy prediction research. Ravi Kumar & Ravi illustrate that ML techniques, particularly neural networks followed by rough sets and evolutionary approaches are currently the most commonly used approaches.

A literature search unearthed a plethora of papers that focus on bankruptcy prediction and financial distress indicators from accounting and AI perspectives. Among the first papers to address the combination of AI with bankruptcy prediction is Odom & Sharda's (1990) 'A neural network model for bankruptcy prediction'. In the midst of the more recently cited papers is Lensberg, Eilifsen & McKee (2006), which focuses on genetic programming and bankruptcy theory development. Lensberg, Eilifsen & McKee is well cited in papers published in refereed journals such as Expert Systems with Applications, Knowledge-Based Systems and Computers & Operations Research (Rom & Slotnick, 2009; Tsai, 2008; Lee & Shih, 2009; Hung & Chen 2008). Lensberg, Eilifsen & McKee is the bankruptcy model that will be drawn upon for this dissertation study.

Credit Scoring Models

In the U.S., credit models are used to calculate a score that is representative of an individual's creditworthiness (myFICO, 2009). Traditional credit scoring models are usually based on accounting ratios and regression analysis (Finlay, 2009). Financial institutions use the scores generated by the models to evaluate the risk involved in lending money to consumers. As such, credit scores determine who qualifies for a loan and the parameters of the loan (interest rate, term etc). The Fair Isaac Corporation created the first credit scoring system in 1958 (myFICO). Though the exact details of their model are unknown, it is largely based on the traditional approach (Finlay). Recently, recognition of the inadequacies of current credit scoring models has led to the application of ML techniques in the pursuit of more robust models. Neural Networks, Support Vector Machines and Genetic Programming have all been applied to the problem with optimistic results (Abdou, 2009; Bellotti & Crook, 2009; Tsai, 2008; Yu & Wu, 2008; Schebesch & Stecking, 2005).

Foreclosure Factors

The *options theory* of foreclosures states that foreclosures occur when a property's value becomes less than what is owed on the mortgage (Grover & Todd, 2008). Additionally, the *trigger event theory*, suggest that foreclosures occur when the borrower experiences financial and physical setbacks which hinder continued payments (Grover & Todd). Though both of these theories hold some validity, neither truly captures the interaction among the micro/macro economic, social, regional and legal factors at play in the foreclosure dynamic.

Summary

The following section summarizes some of the advantages and disadvantages of ML1 - ML3 from the perspective of the technology and relevance to the proposed study.

Classification Trees

Advantages

- Has been applied to the bankruptcy problem (Marais, Patell & Wolfson, 1984; Frydman, Altman & Kao, 1985).
- Excels at feature identification by interpreting interactions among predictors (Abu-Nimeh, Nappa, Wang & Nair, 2007).
- C4.5 has been applied to credit scoring classification (Baesens, Van Gestel, Stepanova, Suykens & Vanthienen, 2003).
- Can handle both categorical and continuous variables (Morasca, 2002).
- Tends to produce models that are easy to interpret and can be used to create set of IF-THEN rules (Russel & Norvig, 2003; p. 654).

Disadvantages

- Classification trees can be unstable and minor data variations can result in the generation of very different looking trees (Russel & Norvig, 2003; p. 654).
- Can succumb to overfitting of data (Russel & Norvig, p. 662).
- Computationally expensive to train. The order of complexity for C4.5 with a dataset of size *n* and each instance having *m* attributes is $O(m.n.\log n) + O(n (\log n)^2)$.

Support Vector Machines

Advantages

- Supports linear, polynomial, radial basis function (RBF) and sigmoid kernels for regression and classification tasks (Moore, 2003).
- Can process multiple continuous and categorical input variables (Schebesch & Stecking, 2005).
- Kernel parameters may be optimized via a hybrid GA-SVM approach as demonstrated by Wu, Tzeng, Goo & Fang (2007).
- Has been used in many recent bankruptcy prediction and credit scoring studies such as Bellotti & Crook, 2009; Lee, 2007; Min & Lee, 2005; Schebesch & Stecking, 2005; Min, Lee & Han (2006); Gao, Cui & Po, 2008.

Disadvantages

- Optimal choice of SVM kernel parameters is critical to classification accuracy and stability (Wu, Tzeng, Goo & Fang, 2007; Min & Lee, 2005).
- Choice of kernel function can have an impact on the classification task and the desired level of accuracy.

Genetic Programming

Advantages

- Symbolic regression has been applied to the bankruptcy problem by Lensberg, Eilifsen, and McKee (2006) with favorable results.

- Most linear and polynomial relationships can be deduced by a simple instruction set *F* such that *F = {+, -, *, /, and, or, not, conditional (if-then-else), loop, recursion*} (Koza, 1992, p.163).
- Additions to set *F* (e.g. sin, cost, log, exp) can create a wider variety of output expressions.
- Has been applied to the credit scoring problem (Abdou, 2009).

Disadvantages

- Identification of the correct fitness function is critical to satisfactory discovery of a workable expression.
- Execution time can be very high (Lensberg, Eilifsen & McKee, 2006).
- Expanded function sets increase the potential of bloat which is an excess of code expansion caused by the genetic operators searching for superior solutions without a resultant enhancement in fitness (Silva & Costa, 2005).
- The number of major and minor control parameters is high in comparison to other machine learning methods. Koza (1992, p. 641) enumerates approximately nineteen parameters of which population size $\boldsymbol{M}$, max number of generations $\boldsymbol{G}$, crossover probability $\boldsymbol{p_c}$, reproduction probability $\boldsymbol{p_r}$, crossover point $\boldsymbol{c_x}$, probability of mutation $\boldsymbol{p_m}$ are critical to accuracy.

Chapter 3

Methodology

Introduction

This chapter focuses on presenting the technology that was implemented, and the steps executed, to develop the proposed foreclosure prediction model. Emphasis was placed on the data that drove the study and the implementation of ML1 - 3 for side-by-side predictive comparison. The chapter concludes with a summary of the macro steps of the study.

Data Acquisition

All macroeconomic data was retrieved using the ALFRED® (2009) API. ALFRED is a RESTful (Representational State Transfer) web service, created by Economic Research Division of the Federal Reserve Bank of St. Louis, which provides access to archived U.S. regional economic data. REST is a client-server architectural style that is stateless, cacheable, exposes a uniform interface, and promotes layered system design (Fielding, 2000).

A stratified random sample of foreclosure data was requested from Dextec Systems. The vendor responded by providing an equal number of randomly selected foreclosed and un-foreclosed data for Miami-Dade, Broward, and Palm Beach counties. Though this does not represent a stratified sample, it does reduce sampling bias since each type of mortgage/county record has an equal chance of being selected. The total record count was 1000 distributed as illustrated in Table 3. The data obtained from Dextec Systems is available upon request.

Table 3 - Mortgage Data Totals

County	Type	Count
Broward County	Foreclosure	167
Broward County	Non-Foreclosure	167
Dade County	Foreclosure	167
Dade County	Non-Foreclosure	167
Palm Beach County	Foreclosure	166
Palm Beach County	Non-Foreclosure	166
	Total	1,000

Crime statistics from the National Archive of Criminal Justice Data (NACJD) was acquired and reviewed. The NACJD is a part of the Inter-University Consortium for Political and Social Research (ICPSR) at the University of Michigan. Though the NACJD data was extensive, a consistent, meaningful and regional (by zip/city) crime index could not be identified.

All data was imported into the database as described in Data Management. There was no need to scrub the data for consistency and balance. A balanced dataset exists if the ratio between the two output classes is not significantly greater than 1:1. The dependent and independent variable used in the study are presented next.

Variables

The variables illustrated in Table 5 - Table 9 were postulated in the development of the foreclosure prediction model. The selected variables were adapted or inferred from bankruptcy and credit scoring models by Lensberg, Eilifsen & McKee (2006) and Bellotti & Crook (2009) respectively. Furthermore, variable selection was limited to the variables that were relevant to the unit of analysis, readily available**,** and not subject to acquisition limitations. In some cases, the impact of a variable is mirrored by another variable thereby rendering the impact of the variable's exclusion moot. Credit score is an example of such a variable as its value is mirrored by interest rate. The following variables were initially identified but were later omitted:

Table 4 – Excluded Variables

Name	**Reason for Omission**
Income	Unavailable due to Privacy Laws.
Credit Score	As above.
Gender	As above.
Mortgage Payment	Not recorded by data vendor.
Average Age of Mortgagee(s)	As above.
Multi-Income	As above.
Crime Rate For Region	Difficulty in identifying consistent index.

.

Table 5 - Independent Variables – Mortgage Parameters

Name	**Data Type**	**Example/Scale of Measure/Comments**	**Data Source**	**Measurement Frequency**
Mortgage Type	Discrete	0= Fixed, 1= ARM, 3=Other	Dextec Systems	Inception and Current Year
Interest Rate	Continuous	6.02%. (Current rate in case of an ARM)	Dextec Systems	Inception and quarterly thereafter
Principal Amount	Continuous	Amount borrowed.	Dextec Systems	Inception
Mortgage Year	Discrete	Inception Year – Current Year	Dextec Systems	Annually
Current Market Value	Continuous	Estimate of amount that can be currently obtained for property if sold within next 3 months.	Dextec Systems	Inception and Current Year

Table 6 - Independent Variables – Macroeconomic

Name	Data Type	Example/Scale of Measure/Comments	Data Source	Measurement Frequency
Prime Rate	Continuous	The interest rate charged by banks to their most creditworthy customers.	ALFRED®	Inception and quarterly thereafter.
Inflation	Continuous	An increase in the cost of goods and services in an economy over a period of time due to loss of purchasing power in the medium of exchange.	ALFRED®	Inception and quarterly thereafter.
Consumer Price Index	Continuous	Average price for a typical market basket consumed by the average household.	ALFRED®	Inception and quarterly thereafter.

Table 7- Independent Variables – Demographic

Name	Data Type	Example/Scale of Measure/Comments	Data Source	Measurement Frequency
Zip	Discrete	Zip code or any integer based regional identifier	Dextec Systems	Inception

Table 8 - Independent Variables – Regional

Name	**Data Type**	**Example/Scale of Measure/Comments**	**Data Source**	**Measurement Frequency**
Regional Home Ownership Rate	Continuous	The homeownership rate is the percentage of homeowning households among all households in the given demographic group.	ALFRED®	Inception and quarterly thereafter
Predatory Lending	Discrete	Prevalence of predatory lending practices. (Strupp, 2009). Indicates whether region has laws which regulates predatory lending (Rose, 2008). {0,1}	MBA	Inception and Current Year.
Unemployment Rate	Continuous	Percentage of those in the labor pool who are unemployed.	ALFRED®	Inception and quarterly thereafter
Per Capita Income	Continuous	Amount each citizen receives if the yearly *regional* income is divided equally among everyone. (Bowden, 1992, p. 92).	ALFRED®	Inception and Semi-annually thereafter

Table 9 - Dependent Variable

Name	**Data Type**	**Possible Values**	**Data Source**	**Measurement Frequency**
Mortgage Status	Discrete	0 = *Status Quo* - Mortgage proceeds to maturity without any significant changes. 1 = *Foreclosure* - Mortgage fails and property is sold by financing house.	Dextec Systems	Inception and quarterly thereafter.

Workbench

For the comparative analysis of ML1 - ML3, a generic workbench was created to facilitate parallel processing of the mortgage data. The workbench, hereinafter referred to as Raptor, was designed with extensibility, scalability and **re-use** in mind. As such, Raptor was built using an SOA pattern that made monolithic and cloud based system deployment possible. In the cloud scenario, Raptor's core services (SDK) were

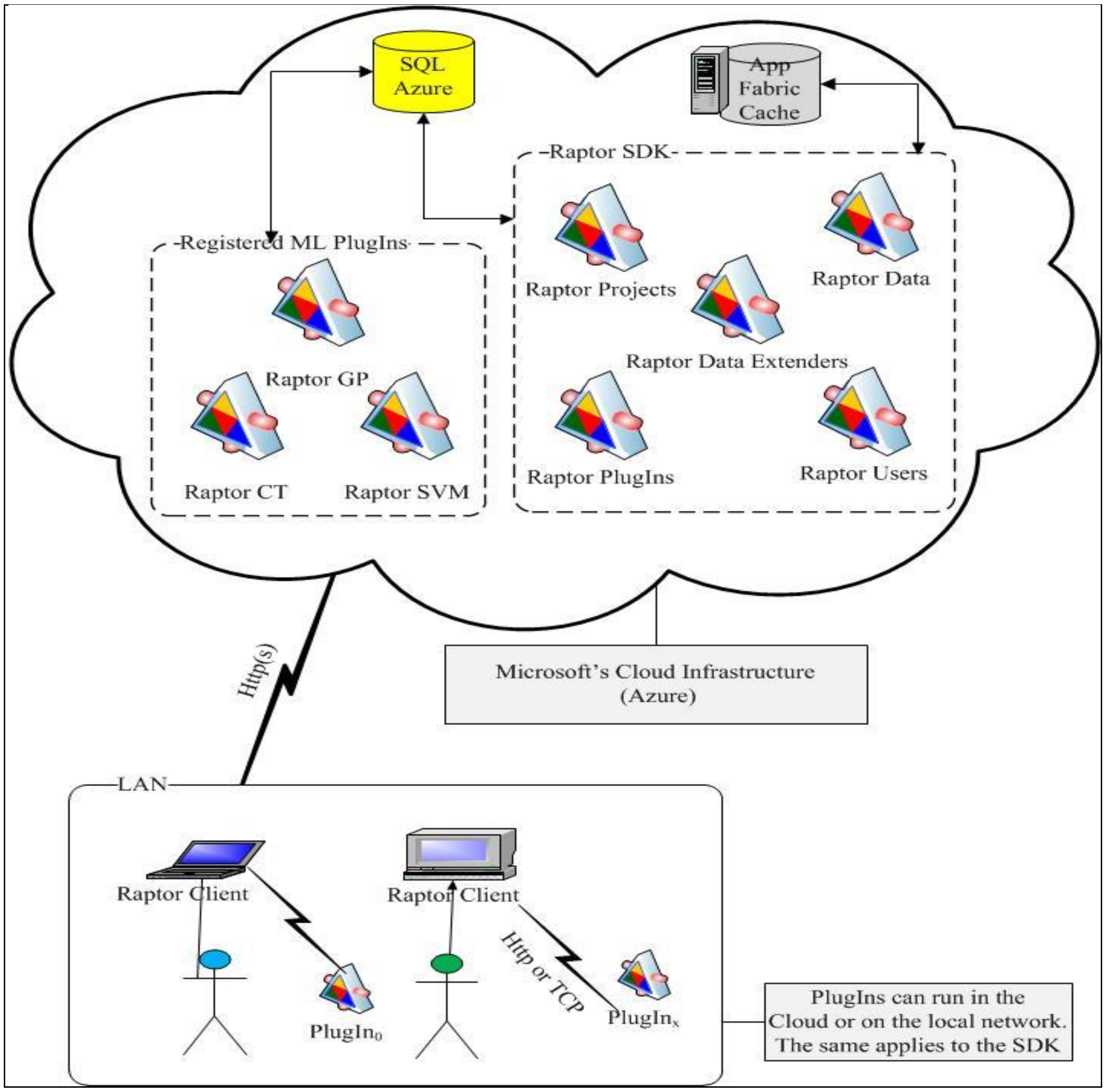


Figure 3 - High level overview of Raptor System.

deployed to the Azure development environment through Visual Studio 2010. The plug-ins for SVM, CT and GP were then deployed in Azure and registered with Raptor. Figure 3 illustrates the high level overview of the system. UML class diagrams for Raptor are presented in Appendix B - D.

Raptor was written in C# 4.0 with Visual Studio 2010. IKVM was used to bridge the Java → .Net gap as JDK limitations with J#.Net were encountered. The plug-ins were based on the following academically embraced open source ML libraries/SDKs. :

1. University of Waikato's machine learning library (**WEKA**) was used for developing the classification tree (ML1) implementation (Holmes, Donkin & Witten, 1997).
2. National Taiwan University's (NTU) Library for Support Vector Machines (**LIBSVM**) was used to develop the ML2 implementation (Chang & Lin, 2009).
3. George Mason University's Evolutionary Computation Research System (**ECJ19**) was initially used for developing the genetic program (ML3) implementation (Luke et al., 2008). Adapting ECJ19 for multi-parameter symbolic regression (MPSR) proved to be somewhat awkward because of its complex interfaces and reliance on configuration files. For this reason, the GP implementation used an MPSR library by Dudley (2011) as a wrapper around ECJ19 for improved ease of use.

Proxy classes to the ALFRED® API were built to promote simple consumption of the service. The Federal Reserve supplies excellent documentation on the API which supports language neutral consumption. The class diagram of the proxies is illustrated in Figure 22**.**

Data Management

The study required a database to manage the large amount of mortgage data that drove the ML plug-Ins. Microsoft's SQL Server 2008 was given preference over other databases (Oracle, IBM DB2) because of its ease of use in importing data, scrubbing data and migration to Azure (MSDN, 2008). Summary descriptions of the database tables that were created are presented in Appendix E, whiles Appendix F displays the relationships amongst the tables. Data management is handled by the use case 'Maintain Data' as illustrated in Figure 4.

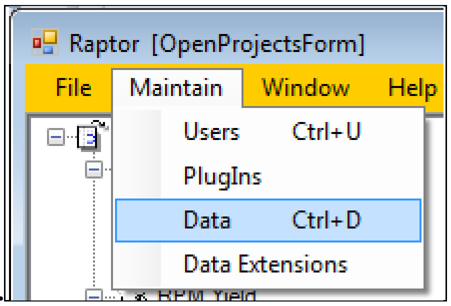
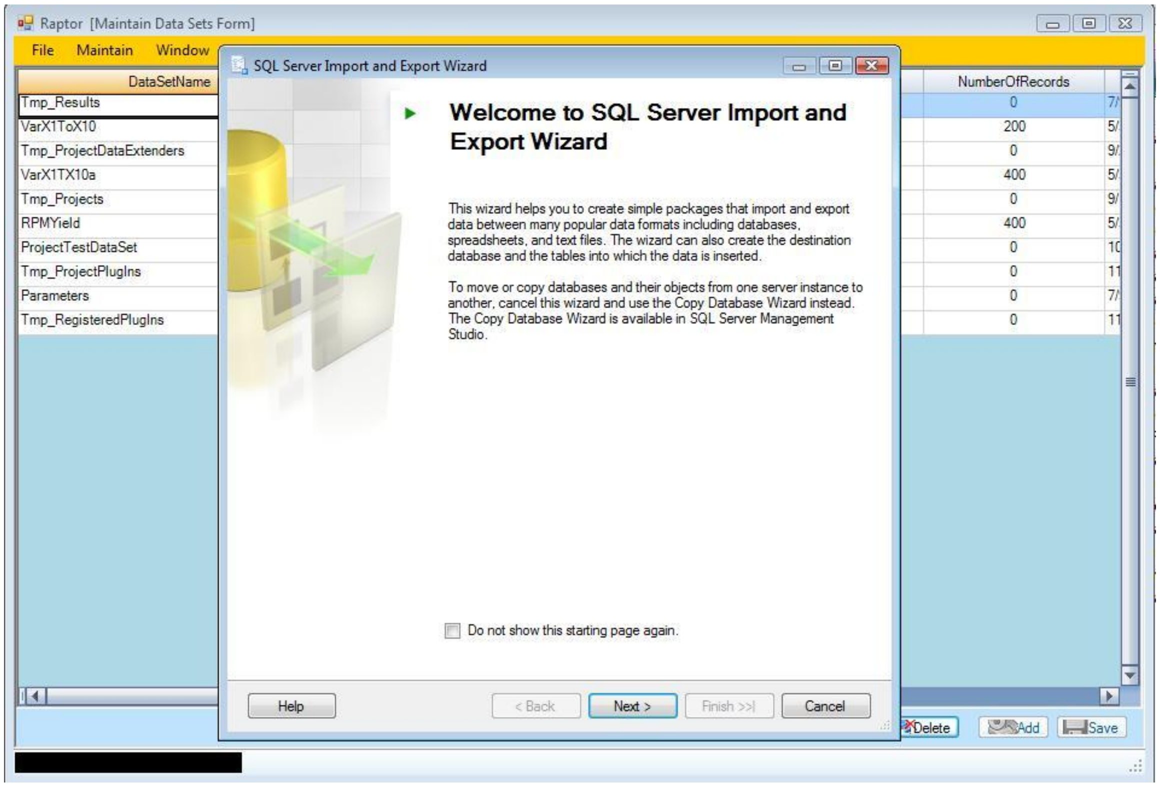


Figure 4 - Maintain Data

Data was imported by using Raptor to invoke a modified version of the Microsoft SQL Server Import and Export Wizard (Figure **5**). The output artifact of the wizard is a SQL Package, which stores the actions (new table etc.) to be performed on the target database. Raptor uses the WCF Service called RaptorData (Figure 3) to transport the package to the database server and to execute the package. All actions performed on the database are logged to a table called 'DatabaseLog' (Table 23). This functionality is intrinsic to SQL Server 2008. A database trigger called 'AddPrimaryKeyToNewTables' is fired for insert events on this table. The trigger's primary purpose is to add a column called ID, of type uniqueidentifier, to the newly imported table and to register (insert meta-data) said table in ‘RegisteredDataSets’ (Table 35). The column ID is used by Raptor to uniquely distinguish each row of data.


Raptor [Maintain Data Sets Form]
File
Maintain
Window
SQL Server Import and Export Wizard
Help
< Back
Next >
Finish >>|
Cancel
Delete
Add
Save

DataSetName	NumberOfRecords	
Tmp_Results	0	7/
VarX1ToX10	200	5/
Tmp_ProjectDataExtenders	0	9/
VarX1TX10a	400	5/
Tmp_Projects	0	9/
RPMYield	400	5/
ProjectTestDataSet	0	10
Tmp_ProjectPlugIns	0	11
Parameters	0	7/
Tmp_RegisteredPlugIns	0	11

Welcome to SQL Server Import and Export Wizard

This wizard helps you to create simple packages that import and export data between many popular data formats including databases, spreadsheets, and text files. The wizard can also create the destination database and the tables into which the data is inserted.

To move or copy databases and their objects from one server instance to another, cancel this wizard and use the Copy Database Wizard instead. The Copy Database Wizard is available in SQL Server Management Studio.

☐ Do not show this starting page again.

[redacted]

Figure 5 – SQL Server Import/Export Wizard.

Data Extenders

A Raptor Data Extender is a function which horizontally extends a registered dataset. As such, each extender maps directly to a column in the total dataset (Figure 8). Data Extenders are implemented as either WCF Services or .Net libraries. Meta-data that describes and facilitates execution of extenders are handled by the use case 'Maintain Data Extensions' (

Figure **6** & Figure 34).

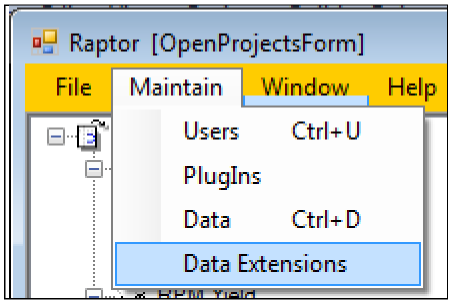


Figure 6 – Maintain Data Extenders

Unlike Raptor ML Plug-Ins, extenders do not implement any specialized interfaces or base classes. Instead, reflection is used to interrogate the service/library to discover available functions and their associated parameters. Data extender parameters can either be constant values or the values of adjacent columns. Since many extenders need run only once, Microsoft's App Fabric Caching Service was used to minimize database stress and network traffic. All ALFRED® (2009) metrics were implemented as data extenders.

ML Plug-Ins

Raptor ML Plug-Ins are logical components which implement the various machine learning algorithms. They are pluggable units controlled and dynamically executed by Raptor. Plug-ins are managed by the use case ‘Maintain PlugIns’ (Figure 7). Plug-Ins can be either .Net libraries, WCF Services or Web Services. Unlike data extenders, plug-ins must implement a common interface called ‘IRaptorPlugIn’ (Figure 10). If this interface is not implemented, the plug-in cannot be registered. Registration is similar to data extenders, in that, the purpose is to acquire and save meta-data that can be used to identify, describe, and execute the logical unit.

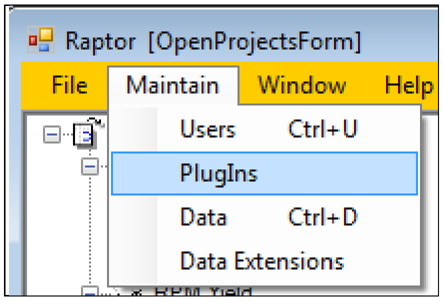
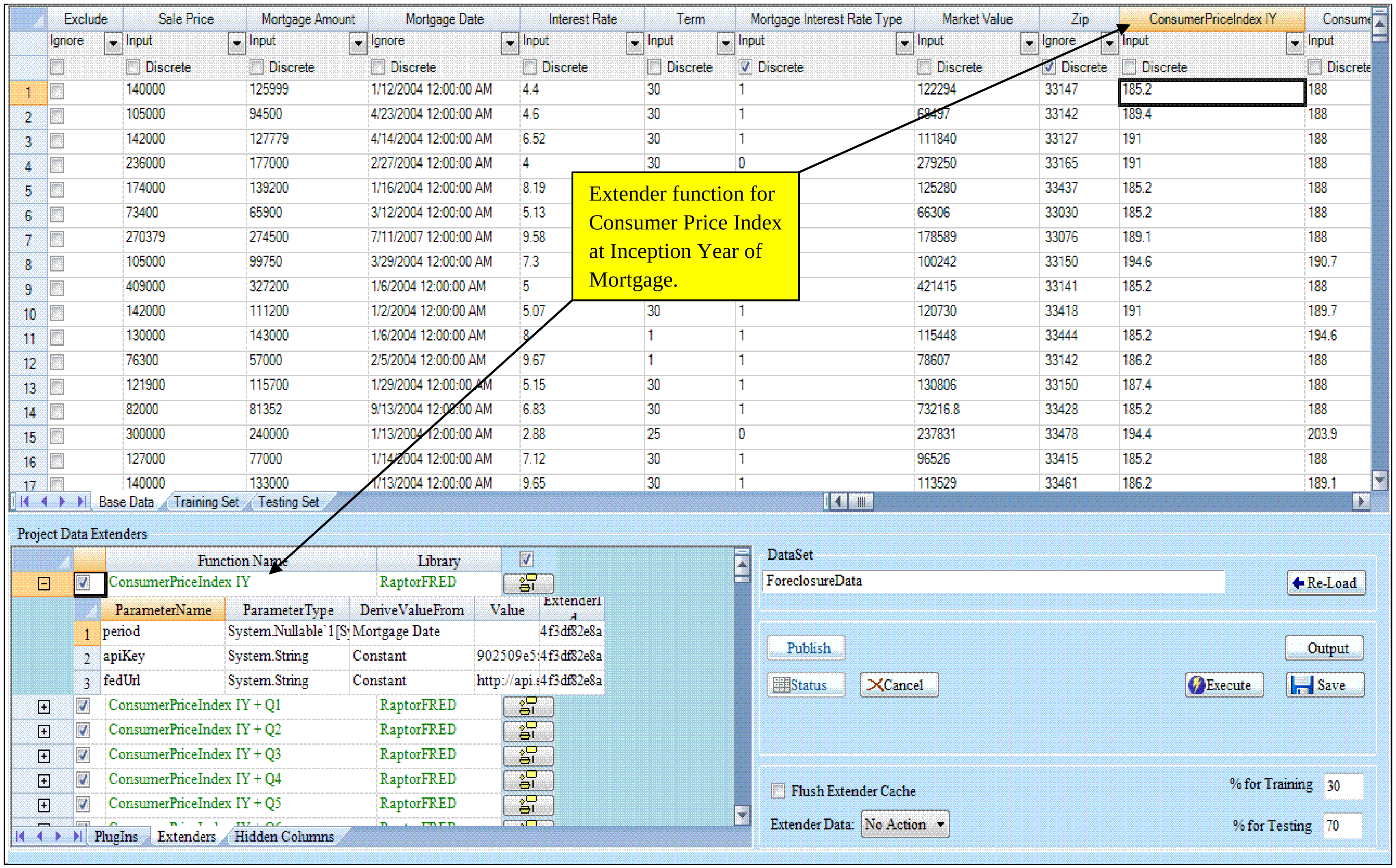


Figure 7- Maintain Plug-Ins

The default parameters for ML Plug-Ins are set during registration, and may be changed before and after the project is opened. The results of each plug-in run can be published to the database thereby creating a historical record for the run (Figure 9). The parameters of a historical record may also be made current at anytime. The results of a published plug-in run can also be viewed before the project is opened.


Extender function for Consumer Price Index at Inception Year of Mortgage.

	Exclude	Sale Price	Mortgage Amount	Mortgage Date	Interest Rate	Term	Mortgage Interest Rate Type	Market Value	Zip	ConsumerPriceIndex IY	Consume
	Ignore	Input	Input	Ignore	Input	Input	Input	Input	Ignore	Input	Input
		Discrete	Discrete	Discrete	Discrete	Discrete	✓ Discrete	Discrete	✓ Discrete	Discrete	Discrete
1		140000	125999	1/12/2004 12:00:00 AM	4.4	30	1	122294	33147	185.2	188
2		105000	94500	4/23/2004 12:00:00 AM	4.6	30	1	68497	33142	189.4	188
3		142000	127779	4/14/2004 12:00:00 AM	6.52	30	1	111840	33127	191	188
4		236000	177000	2/27/2004 12:00:00 AM	4	30	0	279250	33165	191	188
5		174000	139200	1/16/2004 12:00:00 AM	8.19			125280	33437	185.2	188
6		73400	65900	3/12/2004 12:00:00 AM	5.13			66306	33030	185.2	188
7		270379	274500	7/11/2007 12:00:00 AM	9.58			178589	33076	189.1	188
8		105000	99750	3/29/2004 12:00:00 AM	7.3			100242	33150	194.6	190.7
9		409000	327200	1/6/2004 12:00:00 AM	5			421415	33141	185.2	188
10		142000	111200	1/2/2004 12:00:00 AM	5.07	30	1	120730	33418	191	189.7
11		130000	143000	1/6/2004 12:00:00 AM	8	1	1	115448	33444	185.2	194.6
12		76300	57000	2/5/2004 12:00:00 AM	9.67	1	1	78607	33142	186.2	188
13		121900	115700	1/29/2004 12:00:00 AM	5.15	30	1	130806	33150	187.4	188
14		82000	81352	9/13/2004 12:00:00 AM	6.83	30	1	73216.8	33428	185.2	188
15		300000	240000	1/13/2004 12:00:00 AM	2.88	25	0	237831	33478	194.4	203.9
16		127000	77000	1/14/2004 12:00:00 AM	7.12	30	1	96526	33415	185.2	188
17		140000	133000	1/13/2004 12:00:00 AM	9.65	30	1	113529	33461	186.2	189.1

Base Data | Training Set | Testing Set

Project Data Extenders

	Function Name	Library
✓	ConsumerPriceIndex IY	RaptorFRED

	ParameterName	ParameterType	DeriveValueFrom	Value	ExtenderId
1	period	System.Nullable`1[Sy	Mortgage Date		4f3df82e8a
2	apiKey	System.String	Constant	902509e5:	4f3df82e8a
3	fedUrl	System.String	Constant	http://api.s	4f3df82e8a

	Function Name	Library
✓	ConsumerPriceIndex IY + Q1	RaptorFRED
✓	ConsumerPriceIndex IY + Q2	RaptorFRED
✓	ConsumerPriceIndex IY + Q3	RaptorFRED
✓	ConsumerPriceIndex IY + Q4	RaptorFRED
✓	ConsumerPriceIndex IY + Q5	RaptorFRED

PlugIns | Extenders | Hidden Columns

DataSet: ForeclosureData

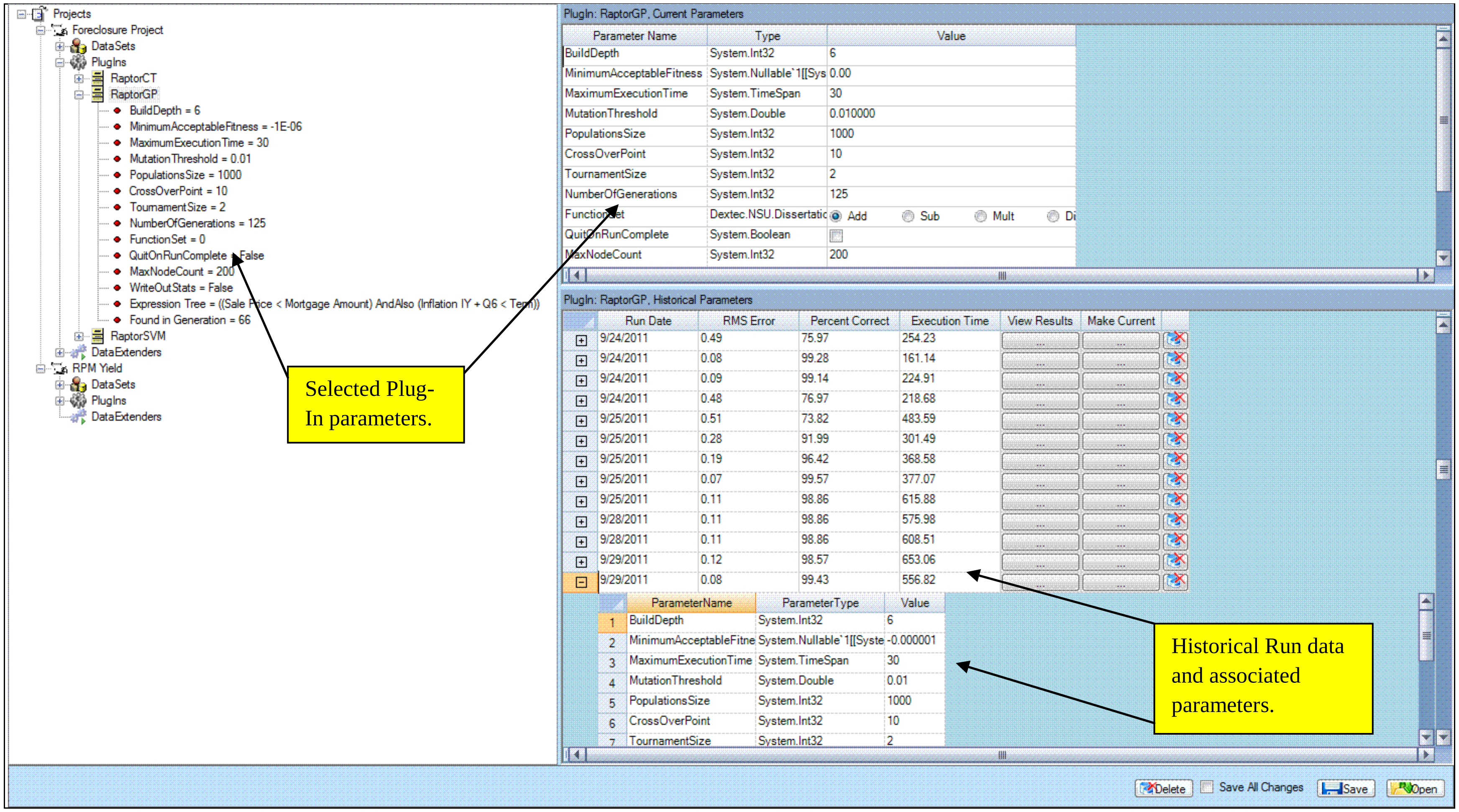
Flush Extender Cache: ☐

Extender Data: No Action

% for Training: 30

% for Testing: 70

Figure 8 - Raptor Data Extenders


Selected Plug-In parameters.
Historical Run data and associated parameters.
Delete
Save All Changes
Save
Open

- Projects
  - Foreclosure Project
    - DataSets
    - PlugIns
      - RaptorCT
      - RaptorGP
        - BuildDepth = 6
        - MinimumAcceptableFitness = -1E-06
        - MaximumExecutionTime = 30
        - MutationThreshold = 0.01
        - PopulationsSize = 1000
        - CrossOverPoint = 10
        - TournamentSize = 2
        - NumberOfGenerations = 125
        - FunctionSet = 0
        - QuitOnRunComplete = False
        - MaxNodeCount = 200
        - WriteOutStats = False
        - Expression Tree = ((Sale Price < Mortgage Amount) AndAlso (Inflation IY + Q6 < Term))
        - Found in Generation = 66
      - RaptorSVM
    - DataExtenders
  - RPM Yield
    - DataSets
    - PlugIns
    - DataExtenders

PlugIn: RaptorGP, Current Parameters

Parameter Name	Type	Value
BuildDepth	System.Int32	6
MinimumAcceptableFitness	System.Nullable`1[[Sys	0.00
MaximumExecutionTime	System.TimeSpan	30
MutationThreshold	System.Double	0.010000
PopulationsSize	System.Int32	1000
CrossOverPoint	System.Int32	10
TournamentSize	System.Int32	2
NumberOfGenerations	System.Int32	125
FunctionSet	Dextec.NSU.Dissertatic	[✓ Add] Sub Mult Di
QuitOnRunComplete	System.Boolean	☐
MaxNodeCount	System.Int32	200

PlugIn: RaptorGP, Historical Parameters

	Run Date	RMS Error	Percent Correct	Execution Time	View Results	Make Current	
⊞	9/24/2011	0.49	75.97	254.23	...	...	
⊞	9/24/2011	0.08	99.28	161.14	...	...	
⊞	9/24/2011	0.09	99.14	224.91	...	...	
⊞	9/24/2011	0.48	76.97	218.68	...	...	
⊞	9/25/2011	0.51	73.82	483.59	...	...	
⊞	9/25/2011	0.28	91.99	301.49	...	...	
⊞	9/25/2011	0.19	96.42	368.58	...	...	
⊞	9/25/2011	0.07	99.57	377.07	...	...	
⊞	9/25/2011	0.11	98.86	615.88	...	...	
⊞	9/28/2011	0.11	98.86	575.98	...	...	
⊞	9/28/2011	0.11	98.86	608.51	...	...	
⊞	9/29/2011	0.12	98.57	653.06	...	...	
⊟	9/29/2011	0.08	99.43	556.82	...	...	

	ParameterName	ParameterType	Value
1	BuildDepth	System.Int32	6
2	MinimumAcceptableFitne	System.Nullable`1[[Syste	-0.000001
3	MaximumExecutionTime	System.TimeSpan	30
4	MutationThreshold	System.Double	0.01
5	PopulationsSize	System.Int32	1000
6	CrossOverPoint	System.Int32	10
7	TournamentSize	System.Int32	2

Figure 9 - Plug-In Parameters

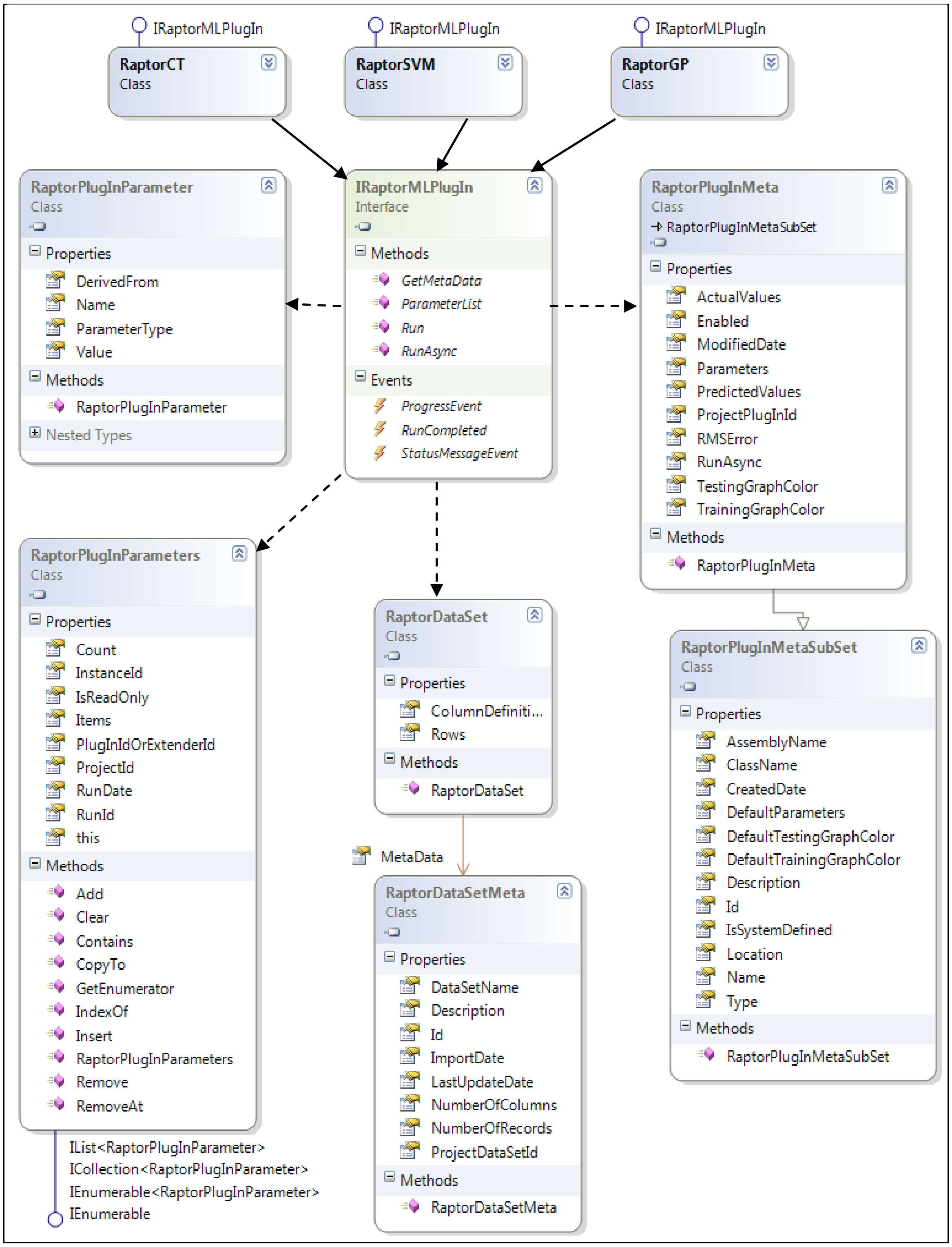


Figure 10 - Machine Learning Libraries Class Diagrams

Raptor Project

To perform analysis on the mortgage data, a Raptor project was created. A Raptor project consists of the following elements:

- One or more datasets (datasets may be joined or unioned).
- One or more ML plug-in.
- Zero or more Data Extenders.

This section defines, by flow chart, the steps to be executed for building a Raptor project. These steps assume that the database has already been populated and scrubbed, and plug-Ins for ML1 - ML3 have been registered. The flow chart in Figure 11 maps the basic flow while Appendix G illustrates the relevant screens~~.~~

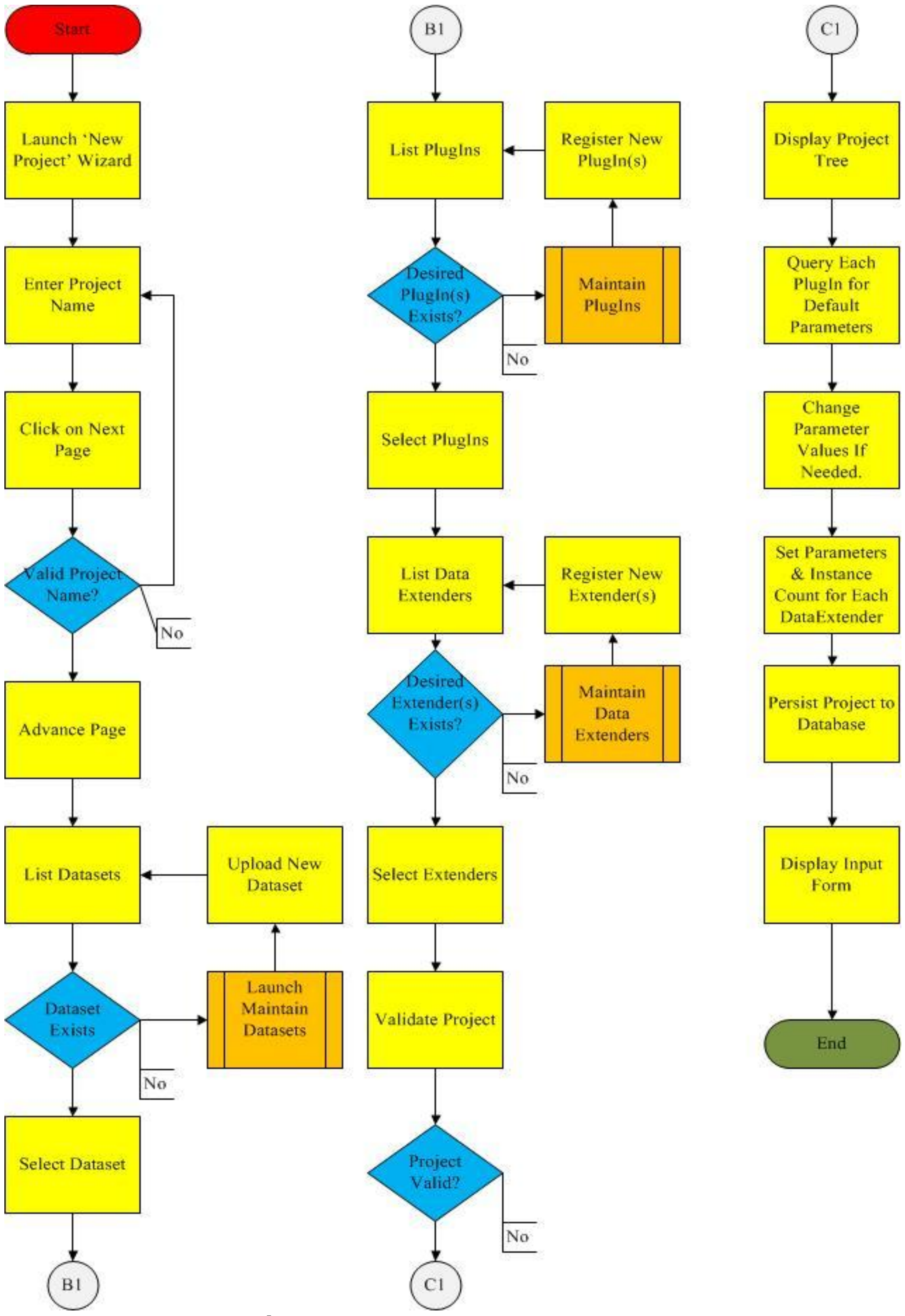


Figure 11 - New Project Flow

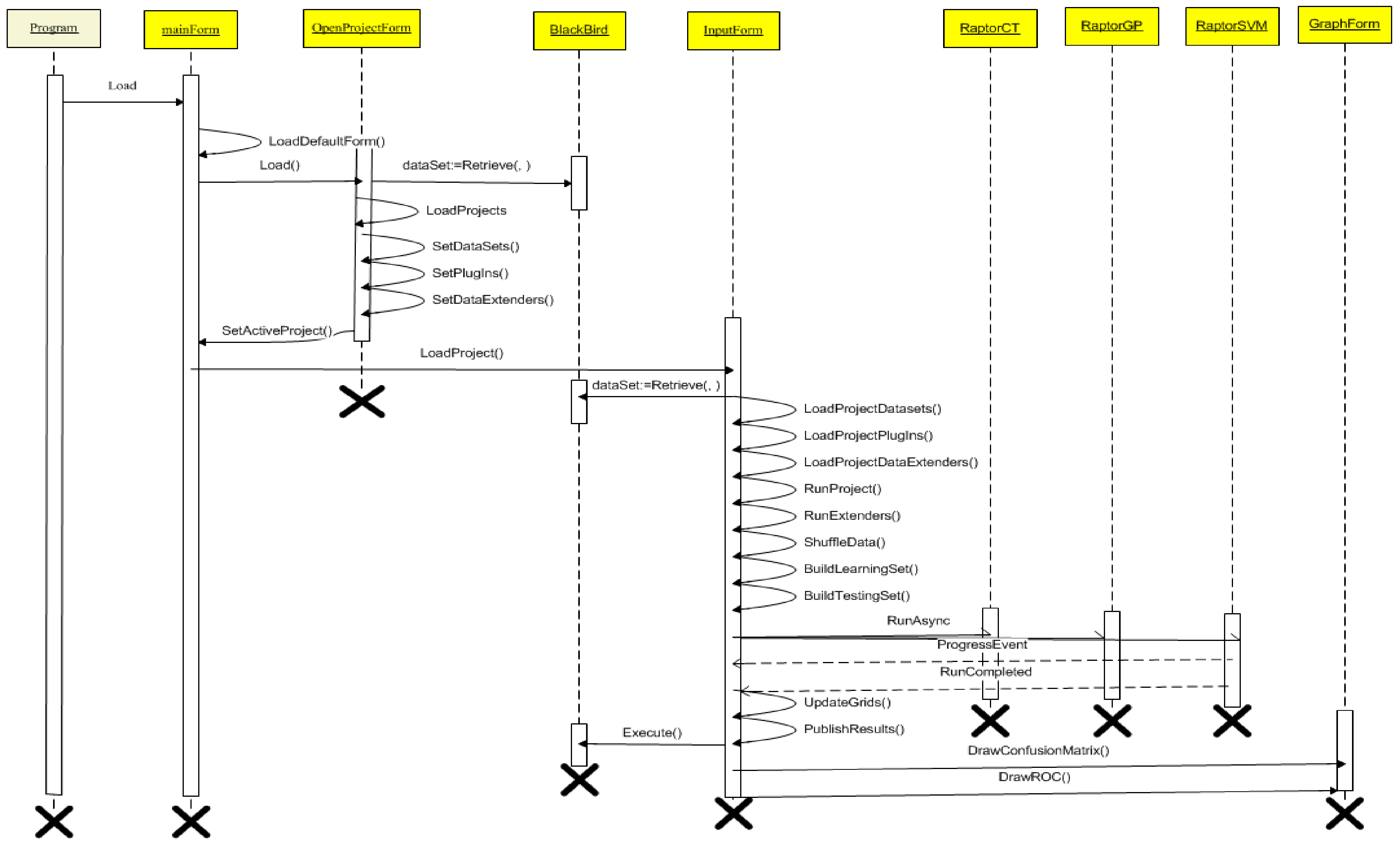


Figure 12 - High Level Sequence Diagram of Workbench Execution

Process

This section delineates the steps to be executed for a comparative run of all registered ML plug-Ins. Steps include the manual steps to be performed by the experimenter and those executed by Raptor (see Figure 12). These steps assume that a Raptor project has already been created.

1. Start Raptor.
2. From menu select 'Open Projects'.
3. Click on desired project.
4. Review plug-Ins.
    a. If all plug-Ins are not linked to project, right click and select 'Add Plugins' as illustrated in Figure 13.

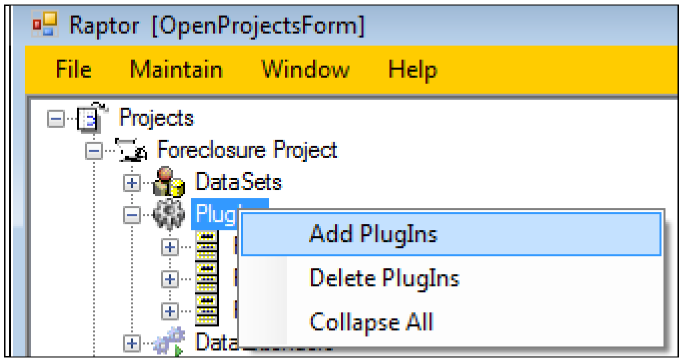
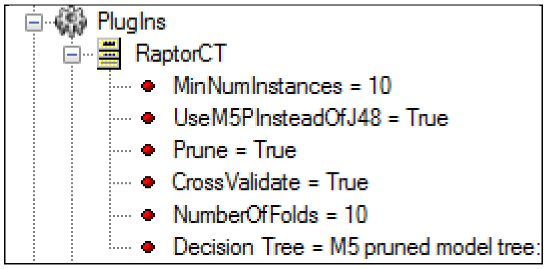


Figure 13 – Add Plug-Ins

    b. From the screen presented in Figure 36, select and save the desired plug-Ins and return to the projects screen (Figure 9).
5. Select ML1 plug-In which uses WEKA based implementation of C4.5 classification tree algorithm.
6. Set run parameters for ML1
    a. Set Minimum Number of Instances.
    b. Set Pruning to true.
    c. Set 'Cross Validate' to true.
        i. Set Number of Folds

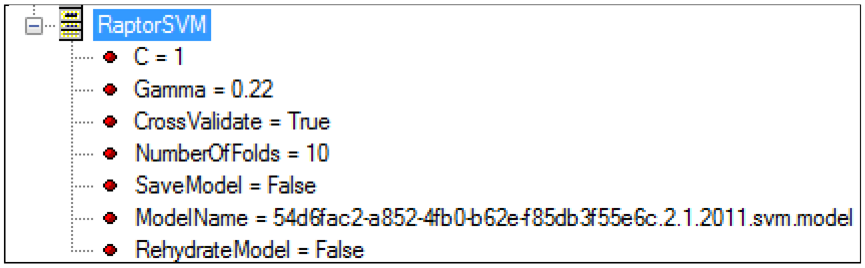
   d. Do not set Decision Tree. This value is returned by the ML engine.
   e. Set UseM5InsteadOfJ48 to true;



```
PlugIns
  RaptorCT
    MinNumInstances = 10
    UseM5PInsteadOfJ48 = True
    Prune = True
    CrossValidate = True
    NumberOfFolds = 10
    Decision Tree = M5 pruned model tree:
```

Figure 14 – ML1 Parameters

7. Select ML2 plug-In which uses NTU's implementation of SVM algorithm.
8. Set run parameters for ML2
   a. Kernel method is locked at RBF and cannot be changed from UI.
   b. Set penalization variable *C*, start at a number less than 1.
   c. Set bandwidth (σ^2) of kernel function. Start at 0.25.
   d. Set 'Cross Validate' to true.
      i. Set Number of Folds, default is 10
   e. Set 'SaveModel' to false.
   f. Set 'Rehydrate Model' to false.
   g. Ignore ModelName parameter; this is set by the ML engine.



```
RaptorSVM
  C = 1
  Gamma = 0.22
  CrossValidate = True
  NumberOfFolds = 10
  SaveModel = False
  ModelName = 54d6fac2-a852-4fb0-b62e-f85db3f55e6c.2.1.2011.svm.model
  RehydrateModel = False
```

Figure 15 – ML2 Parameters

9. Select ML3 plug-In which uses ECJ19/Dudley MPSR implementation of genetic program.
10. Set run parameters for ML3.
   a. Set 'BuildDepth'. Default is 6.
   b. Set 'MinimumAcceptableFitness' to zero.
   c. Set 'MaximumExecutionTime' to 30 minutes.

d. Instruction set is hard coded and cannot be set from UI. Testing done with following combination:

   i. Add
   ii. Subtract
   iii. Multiply
   iv. Divide
   v. <
   vi. >
   vii. =
   viii. AndAlso
   ix. OrElse
   x. Not
   xi. Power

e. Ignore ‘Mutation Threshold’.

f. Set ‘Population Size’. Default is 1000.

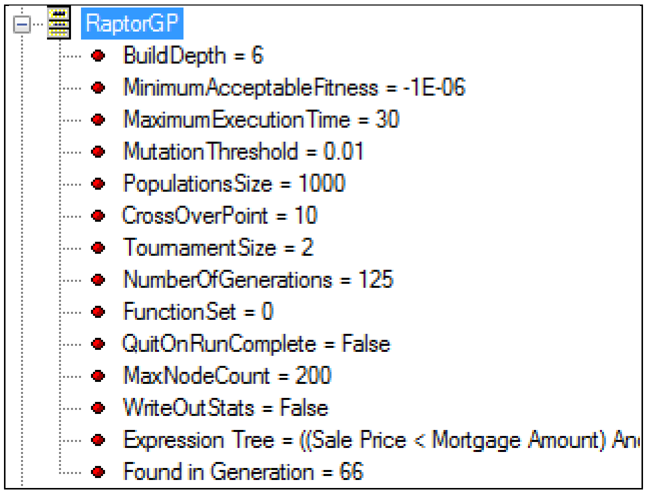
g. Set ‘Crossover Point’ to 10.

h. Set ‘Tournament Size’ to 2.

i. Set ‘Number of Generations’. Default is 100.

j. Set ‘Maximum Node Count’.

k. Ignore ‘Write out Stats’, ‘Expression Tree’ and ‘Found in Generation’. These are output variables set by the engine.



```
RaptorGP
  BuildDepth = 6
  MinimumAcceptableFitness = -1E-06
  MaximumExecutionTime = 30
  MutationThreshold = 0.01
  PopulationsSize = 1000
  CrossOverPoint = 10
  TournamentSize = 2
  NumberOfGenerations = 125
  FunctionSet = 0
  QuitOnRunComplete = False
  MaxNodeCount = 200
  WriteOutStats = False
  Expression Tree = ((Sale Price < Mortgage Amount) An
  Found in Generation = 66
```

Figure 16 – ML3 Parameters

11. Review Data Extenders.

a. If all extenders are not linked to project, right click and select ‘Add Extenders’ as illustrated in Figure 17.

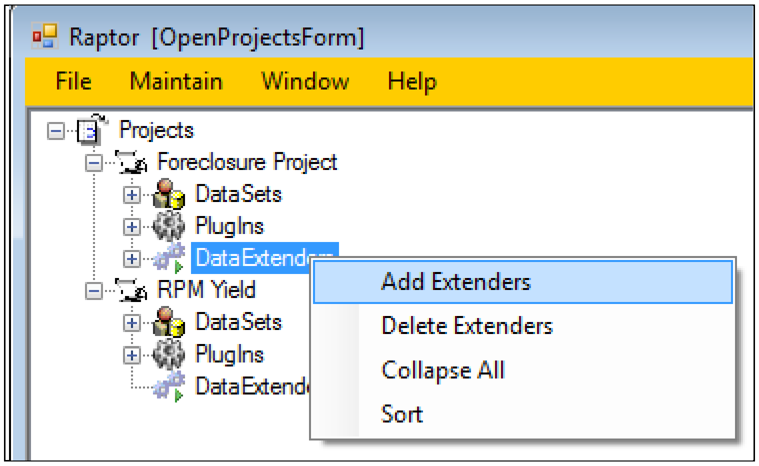
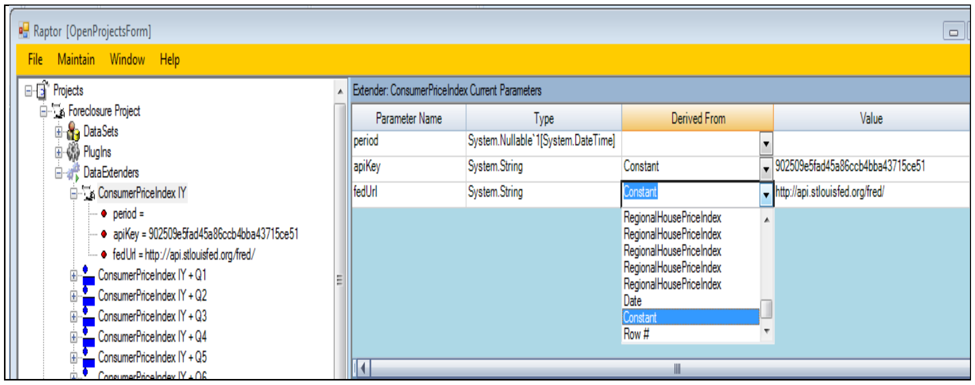


Figure 17 – Add Data Extenders

b. From the screen presented in Figure 37Figure **36**, select and save the desired extenders and return to the projects screen (Figure 9).

12. Review Data Extender parameters.

a. Parameters are either constants or the values of other columns as illustrated in Figure 18.



period =
apiKey = 902509e5fad45a86ccb4bba43715ce51
fedUrl = http://api.stlouisfed.org/fred/

Extender: ConsumerPriceIndex Current Parameters

Parameter Name	Type	Derived From	Value
period	System.Nullable`1[System.DateTime]		
apiKey	System.String	Constant	902509e5fad45a86ccb4bba43715ce51
fedUrl	System.String	Constant	http://api.stlouisfed.org/fred/

Figure 18 – Data Extender Parameter Values Derived From.

b. Set number of instances of selected extender and alias if necessary.

13. Click on the 'Run' button.

14. Raptor executes steps as illustrated in Figure 12.

15. Click on 'Output' button to view Confusion Matrix and ROC.

16. Click on the 'Publish' button to persist run to database.

17. From Input Form, modify plug-In parameters and then go to step 13. Perform this step *n* time.

This section summarizes all the development steps that were necessary to conduct this research.

1. Acquired all hardware and software including necessary licenses.
2. Built database for storing mortgage data.
3. Built ALFRED web service proxy and consumer class.
4. Acquired data from specified sources.
5. Imported data into database and scrubbed.
6. Set up App Fabric Cache service and tested. This replaces MemCache Server as was originally proposed.
7. Built work bench.
    a. Integrated data helper component to interface with database.
    b. Linked web service consumer class.
    c. Designed and built WCF interfaces and data types for SDK.
    d. Built grid, plug-in parameters and graphing forms.
    e. Implemented App Fabric Cache client interface.
8. Built plug-Ins for ML1 - ML3.
    a. Built unit tests.
    b. Ran tests with small datasets to verify accuracy.
    c. Tested dynamic loading and remote execution.
9. Built work bench unit tests.
10. Executed research process.

Chapter 4

Results

Introduction

This chapter presents and comments on the predictive performance of ML1 - ML3. Simple statistical analysis was used to determine base performance. Each plug-In was **concurrently** executed 20 times with varying input parameters and randomized training dataset. For each run, the input parameters, predictive results, and performance metrics were published to the Raptor database.

The primary metric used to compare the performance of ML1 - ML3 was **classification accuracy (*CA*).** This metric has been a standard comparison metric used in classifier induction studies (Perlich, Provost & Simonoff, 2003). Classification accuracy of an ML technique is the percentage of correctly predicted outputs after operation on a test dataset (Perlich, Provost & Simonoff). It is calculated by the sum of True Positives (*TP*) and True Negatives (*TN*) divided by number of records in the test dataset $N_{t,}$ *thus* $CA = (TP + TN)/N_t$. Classification accuracy results are presented in the format known as a Confusion Matrix.

K-fold cross validation was used to select the training and testing sets for ML1 - ML3. Cross-validation is a commonly used technique in machine learning research which uses all available examples as training and test examples (Bengio & Grandvalet, 2004). In cross-validation, initially the original sample is randomly partitioned into K subsamples. Of these K subsamples, a single subsample is retained for testing purposes, whiles K−1 subsamples are retained as training data (Bengio & Grandvalet). The cross-validation process is then repeated K times, whereby each of the K subsamples is used exactly once as validation data. Each ML engine implemented K-fold cross validation switches which were turned on for each of the 20 runs. In some runs K was varied in order to observe the effect on classification accuracy. For each run, 30% of the total dataset was randomly chosen and shuffled to produce the training set upon which K-Fold validation was performed.

ML1: Classification Tree

ML1 used WEKA's C4.5 classification engine, as the J48 engine is not suitable for handling numeric values. Of ML1's parameters, 'Minimum Number of Instances' was varied starting at 1, and progressed through to 200. Pruning was always set to true along with 'Cross Validate'. The run with the highest classification accuracy of 0.82 occurred with the parameters as illustrated in Table 10. Table 13 logs the classification accuracy and execution times for ML1.

ML1 outputted a set of 19 rules (Table 11) each of which points to a specific linear module (LM) that is used to predict foreclosure (Table 12). Figure 19 presents the rules as a classification tree.

Table 10 - Optimum parameters for ML1

Parameter Name	**Value**
MinNumberInstances	2
CrossValidate	True
NumberOfFolds	7
Prune	True

Table 11 - ML1 Generated Rules

ML1 Rules
If Mortgage Interest Rate Type = 0 A**nd** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 A**nd** Sale Price <= 121248 **And** Inflation IY + Q2 <= 3.25 **Then LM1** (10/0%).
If Mortgage Interest Rate Type = 0 **and** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 **And** Sale Price <= 121248 **And** Inflation IY + Q2 > 3.25 **And** ConsumerPriceIndex IY + Q10 <= 203.7 **And** Market Value <= 91246 **Then LM2** (4/0%).
If Mortgage Interest Rate Type = 0 **and** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 **And** Sale Price <= 121248 **And** Inflation IY + Q2 > 3.25 **And** ConsumerPriceIndex IY + Q10 <= 203.7 **And** Market Value > 91246 **And** ConsumerPriceIndex IY + Q9 <= 202.85 **Then LM3** (4/0%).
If Mortgage Interest Rate Type = 0 **and** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 **And** Sale Price <= 121248 **And** Inflation IY + Q2 > 3.25 **And** ConsumerPriceIndex IY + Q10 <= 203.7 **And** Market Value > 91246 **And** ConsumerPriceIndex IY + Q9 > 202.85 **Then LM4** (2/0%).
If Mortgage Interest Rate Type = 0 **and** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 **And** Sale Price > 121248 **And** Inflation IY + Q2 > 3.25 **And** ConsumerPriceIndex IY + Q10 > 203.7 Then **LM5** (4/0%).
If Mortgage Interest Rate Type = 0 A**nd** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 A**nd** Sale Price > 121248 **And** Mortgage Amount <= 127200 **Then LM6** (11/0%).
If Mortgage Interest Rate Type = 0 A**nd** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 A**nd** Sale Price > 121248 **And** Mortgage Amount > 127200 **And** ConsumerPriceIndex IY + Q5 <= 195 **Then LM7** (5/94.464%).
If Mortgage Interest Rate Type = 0 A**nd** Market Value <= 144107.5 **And** Mortgage Amount <= 155250 A**nd** Sale Price > 121248 **And** Mortgage Amount > 127200 **And** ConsumerPriceIndex IY + Q5 > 195 **Then LM8** (5/0%).
If Mortgage Interest Rate Type = 0 A**nd** Market Value <= 144107.5 **And** Mortgage Amount > 155250 **Then LM9** (16/92.176%).
If Mortgage Interest Rate Type = 0 A**nd** Market Value > 144107.5 **Then LM10** (84/83.268%).

If Mortgage Interest Rate Type =1 **And** Sale Price <= 132983 **And** ConsumerSentiment IY + Q2 <= 92.7 **Then LM11** (16/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price <= 132983 **And** ConsumerSentiment IY + Q2 > 92.7 **Then LM12** (35/90.515%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 <= 84.8 **Then LM13** (41/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount <= 171000 **Then LM14** (24/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount > 171000 **And** ConsumerPriceIndex IY + Q9 <= 200.65 **Then LM15** (11/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount > 171000 **And** ConsumerPriceIndex IY + Q9 > 200.65 **And** RegionalHousePriceIndex IY + Q7 <= 432.63 **Then LM16** (9/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount > 171000 **And** ConsumerPriceIndex IY + Q9 > 200.65 **And** RegionalHousePriceIndex IY + Q7 > **And** 432.63 ConsumerPriceIndex IY + Q9 <= 204.626 And Sale Price <= 348912.5 **Then LM17** (9/76.66%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount > 171000 **And** ConsumerPriceIndex IY + Q9 > 200.65 **And** RegionalHousePriceIndex IY + Q7 > **And** 432.63 ConsumerPriceIndex IY + Q9 <= 204.626 And Sale Price > 348912.5 **Then LM18** (3/0%).

If Mortgage Interest Rate Type =1 **And** Sale Price > 132983 **And** ConsumerSentiment IY + Q10 > 84.8 **And** Mortgage Amount > 171000 **And** ConsumerPriceIndex IY + Q9 > 200.65 **And** RegionalHousePriceIndex IY + Q7 > **And** 432.63 ConsumerPriceIndex IY + Q9 > 204.626 **Then LM19** (6/0%).

Table 12 - Linear Models Generated by ML1

Linear Models	
LM1 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0.0118 * ConsumerPriceIndex IY + Q10 - 0.2716 * Inflation IY + Q2 + 3.7789,0)	LM2 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0.0105 * ConsumerPriceIndex IY + Q9 - 0.0101 * ConsumerPriceIndex IY + Q10 - 0.2341 * Inflation IY + Q2 + 1.371,0)
LM3 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0116 * ConsumerPriceIndex IY + Q9 - 0.0101 * ConsumerPriceIndex IY + Q10 - 0.2341 * Inflation IY + Q2 + 1.1226,0)	LM num: 4 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0118 * ConsumerPriceIndex IY + Q9 - 0.0101 * ConsumerPriceIndex IY + Q10 - 0.2341 * Inflation IY + Q2 + 1.0899,0)
LM5 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0087 * ConsumerPriceIndex IY + Q9 - 0.0101 * ConsumerPriceIndex IY + Q10 - 0.2341 * Inflation IY + Q2 + 1.6315,0)	LM6 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0106 * ConsumerPriceIndex IY + Q5 - 1.9975,0)

Linear Models	
LM7 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0169 * ConsumerPriceIndex IY + Q5 - 3.1838,0)	LM8 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.0169 * ConsumerPriceIndex IY + Q5 - 3.1584,0)
LM9 Foreclosure = Round(0.0126 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value + 0.5108,0)	LM10 Foreclosure = Round(0.0103 * Interest Rate - 0.0235 * Mortgage Interest Rate Type - 0 * Market Value - 0.0104 * ConsumerPriceIndex IY + Q11 + 2.1702,0)
LM11 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type + 0 * Market Value + 0.0046 * ConsumerSentiment IY + Q2 - 0.294,0)	LM12 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type + 0 * Market Value + 0.0029 * ConsumerSentiment IY + Q2 + 0.0465,0)

Linear Models	
LM13 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value + 0.0274,0)	LM14 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value + 0.0075,0)
LM15 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value - 0.0137,0)	LM16 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value - 0.0061 * ConsumerPriceIndex IY + Q9 + 0.0006 * RegionalHousePriceIndex IY + Q7 + 0.9495,0)
LM17 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value - 0.0085 * ConsumerPriceIndex IY + Q9 + 0.0004 * RegionalHousePriceIndex IY + Q7 + 1.5567,0)	LM18 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value - 0.0085 * ConsumerPriceIndex IY + Q9 + 0.0004 * RegionalHousePriceIndex IY + Q7 + 1.53,0)

Linear Models	
LM num: 19 Foreclosure = Round(0.0025 * Interest Rate - 0.0222 * Mortgage Interest Rate Type - 0 * Market Value - 0.0096 * ConsumerPriceIndex IY + Q9 + 0.0004 * RegionalHousePriceIndex IY + Q7 + 1.7478,0)	

Table 13 – Classification Accuracy for ML1

MinNumberInstances	Number of Folds	False Positive (%)	False Negative (%)	TP	FP	TN	FN	Classification Accuracy	Execution Time (sec)
1	5	13.734	8.870	82	62	459	96	0.77	10.9472
2	**7**	**7.015**	**3.868**	**74**	**27**	**499**	**98**	**0.82**	15.5627
3	5	8.160	3.290	70	23	492	114	0.8	13.1124
4	5	6.724	5.007	77	35	493	94	0.81	10.9778
5	10	8.155	3.290	70	23	492	114	0.8	19.186
6	10	5.866	8.584	97	60	460	82	0.8	14.9707
7	15	5.866	8.584	97	60	460	82	0.8	11.7468
8	5	8.011	8.155	66	57	464	112	0.76	10.6973
9	20	5.866	8.441	97	59	461	82	0.8	15.4353
12	5	8.226	8.011	63	56	465	115	0.76	12.8809
15	20	10.658	0.858	17	6	527	149	0.78	37.097
20	25	10.515	1.001	19	7	526	147	0.78	39.7959
22	5	8.441	7.439	60	52	469	118	0.76	3.4029
25	10	11.159	0.715	10	5	528	156	0.77	14.8202
30	15	9.657	6.295	53	44	467	135	0.74	19.0104
35	10	9.728	6.581	44	46	473	136	0.74	19.8101
40	10	9.871	6.581	42	46	473	138	0.74	12.8107
55	15	10.014	5.866	40	41	478	140	0.74	8.4886
60	20	12.303	0.572	13	4	510	172	0.75	20.5286
200	20	12.732	0.000	0	0	521	178	0.75	7.4209
	Mean:	9.135	5.100	54.55	35.65	485.85	122.9	0.78	15.935
	Standard Deviation:	2.308	3.145	30.431	21.984	24.941	29.203	0.028	8.833
	Min:	5.866	0	0	0	459	82	0.74	3.403
	Max:	13.734	8.870	97	62	528	178	0.82	39.80

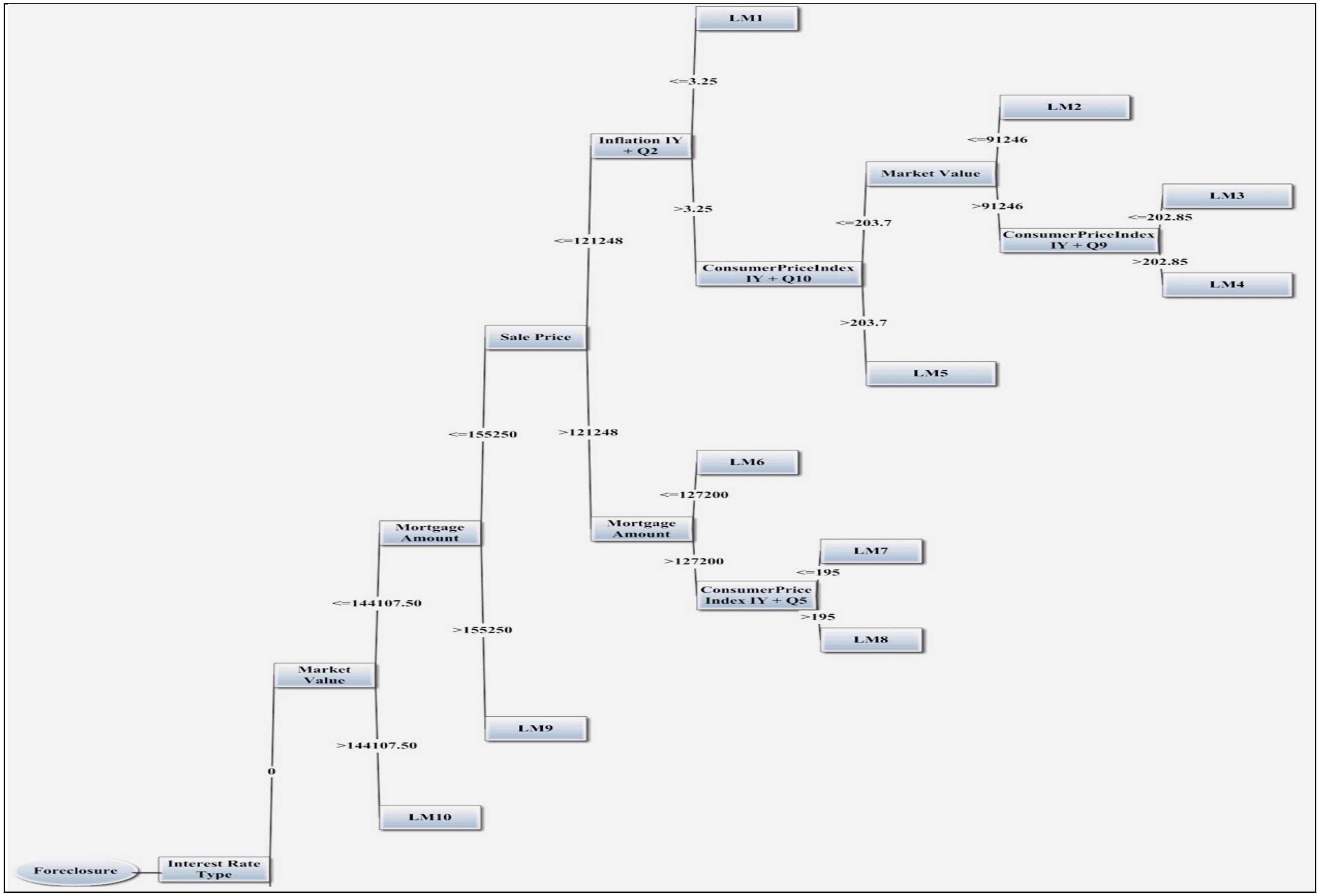


Figure 19 – Classification Tree for ML1 (Interest Rate Type=0)

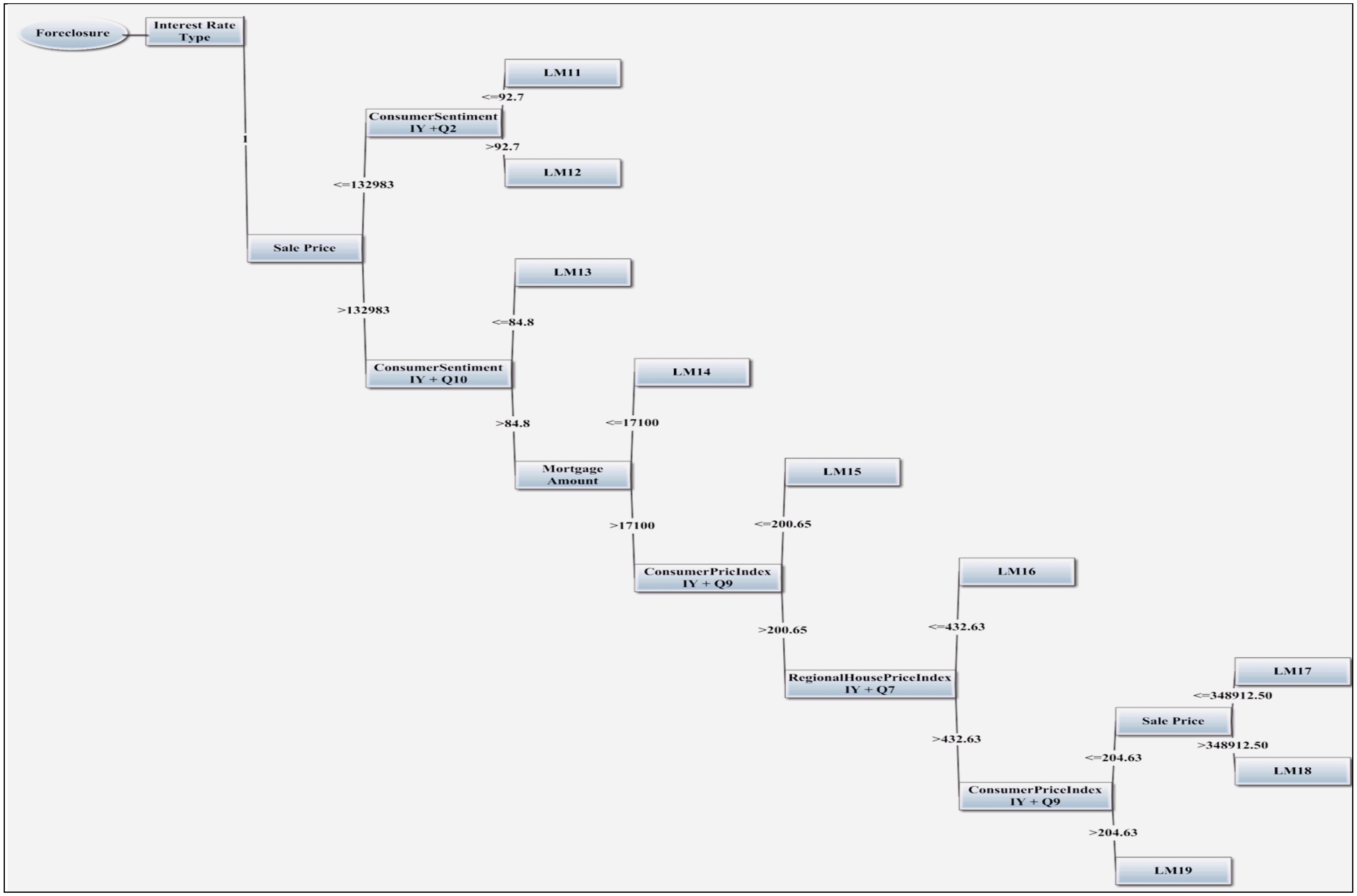


Figure 20 – Classification Tree for ML1 (Interest Rate Type =1)

ML2: Support Vector Machine

ML2 had two main input parameters that affected classification accuracy. These parameters are ‘C’ and ‘Gamma’. As with the other plug-Ins, ‘Cross Validation’ was always turned on. Varying the ‘Number of Folds’ did not have much influence on the classification accuracy of ML2. The parameter ‘Gamma’ was varied through 0.10 to 0.25 and C was varied through 0.70 to 1. The run with the highest percentage of correct predictions (84%) on the test dataset occurred with the parameters as illustrated in Table 14. Table 15 logs the classification accuracy and execution times for ML2. The mean ‘Classification Accuracy’ was 0.796 with a Standard Deviation of 0.038. The average execution time was 9.099 seconds. ML2 displayed a consistent ability to correctly predict all positive values.

Table 14- Optimum parameters for ML2

Parameter Name	**Value**
C	1
Gamma	0.4

Table 15 – Classification Accuracy for ML2

C	Gamma	False Positive (%)	False Negative (%)	TP	FP	TN	FN	Classification Accuracy	Execution Time
0.05	0.25	0	24.866	0	0	525	174	0.75	1.7088
0.4	0.25	0	25.724	0	0	519	180	0.74	2.6728
0.5	0.25	0	25.724	0	0	519	180	0.74	1.7088
0.65	0.65	0	24.294	0	0	529	170	0.76	2.3186
0.65	0.25	0	26.438	0	0	514	185	0.74	2.3389
0.7	0.7	0	26.438	0	0	514	185	0.74	139.0241
0.75	0.3	0	20.408	53	0	514	132	0.81	2.2817
0.75	0.3	0	23.436	0	0	535	164	0.77	1.7258
0.75	0.25	0	18.882	59	0	519	121	0.83	1.9063
0.8	0.8	0	20.408	53	0	514	132	0.81	8.2048
0.9	0.3	0	18.297	44	0	535	120	0.83	1.7058
1	0.25	0	20.224	57	0	512	130	0.81	2.4041
1	0.3	0	18.466	50	0	529	120	0.83	1.6837
1	**0.4**	**0**	**17.336**	**48**	**0**	**538**	**113**	**0.84**	1.9265
1	0.45	0	18.116	54	0	528	117	0.83	1.7018
1	0.05	0	18.369	52	0	528	119	0.83	1.7048
1	0.15	0	22.229	43	0	510	146	0.79	1.6978
1	0.2	0	18.215	52	0	529	118	0.83	1.7519
1	0.25	0	18.882	59	0	519	121	0.83	1.7118
1.1	0.25	0	20.224	57	0	512	130	0.81	1.8091
	Mean:	0	21.349	34.05	0	522.10	142.85	0.796	9.099
	Standard Deviation:	0	3.209	25.958	0	8.759	26.939	0.038	30.615
	Min:	0	17.336	0	0	510.00	113.00	0.740	1.684
	Max:	0	26.438	59.00	0	538.00	185.00	0.840	139.024

ML3: Genetic Programming Symbolic Regression

ML3 used symbolic regression via genetic programming to build an optimal solution in the form of an expression tree. An expression tree is executable code represented as a data structure. ML3 has eleven input parameters, four of which significantly varied the results. Of these four, ‘Number of Generations’ was varied starting at 25, and progressing through to 125. With the number of generations set between 100 and 125, the GP was more likely to find an optimal solution. Population size was also a sensitive input parameter that affected results when set below 400. Population size was varied between 200 and 1000. The run with the highest percentage of correct predictions (99.49%) on the test dataset occurred with the parameters as indicated in Table 16. Table 18 logs the classification accuracy and execution times for ML3.

Table 16 - Optimum parameters for ML3

Parameter Name	**Value**
BuildDepth	6
PopulationSize	600
NumberOfGenerations	125
MaxNodeCount	200

The foreclosure expression generated for the best run was treated as follows:

Let

RegionalHousePriceIndex for quarter x= $\mathrm{RHPI}_{\mathrm{Qx}}$,

RegionalHomeOwnershipRate for quarter x = $\mathrm{RHOR}_{\mathrm{Qx}}$,

ConsumerPriceIndex for quarter x = $\mathrm{CPI}_{\mathrm{Qx}}$,

PrimeRate for quarter x = $\mathrm{PR}_{\mathrm{Qx}}$

Inflation for quarter x = I_{Qx}

ConsumerSentiment for quarter x = $\mathrm{CS}_{\mathrm{Qx}}$

And

OrElse – Logical short circuit for ‘*Or’* operator, such that, if the result of the first expression evaluated determines the final result of the operation, there is no need to evaluate the second expression.

AndAlso – Logical short circuit for ‘*And’* operator. Examples:

Table 17 – OrElse and AndAlso

Expression1 is	**Operator**	**Expression2 is**	**Result is**
True	AndAlso	True	True
True	AndAlso	False	False
False	AndAlso	(not evaluated)	False
True	OrElse	(not evaluated)	True
True	OrElse	False	False
False	OrElse	True	False
False	False	False	False

Then

Equation 1 - Foreclosure Formula

Foreclosure = (Sale Price < Mortgage Amount)

OrElse $(((CS_{Q6} – I_{Q0}) - CPI_{Q4}) > (CPI_{Q3} \wedge (((I_{Q9} + ((((\text{Sale Price} * RHOR_{Q3}) – (CPI_{Q7} / PR_{Q7})) - I_{Q7}) * RHOR_{Q8})) \wedge (((\text{Sale Price} / (I_{Q0} / (PR_{Q9} / (CPI_{Q2} - RHPI_{Q10})))) \wedge CS_{Q8})^{CS}{}_{Q10})) - (CS_{Q1}{}^{\text{Sale Price}}))))))$

AndAlso $(\text{Term} > PR_{Q3})))$

The Expression Tree for this formula is presented in Figure 21

Table 18 – Classification Accuracy for ML3

Build Depth	Population Size	NumberOf Generations	MaxNode Count	False Positive (%)	False Negative (%)	TP	FP	TN	FN	Classification Accuracy	Execution Time (sec)
2	200	50	100	3.433	25.179	13	24	486	176	0.71	56.5785
2	200	50	100	0.286	24.893	5	2	518	174	0.75	52.0811
4	300	50	100	5.866	11.588	98	41	479	81	0.83	83.4023
4	350	60	150	0.858	0.000	179	6	514	0	0.99	121.4955
6	375	70	175	4.149	21.602	28	29	491	151	0.74	123.0094
8	400	70	200	0.715	24.607	7	5	515	172	0.75	136.984
10	425	80	225	0.429	24.607	7	3	517	172	0.75	160.8952
10	425	80	225	0.429	23.748	3	3	527	166	0.76	218.9903
10	425	80	225	1.001	0.000	160	7	532	0	0.99	310.8253
10	450	90	250	1.001	0.000	175	7	517	0	0.99	218.7959
10	475	100	275	4.149	20.744	30	29	495	145	0.75	248.2699
10	500	110	300	2.003	23.319	12	14	510	163	0.75	269.4796
6	500	125	200	0.572	0.000	175	4	520	0	0.99	310.3154
8	500	125	300	0.572	25.036	4	4	516	175	0.74	274.3491
6	600	125	**200**	**0.572**	**0.000**	**179**	**4**	**516**	**0**	**0.995**	370.7536
6	700	125	200	0.572	0.000	179	4	516	0	0.99	385.4787
8	750	125	200	1.574	22.747	13	11	516	159	0.76	405.4563
10	500	120	300	1.288	21.459	22	9	518	150	0.77	335.2664
10	1000	120	400	0.715	0.286	170	5	522	2	0.99	637.8624
8	1000	120	300	0.715	0.000	172	5	522	0	0.99	610.7911
			Mean:	1.545	13.491	81.550	10.800	512.350	94.300	0.849	266.554
			Standard Deviation:	1.574	11.627	79.753	11.000	13.735	81.270	0.120	163.424
			Min:	0.286	0.000	3.000	2.00	479.000	0.000	0.710	52.081
			Max:	5.866	25.036	179.000	41.000	532.000	175.000	0.995	637.862

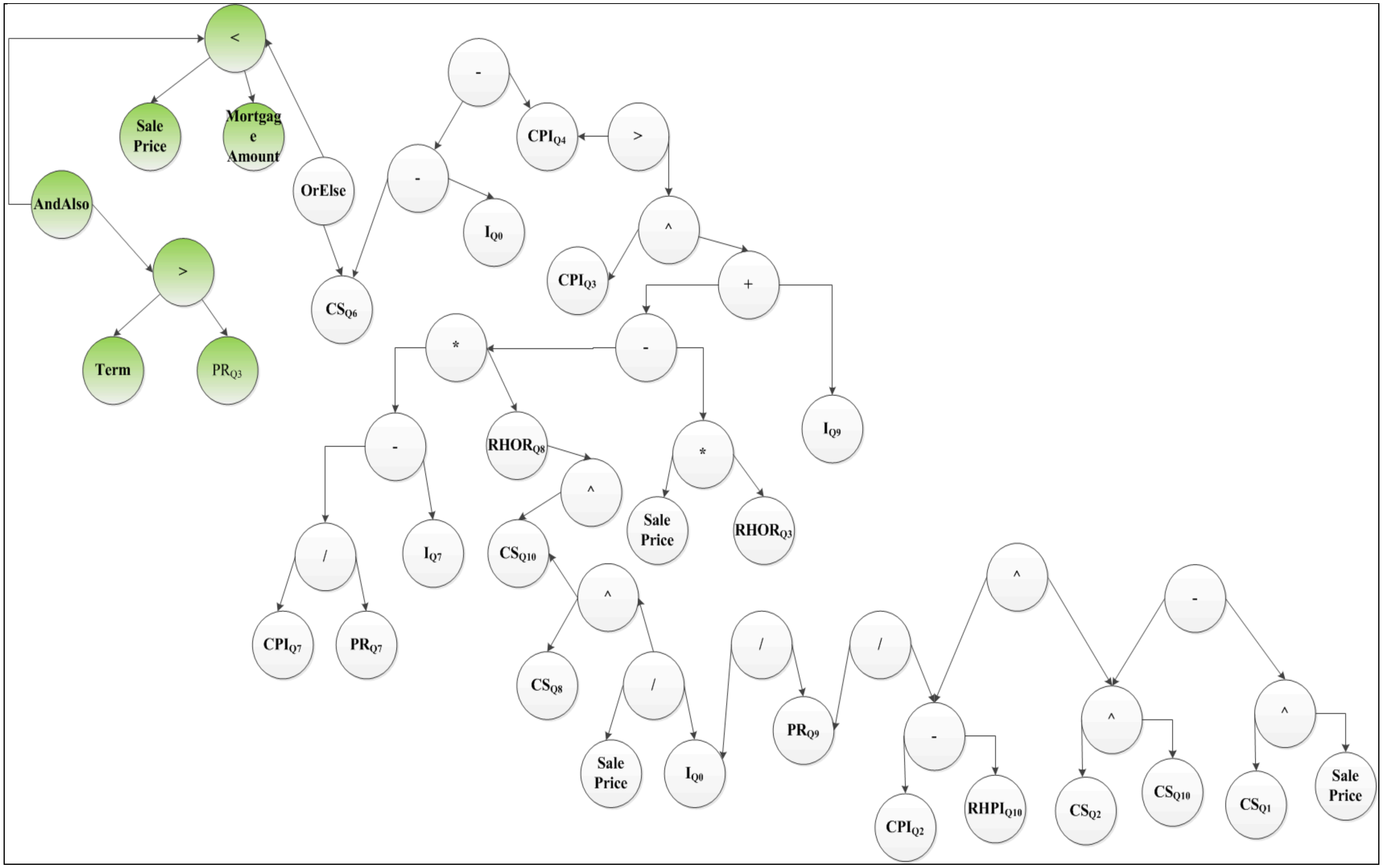


Figure 21 – Expression Tree of ML3 Optimal Solution

Summary

This chapter focused on presenting the performance results of ML1 - ML3. Of these engines, ML3 had the highest classification accuracy and hit the 90+% mark on several occasions. ML3 was significantly slower that ML1 or ML2, and also had the widest range of results with the lowest being in the low 70s. ML3 had the most input parameters, all of which demonstrated a significant effect on classification accuracy.

ML1s performance was disappointing and it is unclear whether this was a result of not discretizing input variables other than the output. WEKA was designed as a monolithic machine learning application at a time when component design was not widely used. As such, WEKA does not expose an easily workable API and depends on text files to set run-time parameters. Also, the documentation does not clearly indicate how certain tasks, like discretization, are performed. It would be interesting to see how ML1 performs with an independent classification tree engine.

ML1 & ML3 commonly indicated that the following variables were significant for predication:

1. RegionalHousePriceIndex
2. ConsumerPriceIndex
3. Inflation

From the perspective of consistency, accuracy and speed, it would appear that ML2 is the best choice for developing a foreclosure prediction model. This however is deceptive, because unlike the other engines, ML2 does not output any useable artifact. ML2 was, however, the easiest to implement and use. When combining all these factors it is hard to overlook ML2 as the primary choice for model development.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

This study focused on a difficult prediction task of significant societal import. The hypothesis that drove the study theorized that mortgage performance, projected over a three year period, could be predicted with a reasonable degree of accuracy. To support this hypothesis, the field of machine learning was researched and three suitable prediction algorithms were identified. The ML algorithms were supported by a purpose built workbench which managed the execution of the ML engines. The results were better than expected, with each algorithm scoring greater that 75% classification accuracy and in one case the high 90s%. Given these performance figures, it is quite sufficient to state that the hypothesis was positively supported by the research outcome.

Implications

The primary implication of this study is that it has the potential to stir additional research interest as identified in ‘Recommendations’. Furthermore, it is hoped that other researchers attempt to reproduce the results herein by using other ML algorithms. Finally, it is hoped that this study advances the understanding of machine learning algorithms and their effectiveness in prediction tasks in general.

Recommendations

Based on the findings of the research conducted within, the following recommendations are made:

1. Expand the dataset to include regions beyond South Florida and re-execute ML1 - ML3 on this expanded dataset.
2. Add, if possible, relevant psychometric variables to the dataset. Examples of such variables are Religion, Ethnicity and Occupation.
3. Continue the development of the Raptor workbench with the goal of eliminating dependencies on WEKA, ECJ*x* and other heavyset libraries.
4. Include ROC analysis and automatic calculation of Area Under ROC.
5. Expand the machine learning techniques to include Artificial Neural Networks and hybrid methods.
6. Expand the mortgage projection out to at least 5 years.
7. Seek additional macroeconomic variables and eliminate those which have little or no impact on the prediction task.
8. Contrast performance of ML1 - ML3 against logistic regression.
9. Expand the output to include 'Refinance' and 'Sell with Profit'.

Summary

This paper focused on the comparison of machine learning techniques in the problem domain of foreclosure prediction. The fundamental hypothesis was that given a dataset of mortgages, machine learning techniques could be used to forecast the mortgages' performance over a three year period. The machine learning techniques used were Classification Trees (ML1), Support Vector Machines (ML2) and Genetic Programming (ML3).

The dataset of mortgages was focused on the Tri-County (Dade, Broward and Palm Beach counties) area of South Florida. The dataset included Mortgage Amount, Sale Price, Market Value, Mortgage date, and Interest Rate. Macroeconomic indicators were used to expand the dataset horizontally and were measured quarterly. Chosen indicators included

1. Regional Per Capita Income
2. Regional Home Ownership Rate
3. Unemployment Rate
4. Consumer Price Index
5. Inflation
6. Prime Rate

A workbench was created in order to manage the dataset and record the performance results of ML1 - ML3. The workbench was designed using an SOA architecture which permitted monolithic or cloud based deployment. For extensibility, ML1 - ML3 were designed as plug-Ins. ML1 was based on the C4.5 engine of the WEKA system (Holmes, Donkin & Witten, 1997). ML2 used LibSVM by National Taiwan University (Chang & Lin, 2009). ML3 used George Mason University's ECJ*x* (Luke et al., 2008) and Dudley's (2011) MPSR library.

The primary metric used to compare the performance of ML1 - ML3 was classification accuracy**.** This metric has been a standard comparison metric used in classifier induction studies (Perlich, Provost & Simonoff, 2003). Classification accuracy of an ML technique is the percentage of correctly predicted outputs after operation on a test dataset (Perlich, Provost & Simonoff). It is calculated by the sum of True Positives (TP) and True Negatives (TN) divided by number of records in the test dataset N_t = (TP +

$TN)/N_t$. Classification accuracy results are presented in the format known as a Confusion Matrix.

The plug-Ins were run concurrently whiles varying their input parameters. A total of 20 runs were published to the workbench database. ML3 (Genetic Program) delivered the highest classification accuracy figure but also had the highest standard deviation. ML3 showed the highest sensitivity to change in its input parameters. ML2 (SVM) delivered the most stable performance and second highest classification accuracy. ML1's (Classification Tree) performance was disappointing but consistently demonstrated minor sensitivity to input variable changes. The following summarizes the performance of all plug-Ins.

Table 19 – Summary Results

Plug-In Name	**Highest Classification Accuracy**	**Standard Deviation**
ML1	0.82	0.028
ML2	0.84	0.038
ML3	0.995	0.120

As part of the process, ML1 and ML3 generated artifacts which can be used as prediction models. ML1's classification tree consists of eighteen rules, each invoked dependent on the state of key input parameters. It is possible to improve classification accuracy by focusing on the rule which nets the largest part of the dataset. Likewise, ML3's expression tree can be explored and simplified to improve efficiency.

Appendices

A. Alfred Proxies

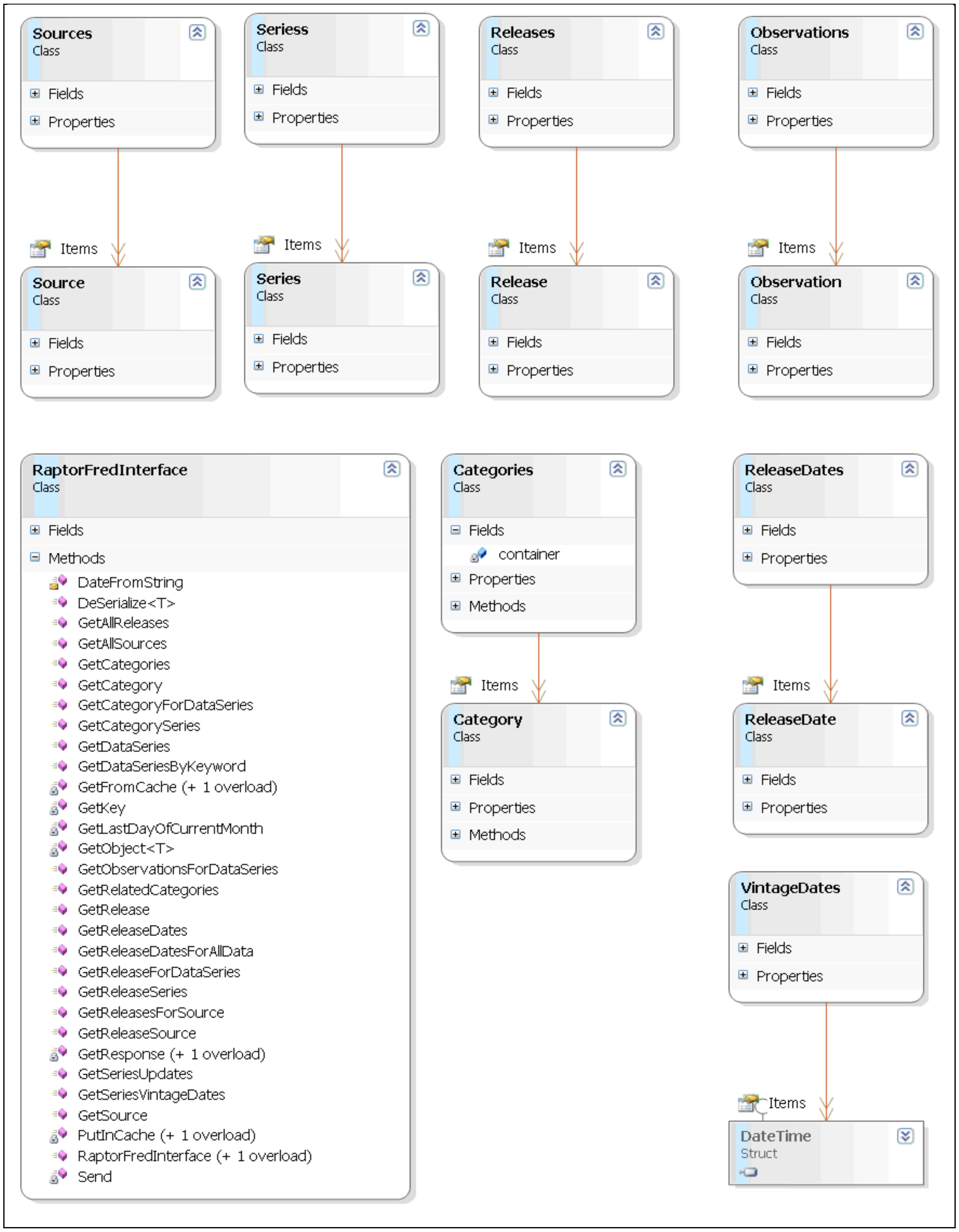


Figure 22 - High Level Class Diagram of ALFRED® Web Service Proxies

B. Raptor User Interface Class Diagrams

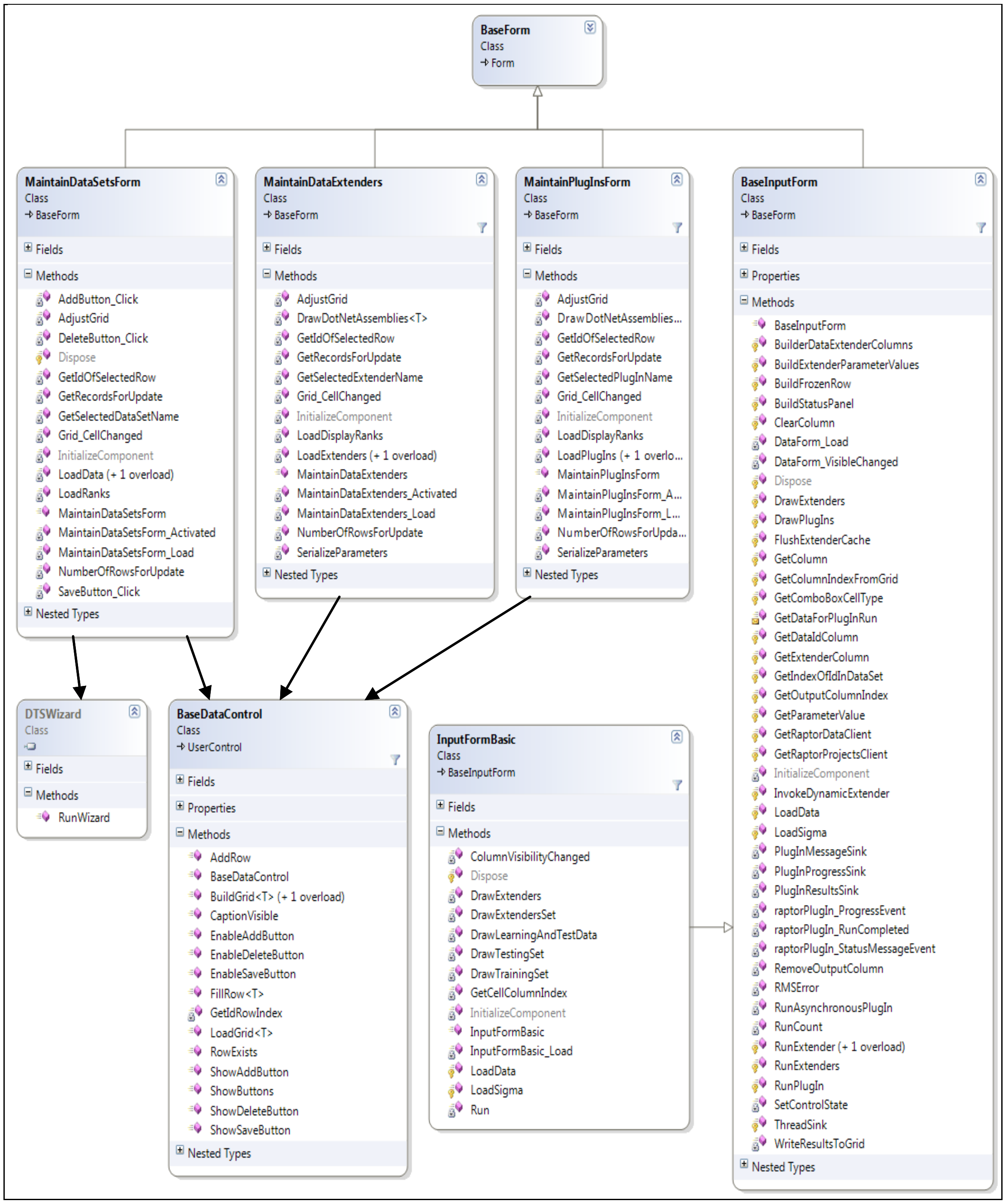


Figure 23 - Class Diagram of Raptor UI (a)

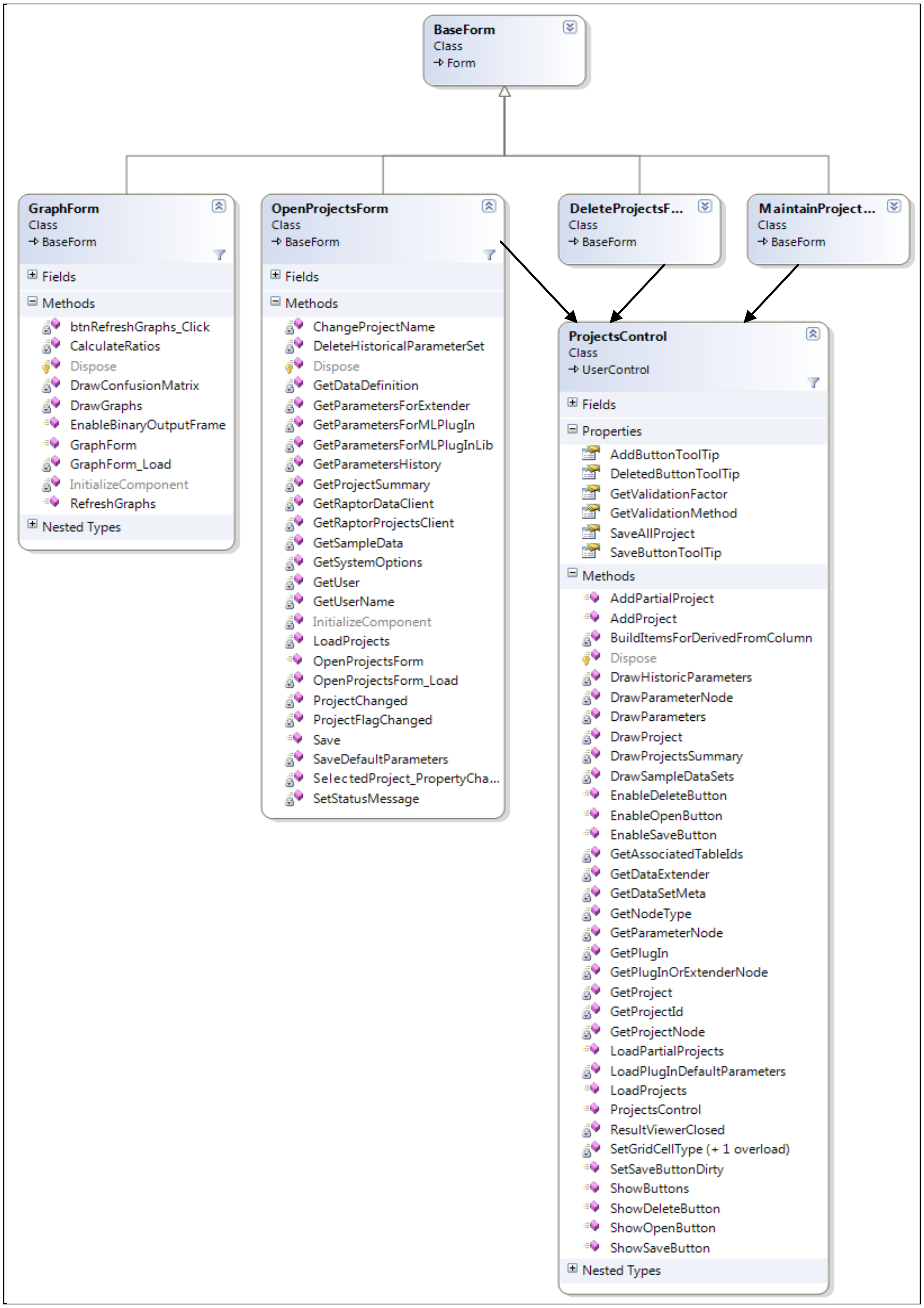


Figure 24 - Class Diagram of Raptor UI (b)

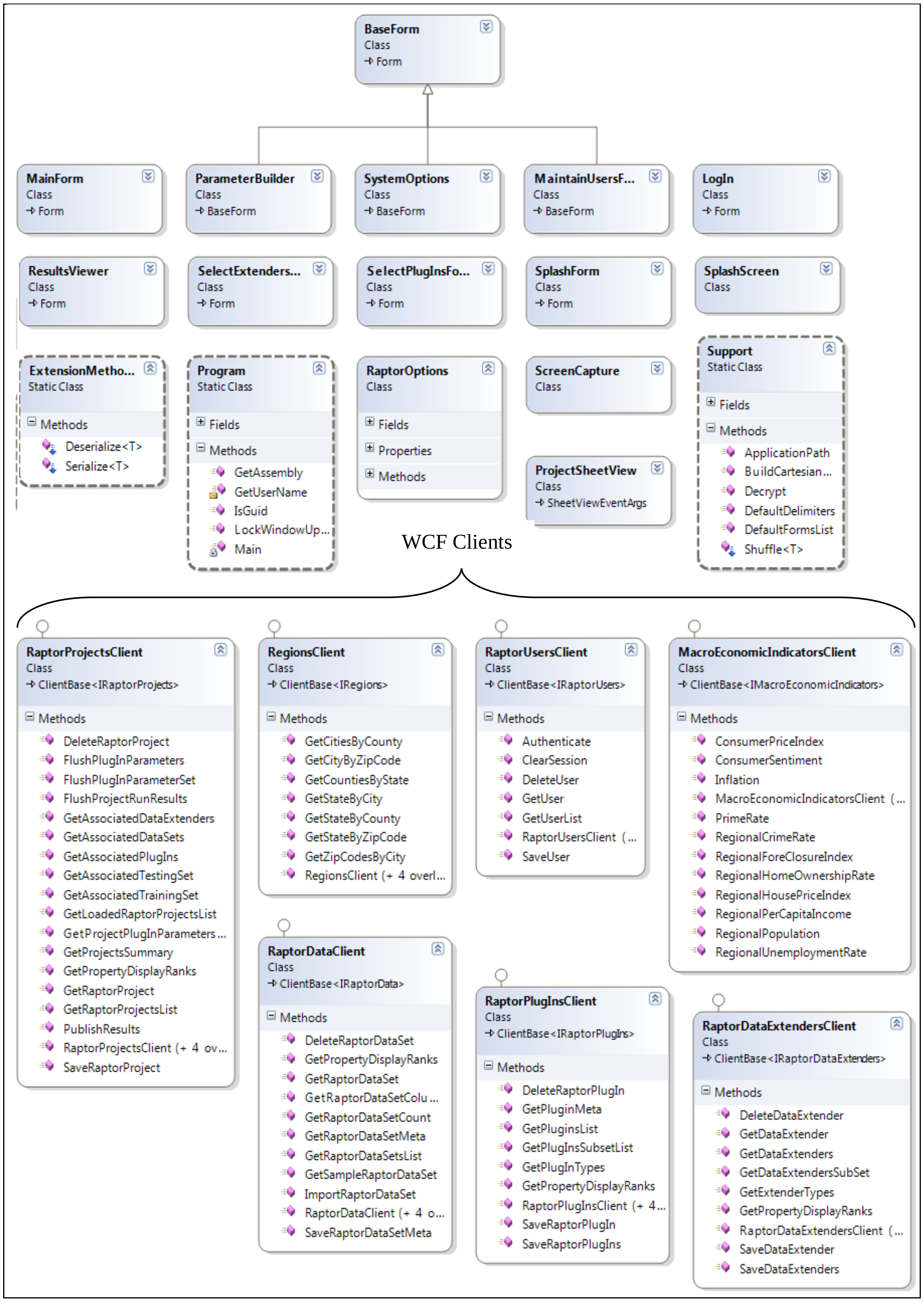


Figure 25 - Class Diagram of Raptor UI (c)

C. Raptor Services Class Diagrams

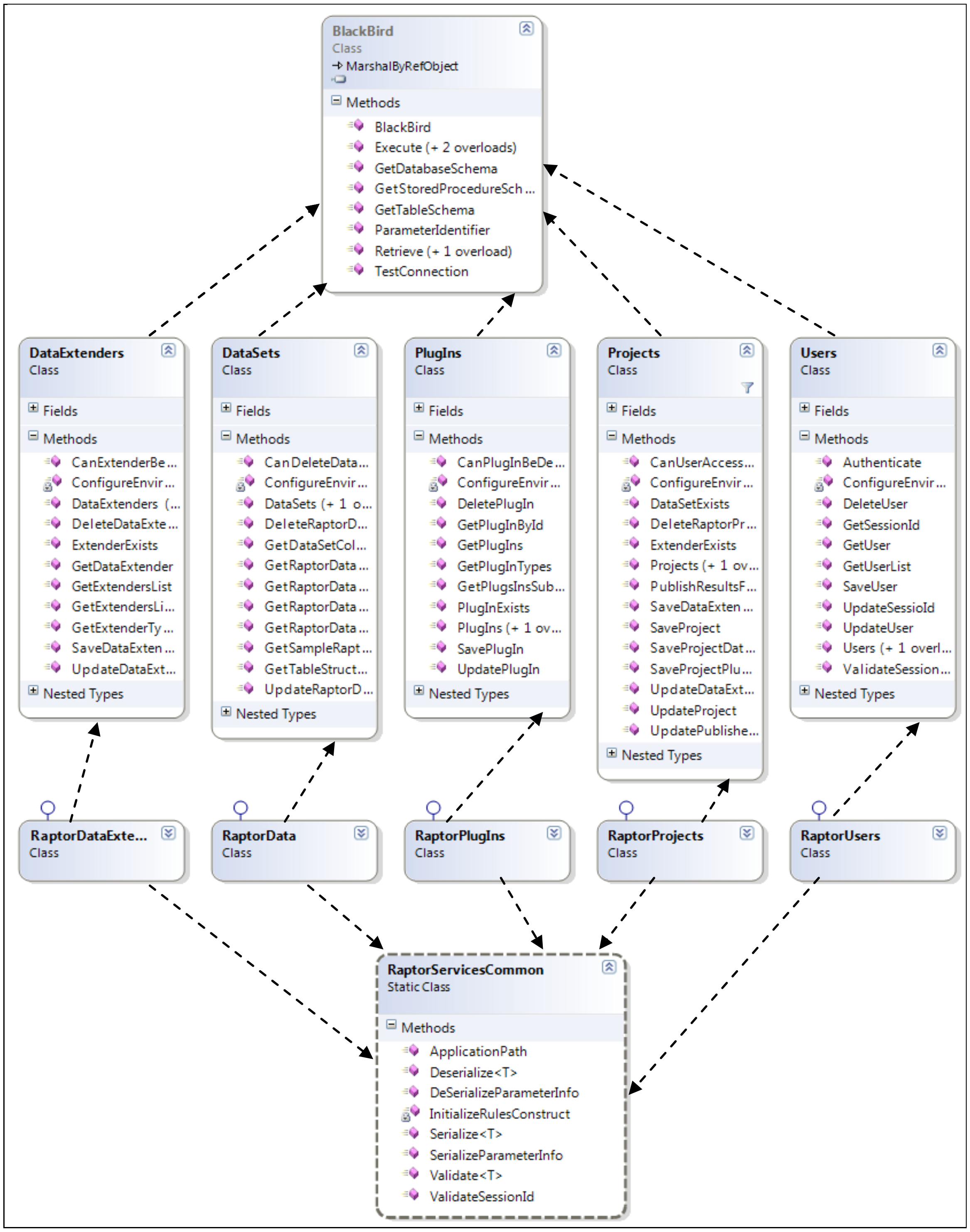


Figure 26 - Class Diagram of Raptor Services (a).

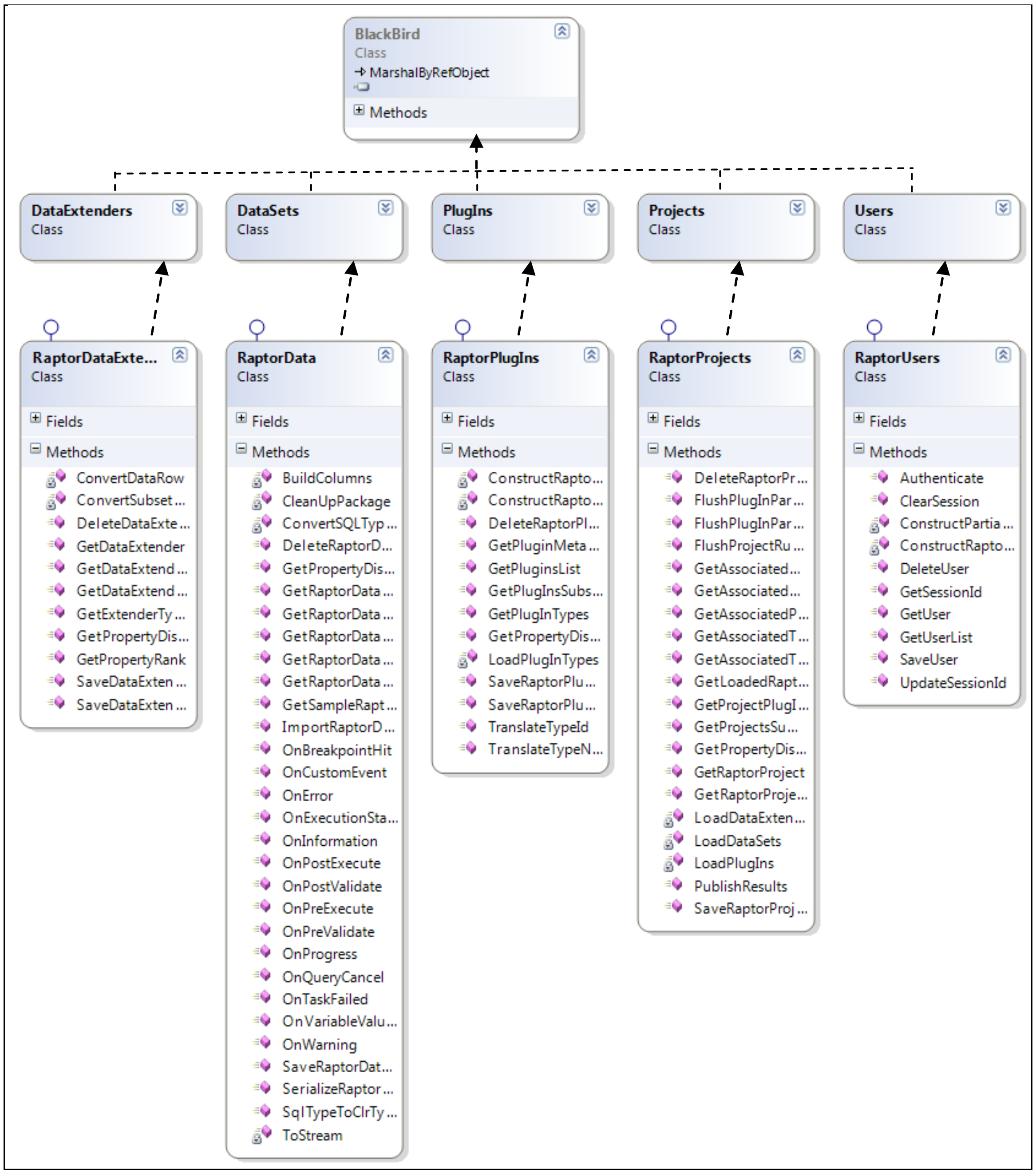


Figure 27 - Class Diagram of Raptor Services (b).

D. Raptor Machine Learning WCF Services Class Diagram

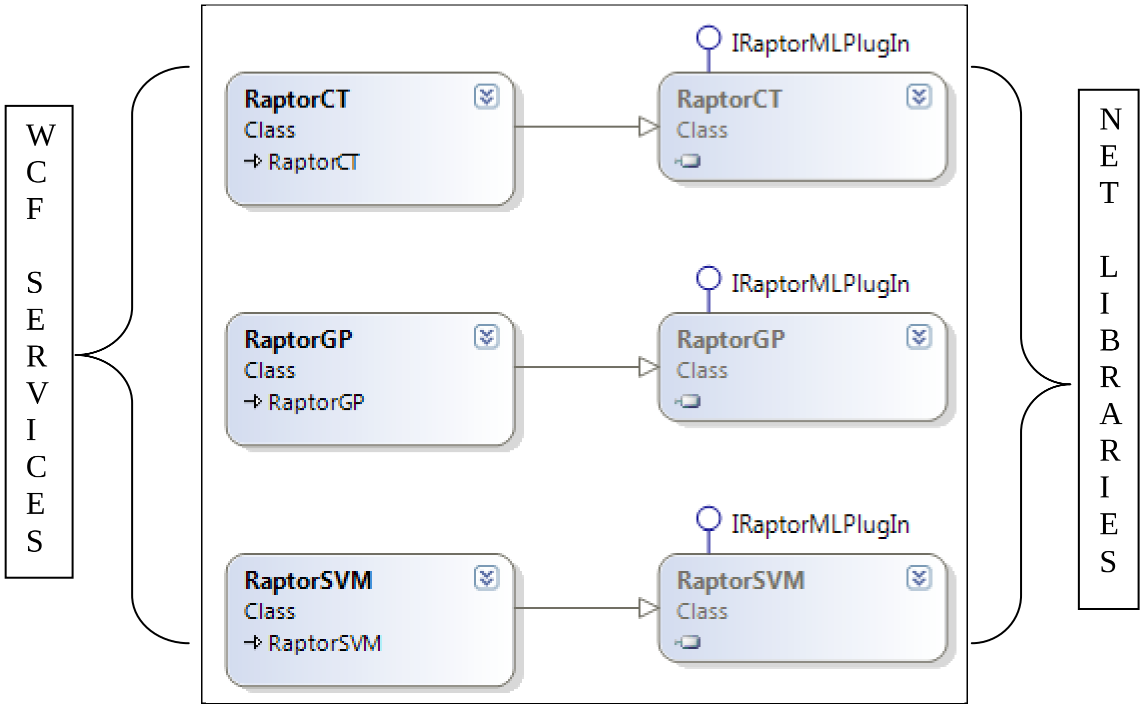
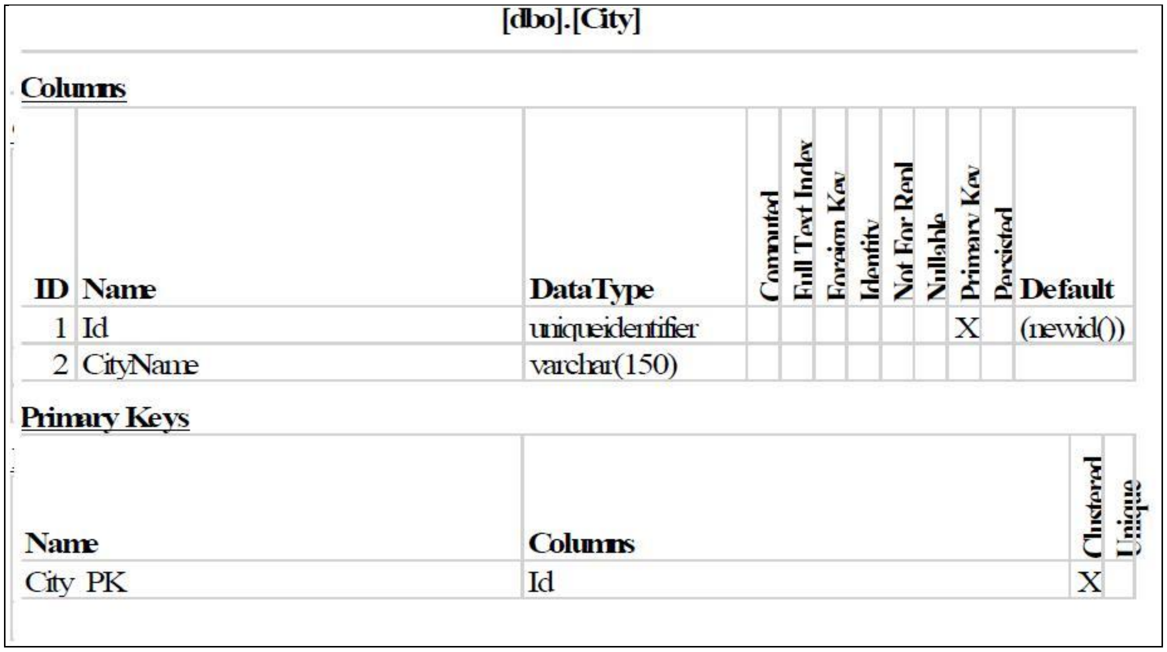


Figure 28 - Machine Learning WCF Services Class Diagram

E. Database Tables

Table 20 - City Table



[dbo].[City]

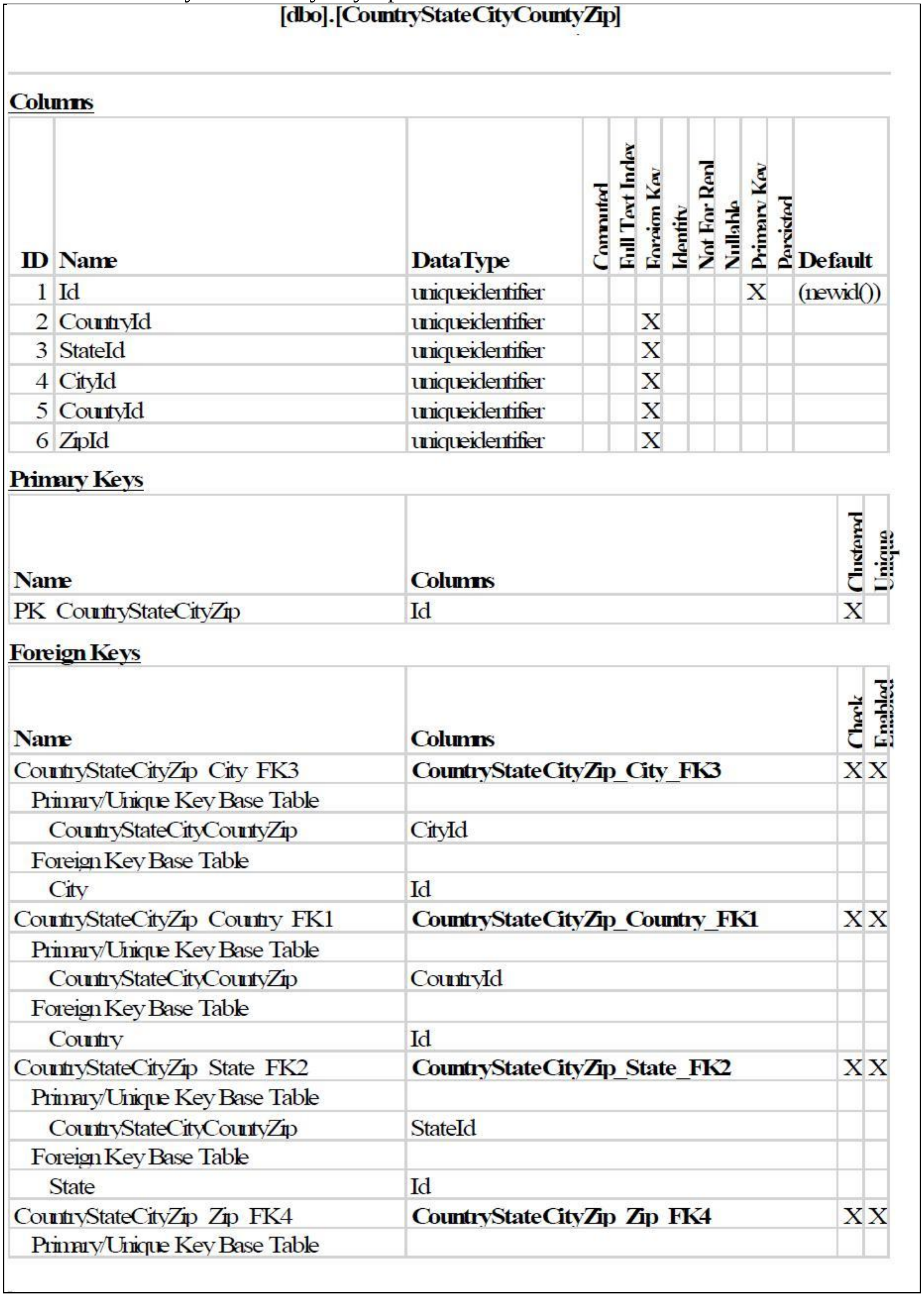
Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	CityName	varchar(150)									

Primary Keys

Name	Columns	Clustered	Unique
City PK	Id	X	

Table 21 - CountryStateCountyCityZip Table



[dbo].[CountryStateCityCountyZip]

Columns

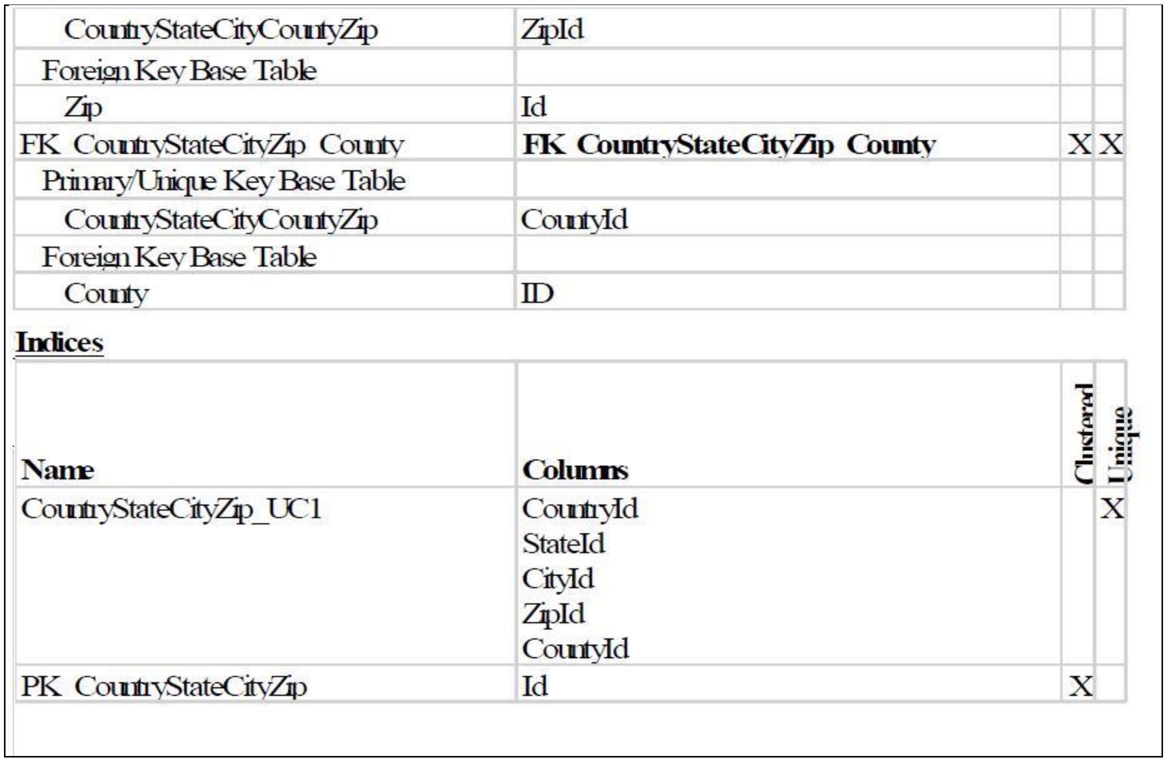
ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	CountryId	uniqueidentifier			X						
3	StateId	uniqueidentifier			X						
4	CityId	uniqueidentifier			X						
5	CountyId	uniqueidentifier			X						
6	ZipId	uniqueidentifier			X						

Primary Keys

Name	Columns	Clustered	Unique
PK CountryStateCityZip	Id	X	

Foreign Keys

Name	Columns	Check	Enabled
CountryStateCityZip City FK3	**CountryStateCityZip_City_FK3**	X	X
Primary/Unique Key Base Table			
CountryStateCityCountyZip	CityId		
Foreign Key Base Table			
City	Id		
CountryStateCityZip Country FK1	**CountryStateCityZip_Country_FK1**	X	X
Primary/Unique Key Base Table			
CountryStateCityCountyZip	CountryId		
Foreign Key Base Table			
Country	Id		
CountryStateCityZip State FK2	**CountryStateCityZip_State_FK2**	X	X
Primary/Unique Key Base Table			
CountryStateCityCountyZip	StateId		
Foreign Key Base Table			
State	Id		
CountryStateCityZip Zip FK4	**CountryStateCityZip Zip FK4**	X	X
Primary/Unique Key Base Table			

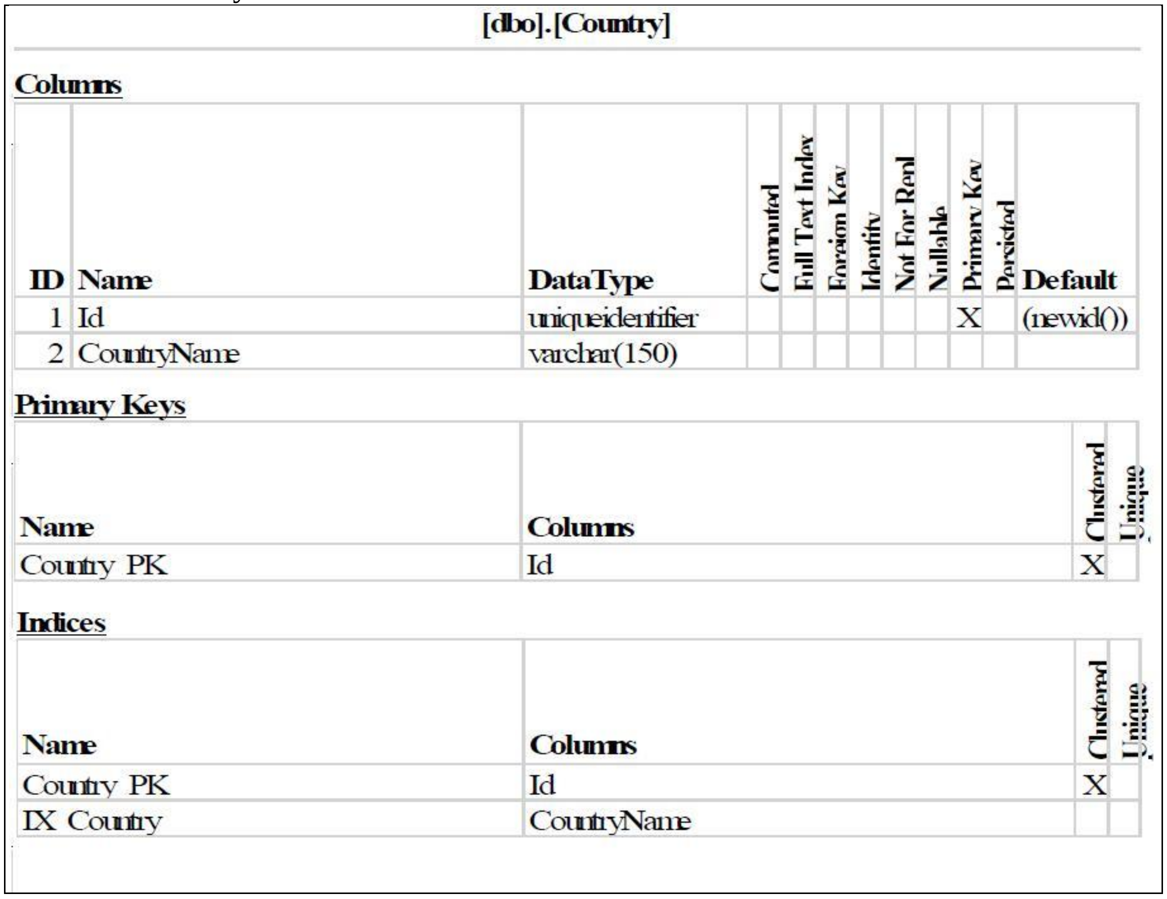


CountryStateCityCountyZip	ZipId		
Foreign Key Base Table			
Zip	Id		
FK CountryStateCityZip County	**FK CountryStateCityZip County**	X	X
Primary/Unique Key Base Table			
CountryStateCityCountyZip	CountyId		
Foreign Key Base Table			
County	ID		

Indices

Name	Columns	Clustered	Unique
CountryStateCityZip_UC1	CountryId StateId CityId ZipId CountyId		X
PK CountryStateCityZip	Id	X	

Table 22 - Country Table



[dbo].[Country]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	CountryName	varchar(150)									

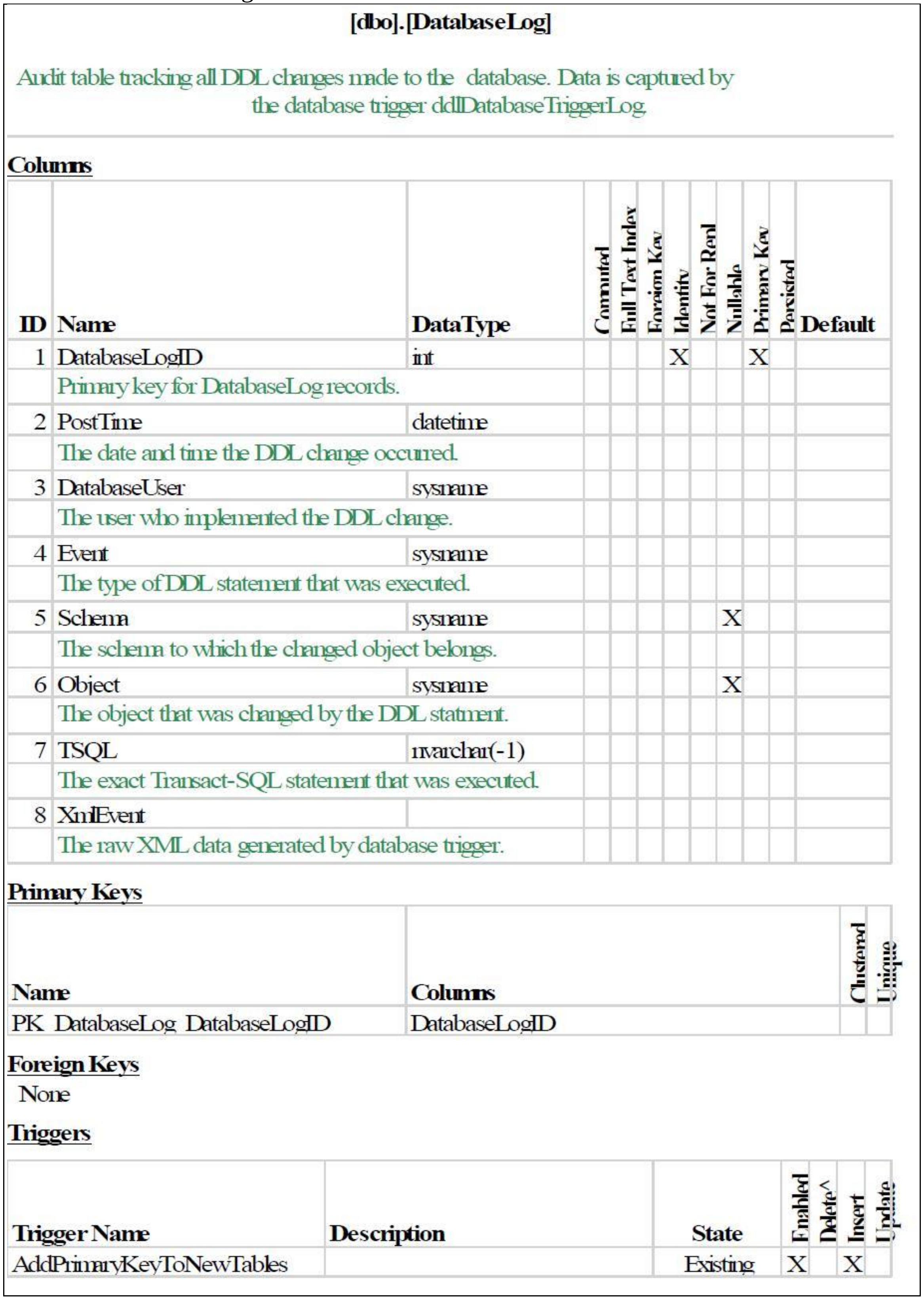
Primary Keys

Name	Columns	Clustered	Unique
Country PK	Id	X	

Indices

Name	Columns	Clustered	Unique
Country PK	Id	X	
IX Country	CountryName		

Table 23 - Database Log Table



[dbo].[DatabaseLog]

Audit table tracking all DDL changes made to the database. Data is captured by the database trigger ddlDatabaseTriggerLog.

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	DatabaseLogID	int				X			X		
	Primary key for DatabaseLog records.										
2	PostTime	datetime									
	The date and time the DDL change occurred.										
3	DatabaseUser	sysname									
	The user who implemented the DDL change.										
4	Event	sysname									
	The type of DDL statement that was executed.										
5	Schema	sysname						X			
	The schema to which the changed object belongs.										
6	Object	sysname						X			
	The object that was changed by the DDL statment.										
7	TSQL	nvarchar(-1)									
	The exact Transact-SQL statement that was executed.										
8	XmlEvent										
	The raw XML data generated by database trigger.										

Primary Keys

Name	Columns	Clustered	Unique
PK_DatabaseLog_DatabaseLogID	DatabaseLogID		

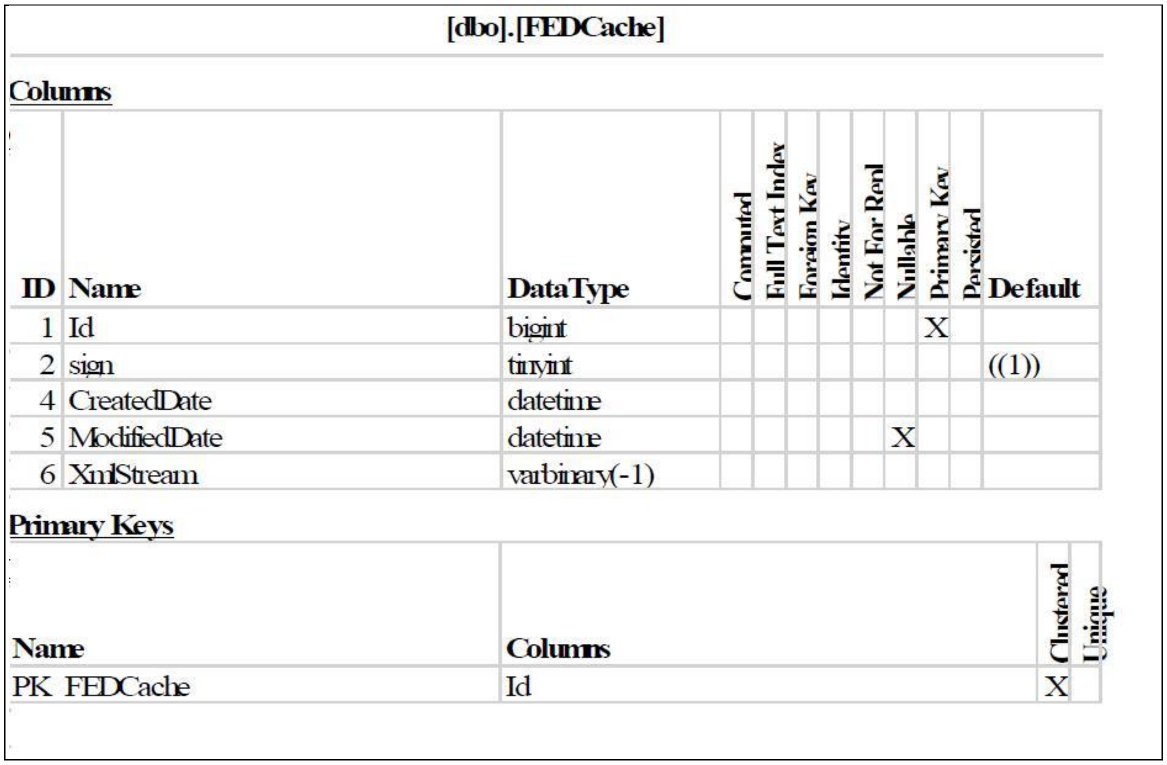
Foreign Keys

None

Triggers

Trigger Name	Description	State	Enabled	Delete^	Insert	Update
AddPrimaryKeyToNewTables		Existing	X		X	

Table 24 – FedCache Table



[dbo].[FEDCache]

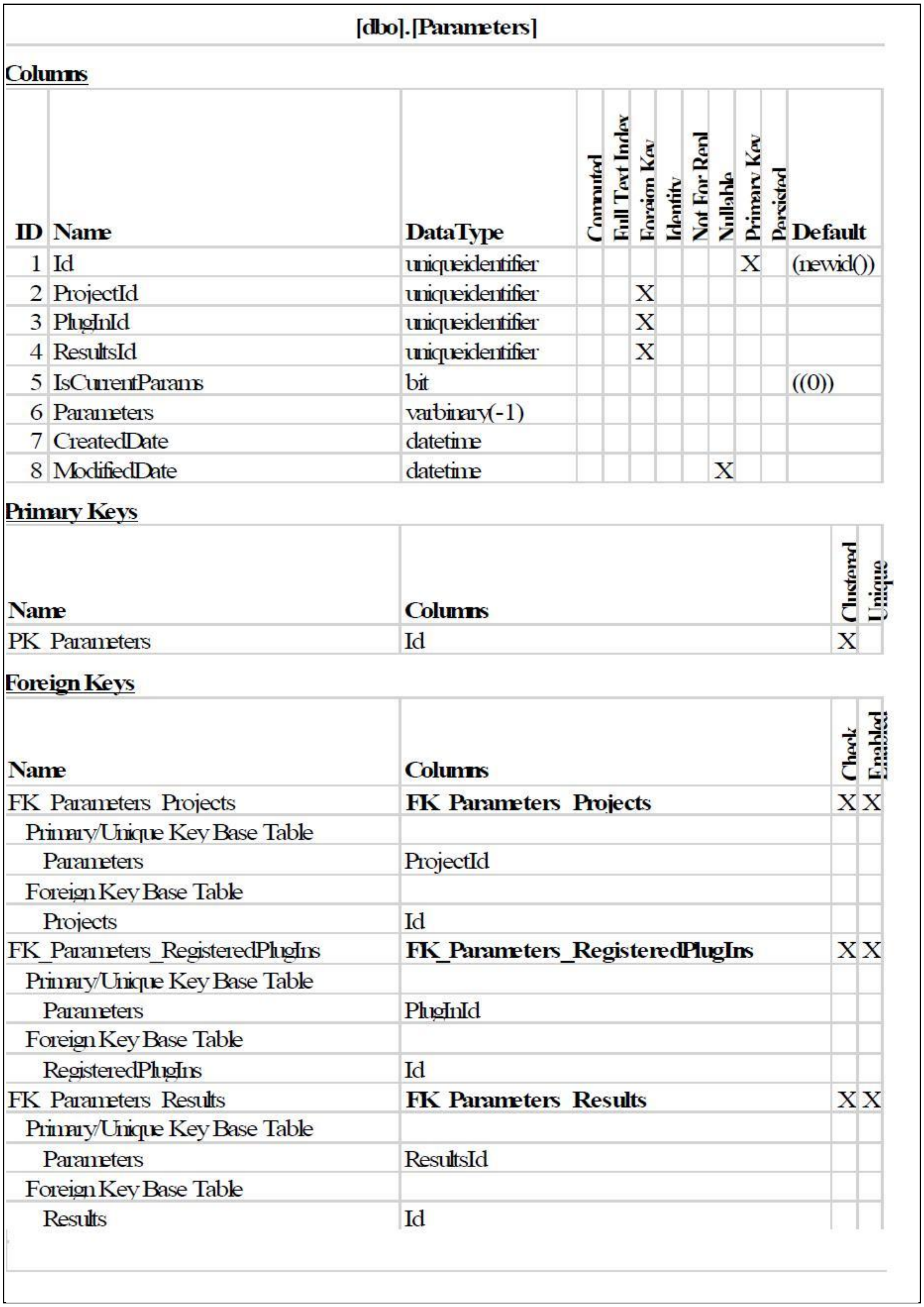
Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	bigint							X		
2	sign	tinyint									((1))
4	CreatedDate	datetime									
5	ModifiedDate	datetime						X			
6	XmlStream	varbinary(-1)									

Primary Keys

Name	Columns	Clustered	Unique
PK FEDCache	Id	X	

Table 25 - Parameters Table



[dbo].[Parameters]

Columns

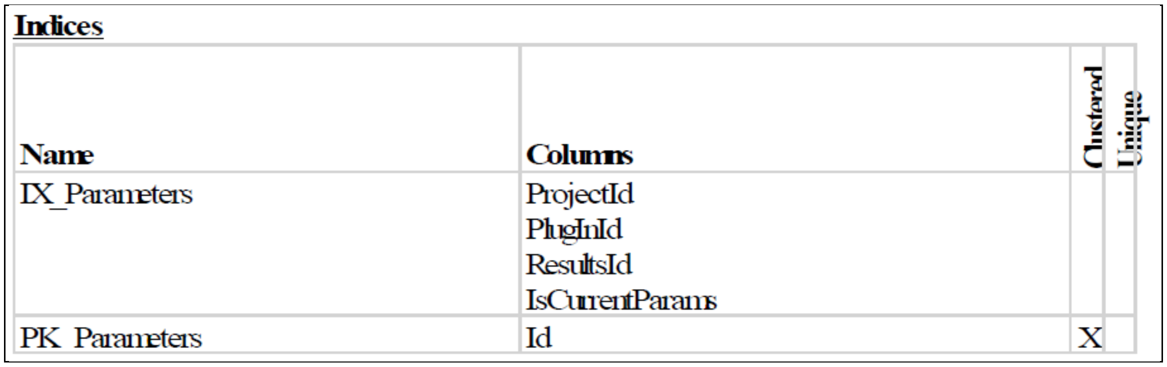
ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier			X						
3	PlugInId	uniqueidentifier			X						
4	ResultsId	uniqueidentifier			X						
5	IsCurrentParams	bit									((0))
6	Parameters	varbinary(-1)									
7	CreatedDate	datetime									
8	ModifiedDate	datetime						X			

Primary Keys

Name	Columns	Clustered	Unique
PK Parameters	Id	X	

Foreign Keys

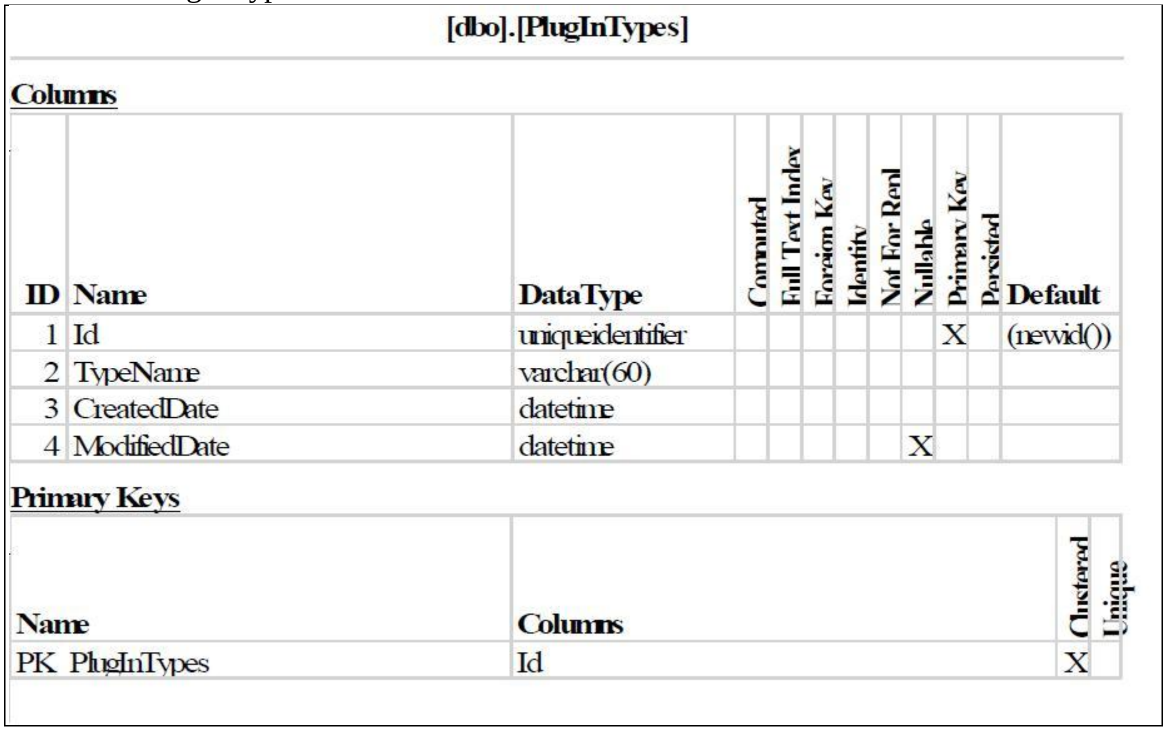
Name	Columns	Check	Enabled
FK Parameters Projects	**FK Parameters Projects**	X	X
Primary/Unique Key Base Table			
Parameters	ProjectId		
Foreign Key Base Table			
Projects	Id		
FK_Parameters_RegisteredPlugIns	**FK_Parameters_RegisteredPlugIns**	X	X
Primary/Unique Key Base Table			
Parameters	PlugInId		
Foreign Key Base Table			
RegisteredPlugIns	Id		
FK Parameters Results	**FK Parameters Results**	X	X
Primary/Unique Key Base Table			
Parameters	ResultsId		
Foreign Key Base Table			
Results	Id		



Indices

Name	Columns	Clustered	Unique
IX_Parameters	ProjectId PlugInId ResultsId IsCurrentParams		
PK_Parameters	Id	X	

Table 26 - PlugInTypes Table



[dbo].[PlugInTypes]

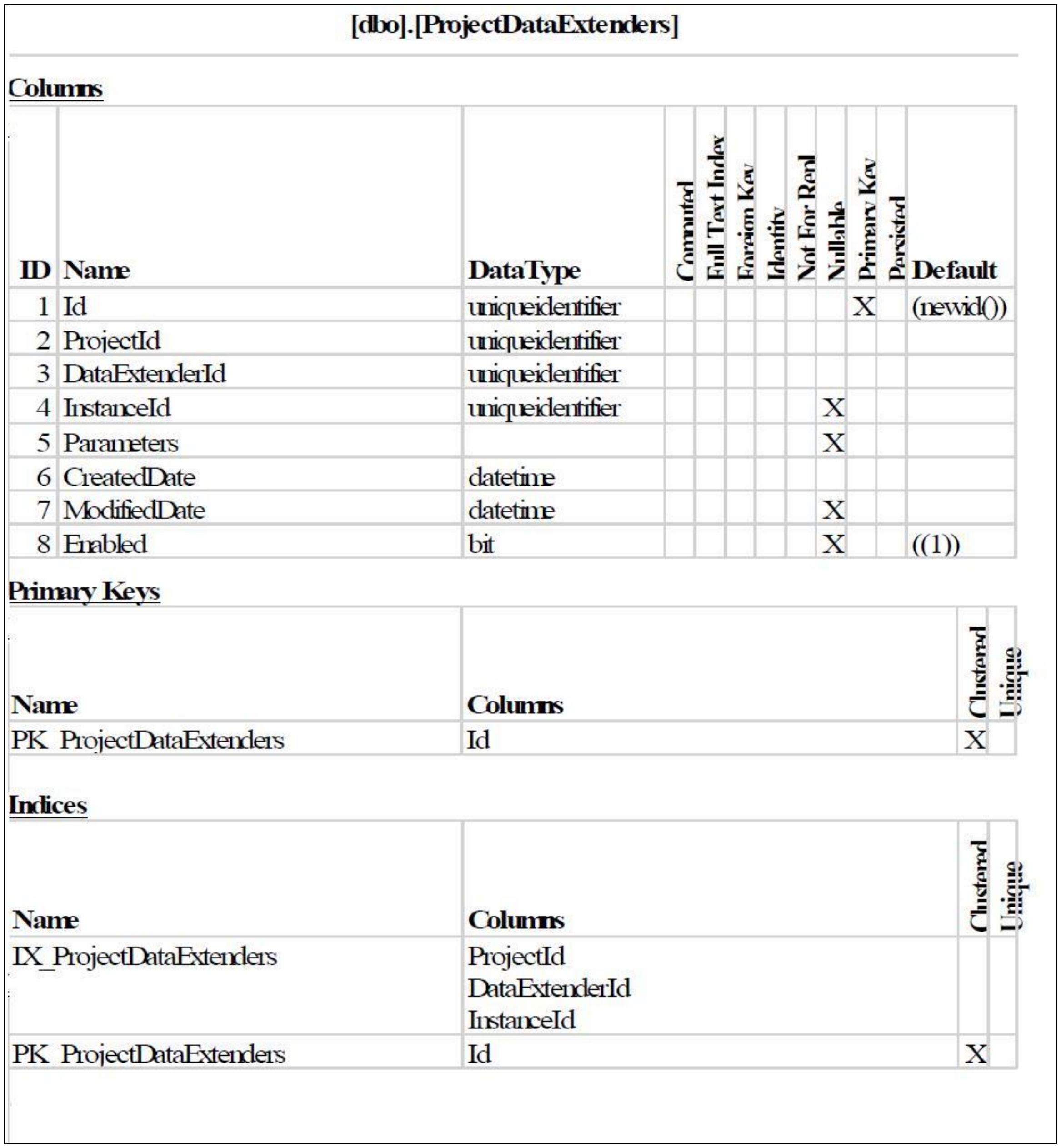
Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	TypeName	varchar(60)									
3	CreatedDate	datetime									
4	ModifiedDate	datetime						X			

Primary Keys

Name	Columns	Clustered	Unique
PK_PlugInTypes	Id	X	

Table 27 - ProjectDataExtenders Table



[dbo].[ProjectDataExtenders]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier									
3	DataExtenderId	uniqueidentifier									
4	InstanceId	uniqueidentifier						X			
5	Parameters							X			
6	CreatedDate	datetime									
7	ModifiedDate	datetime						X			
8	Enabled	bit						X			((1))

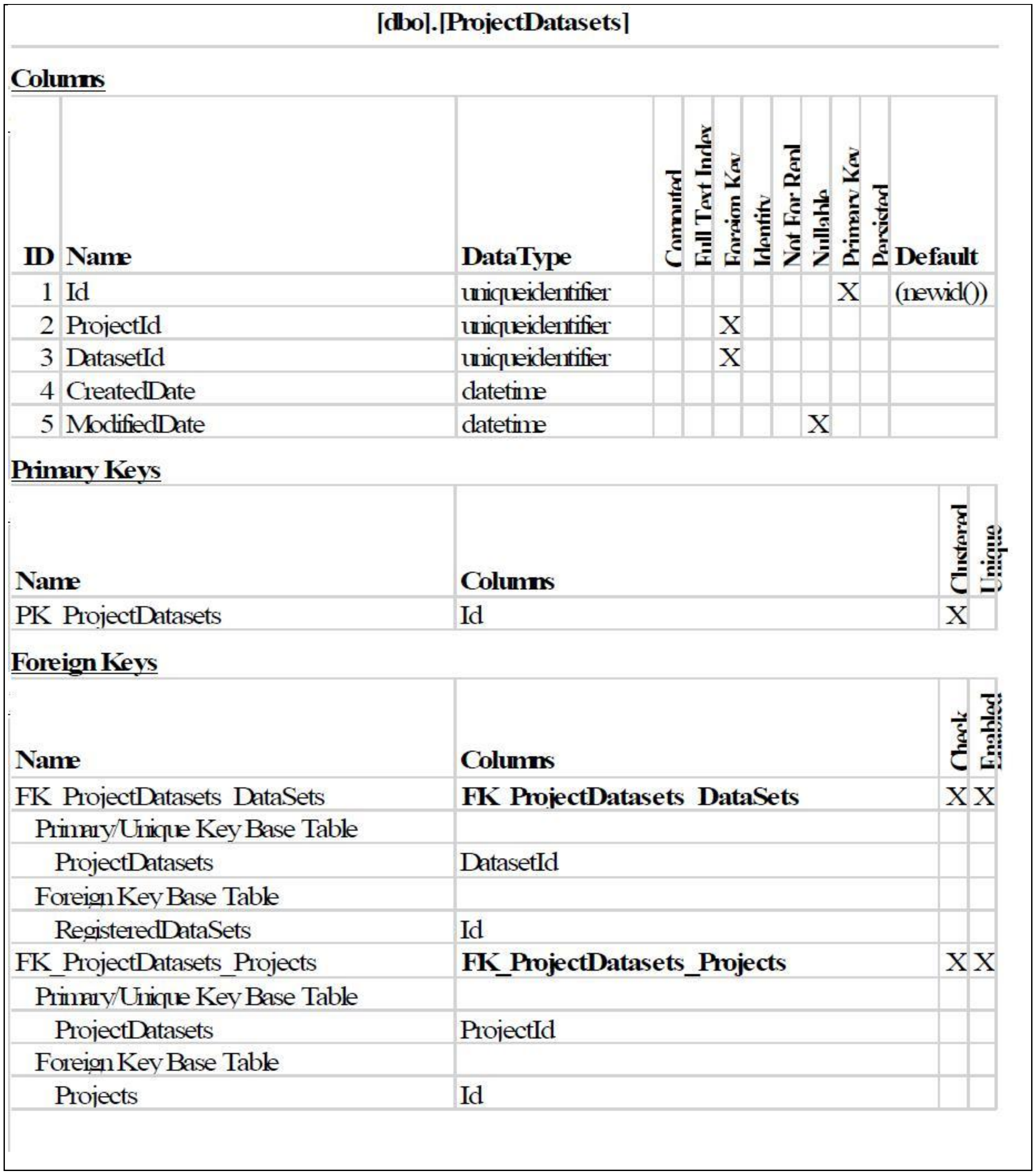
Primary Keys

Name	Columns	Clustered	Unique
PK_ProjectDataExtenders	Id	X	

Indices

Name	Columns	Clustered	Unique
IX_ProjectDataExtenders	ProjectId DataExtenderId InstanceId		
PK_ProjectDataExtenders	Id	X	

Table 28 - ProjectDataSets Table



[dbo].[ProjectDatasets]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier			X						
3	DatasetId	uniqueidentifier			X						
4	CreatedDate	datetime									
5	ModifiedDate	datetime						X			

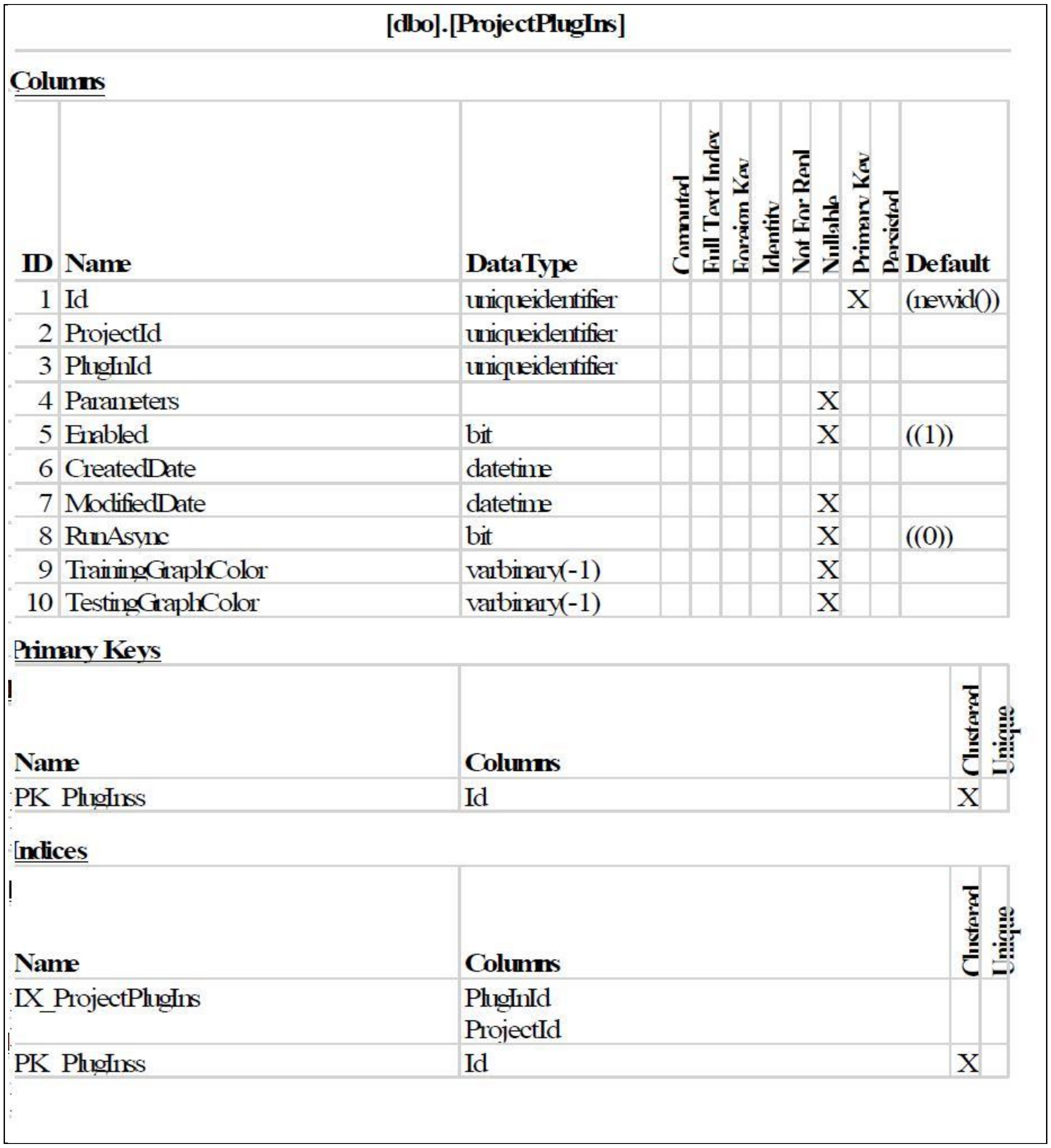
Primary Keys

Name	Columns	Clustered	Unique
PK ProjectDatasets	Id	X	

Foreign Keys

Name	Columns	Check	Enabled
FK ProjectDatasets DataSets	**FK ProjectDatasets DataSets**	X	X
Primary/Unique Key Base Table			
ProjectDatasets	DatasetId		
Foreign Key Base Table			
RegisteredDataSets	Id		
FK_ProjectDatasets_Projects	**FK_ProjectDatasets_Projects**	X	X
Primary/Unique Key Base Table			
ProjectDatasets	ProjectId		
Foreign Key Base Table			
Projects	Id		

Table 29 - ProjectPlugIns Table



[dbo].[ProjectPlugIns]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier									
3	PlugInId	uniqueidentifier									
4	Parameters							X			
5	Enabled	bit						X			((1))
6	CreatedDate	datetime									
7	ModifiedDate	datetime						X			
8	RunAsync	bit						X			((0))
9	TrainingGraphColor	varbinary(-1)						X			
10	TestingGraphColor	varbinary(-1)						X			

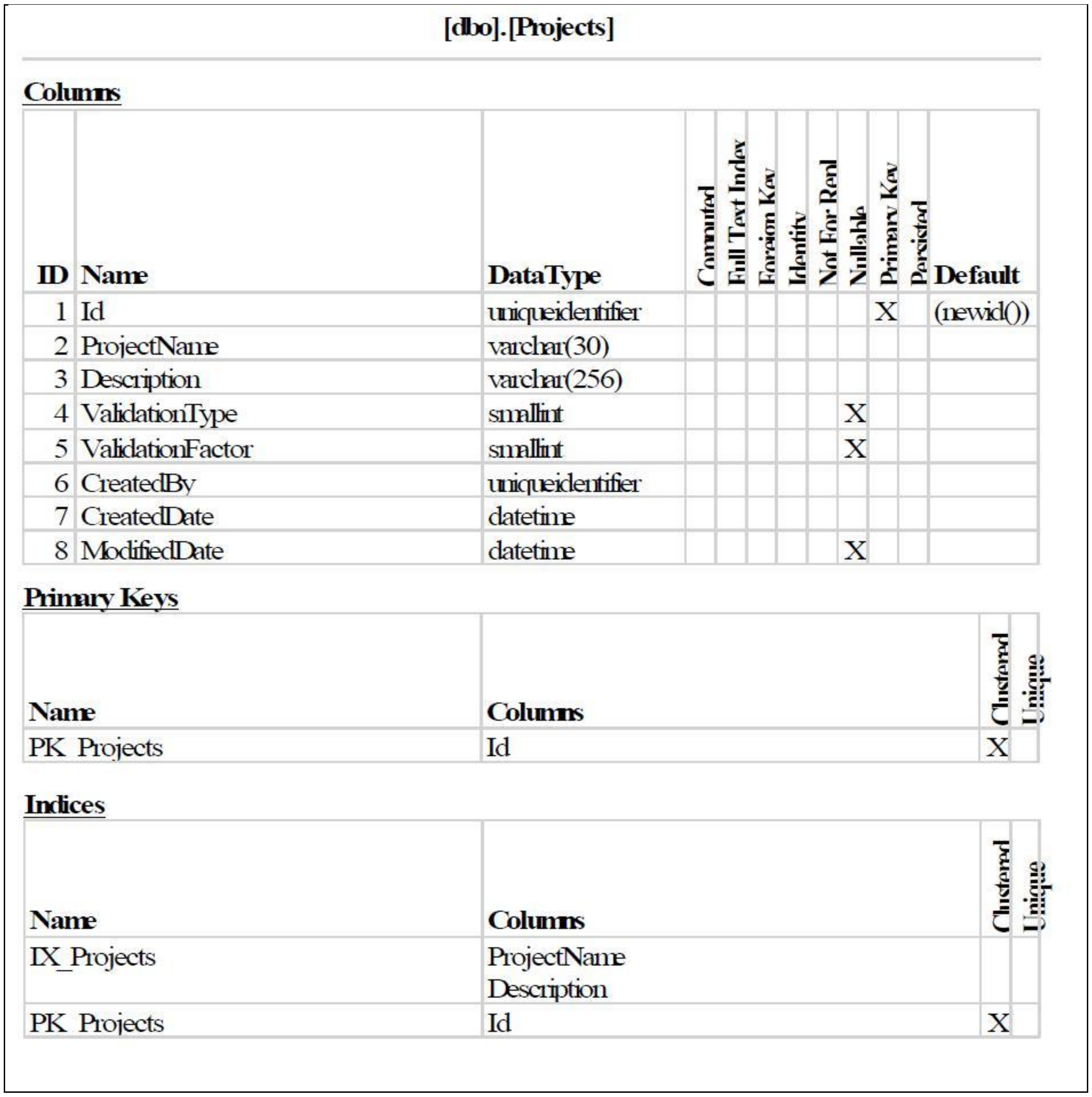
Primary Keys

Name	Columns	Clustered	Unique
PK_PlugInss	Id	X	

Indices

Name	Columns	Clustered	Unique
IX_ProjectPlugIns	PlugInId ProjectId		
PK_PlugInss	Id	X	

Table 30 - Projects Table



[dbo].[Projects]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectName	varchar(30)									
3	Description	varchar(256)									
4	ValidationType	smallint						X			
5	ValidationFactor	smallint						X			
6	CreatedBy	uniqueidentifier									
7	CreatedDate	datetime									
8	ModifiedDate	datetime						X			

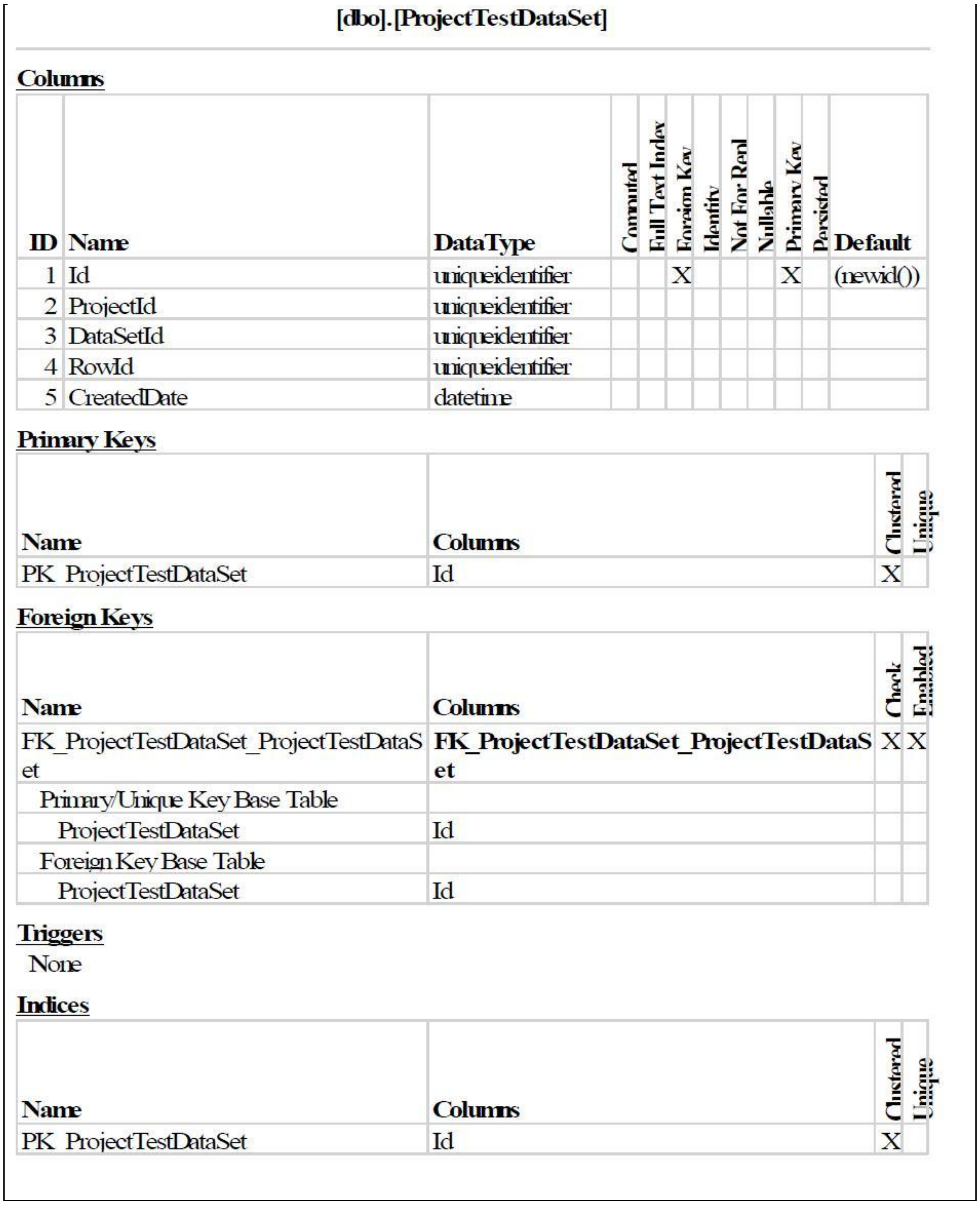
Primary Keys

Name	Columns	Clustered	Unique
PK_Projects	Id	X	

Indices

Name	Columns	Clustered	Unique
IX_Projects	ProjectName Description		
PK_Projects	Id	X	

Table 31 - ProjectTestDataSet Table



[dbo].[ProjectTestDataSet]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier			X				X		(newid())
2	ProjectId	uniqueidentifier									
3	DataSetId	uniqueidentifier									
4	RowId	uniqueidentifier									
5	CreatedDate	datetime									

Primary Keys

Name	Columns	Clustered	Unique
PK ProjectTestDataSet	Id	X	

Foreign Keys

Name	Columns	Check	Enabled
FK_ProjectTestDataSet_ProjectTestDataSet	**FK_ProjectTestDataSet_ProjectTestDataSet**	X	X
Primary/Unique Key Base Table			
ProjectTestDataSet	Id		
Foreign Key Base Table			
ProjectTestDataSet	Id		

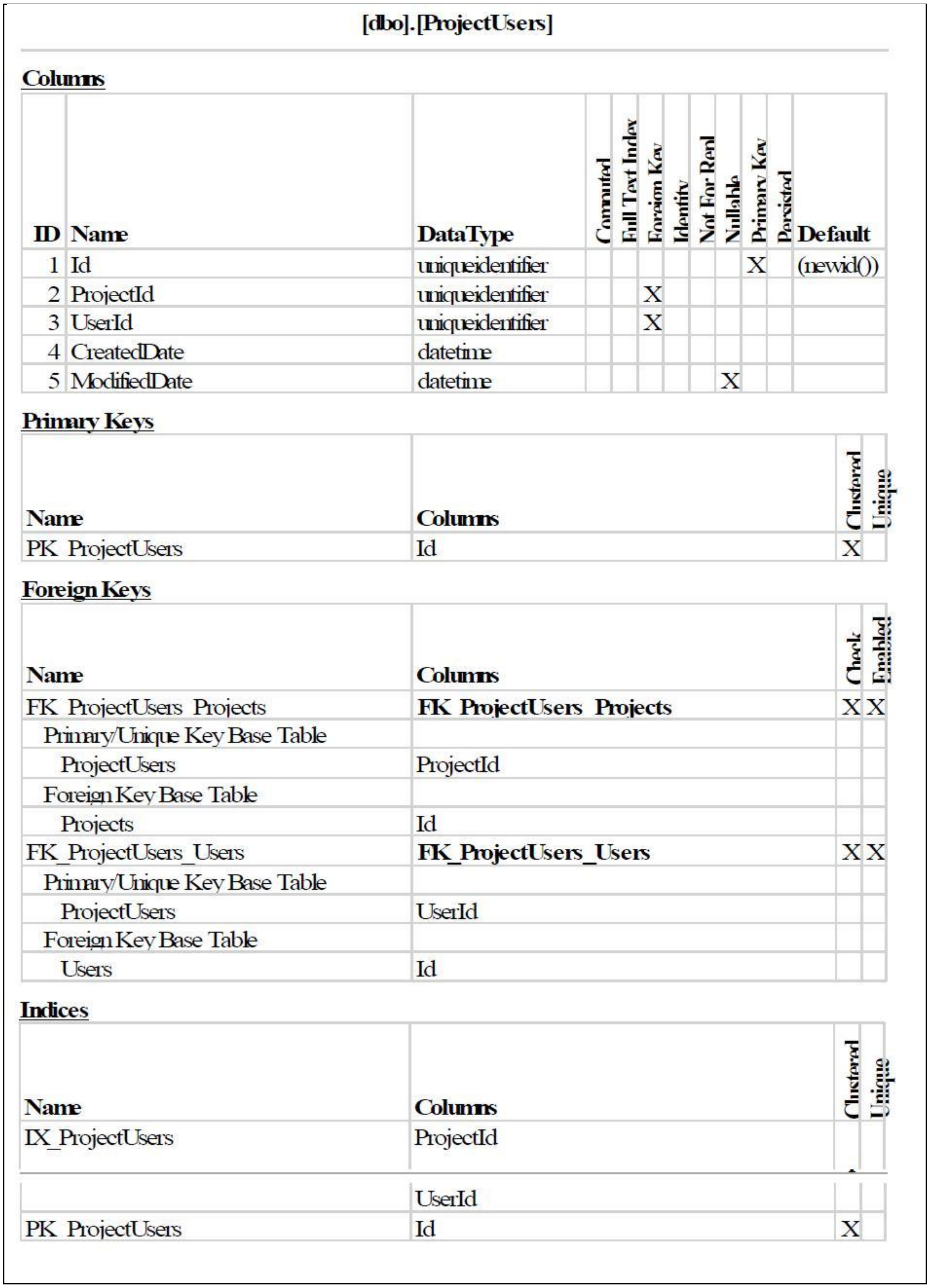
Triggers

None

Indices

Name	Columns	Clustered	Unique
PK ProjectTestDataSet	Id	X	

Table 32 - ProjectUsers Table



[dbo].[ProjectUsers]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier			X						
3	UserId	uniqueidentifier			X						
4	CreatedDate	datetime									
5	ModifiedDate	datetime						X			

Primary Keys

Name	Columns	Clustered	Unique
PK ProjectUsers	Id	X	

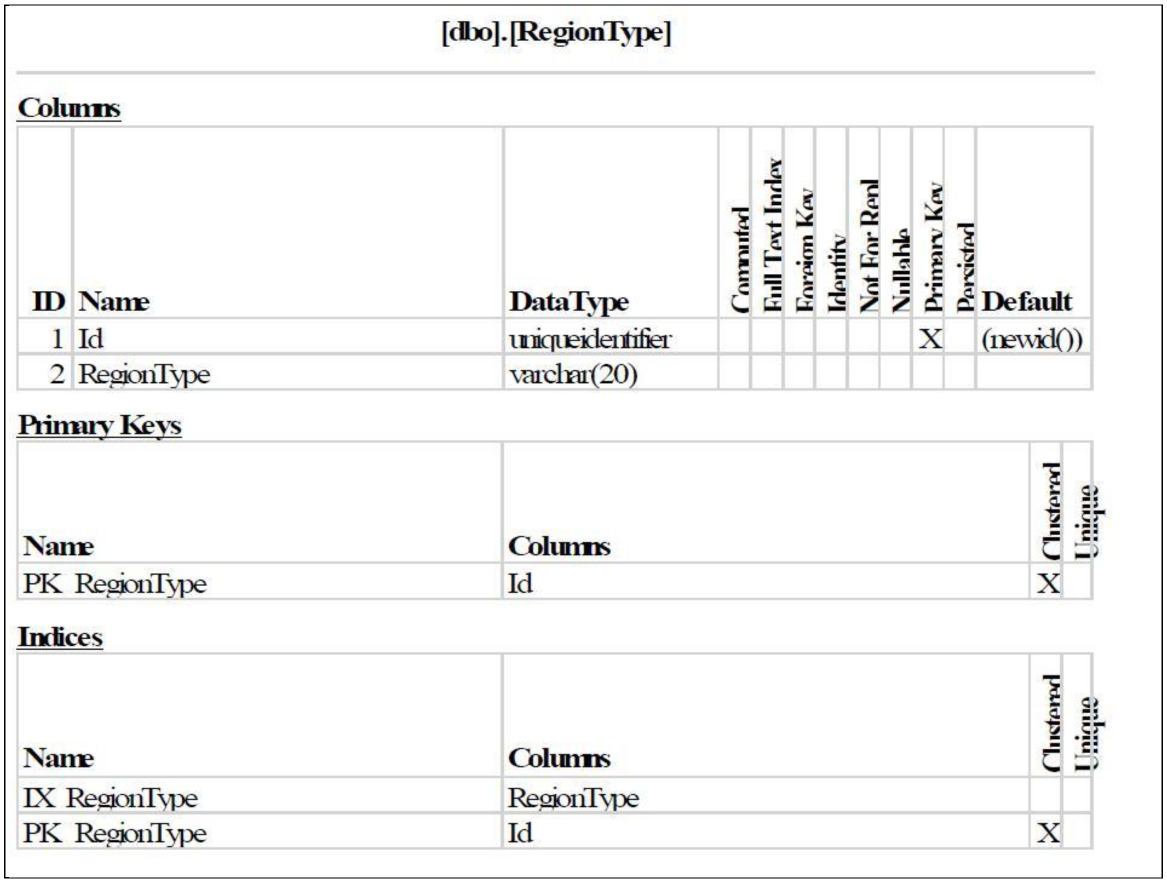
Foreign Keys

Name	Columns	Check	Enabled
FK ProjectUsers Projects	**FK ProjectUsers Projects**	X	X
Primary/Unique Key Base Table			
ProjectUsers	ProjectId		
Foreign Key Base Table			
Projects	Id		
FK_ProjectUsers_Users	**FK_ProjectUsers_Users**	X	X
Primary/Unique Key Base Table			
ProjectUsers	UserId		
Foreign Key Base Table			
Users	Id		

Indices

Name	Columns	Clustered	Unique
IX_ProjectUsers	ProjectId		
	UserId		
PK ProjectUsers	Id	X	

Table 33 - RegionType Table



[dbo].[RegionType]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	RegionType	varchar(20)									

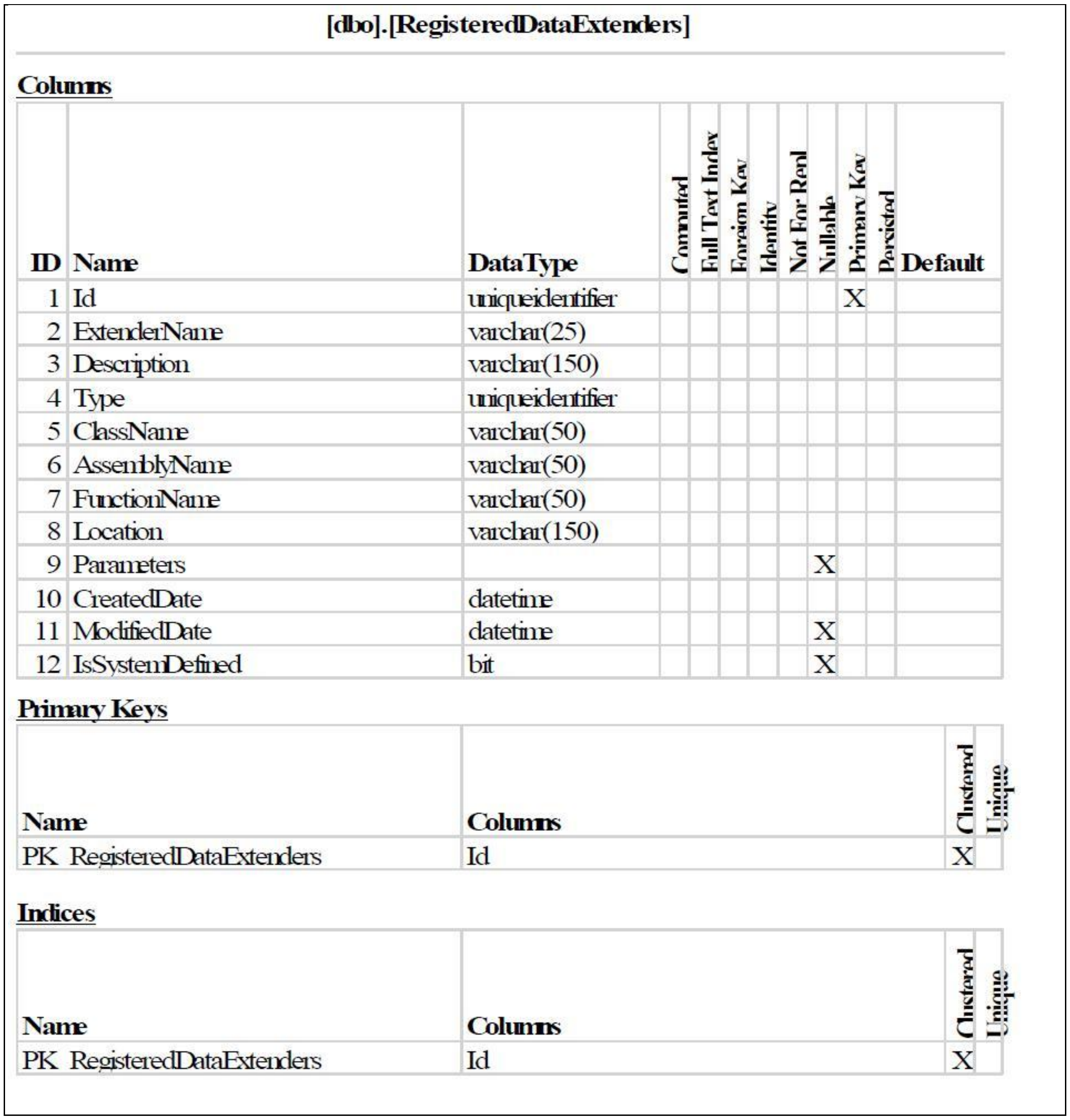
Primary Keys

Name	Columns	Clustered	Unique
PK RegionType	Id	X	

Indices

Name	Columns	Clustered	Unique
IX RegionType	RegionType		
PK RegionType	Id	X	

Table 34 - RegisteredDataExtenders Table



[dbo].[RegisteredDataExtenders]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		
2	ExtenderName	varchar(25)									
3	Description	varchar(150)									
4	Type	uniqueidentifier									
5	ClassName	varchar(50)									
6	AssemblyName	varchar(50)									
7	FunctionName	varchar(50)									
8	Location	varchar(150)									
9	Parameters							X			
10	CreatedDate	datetime									
11	ModifiedDate	datetime						X			
12	IsSystemDefined	bit						X			

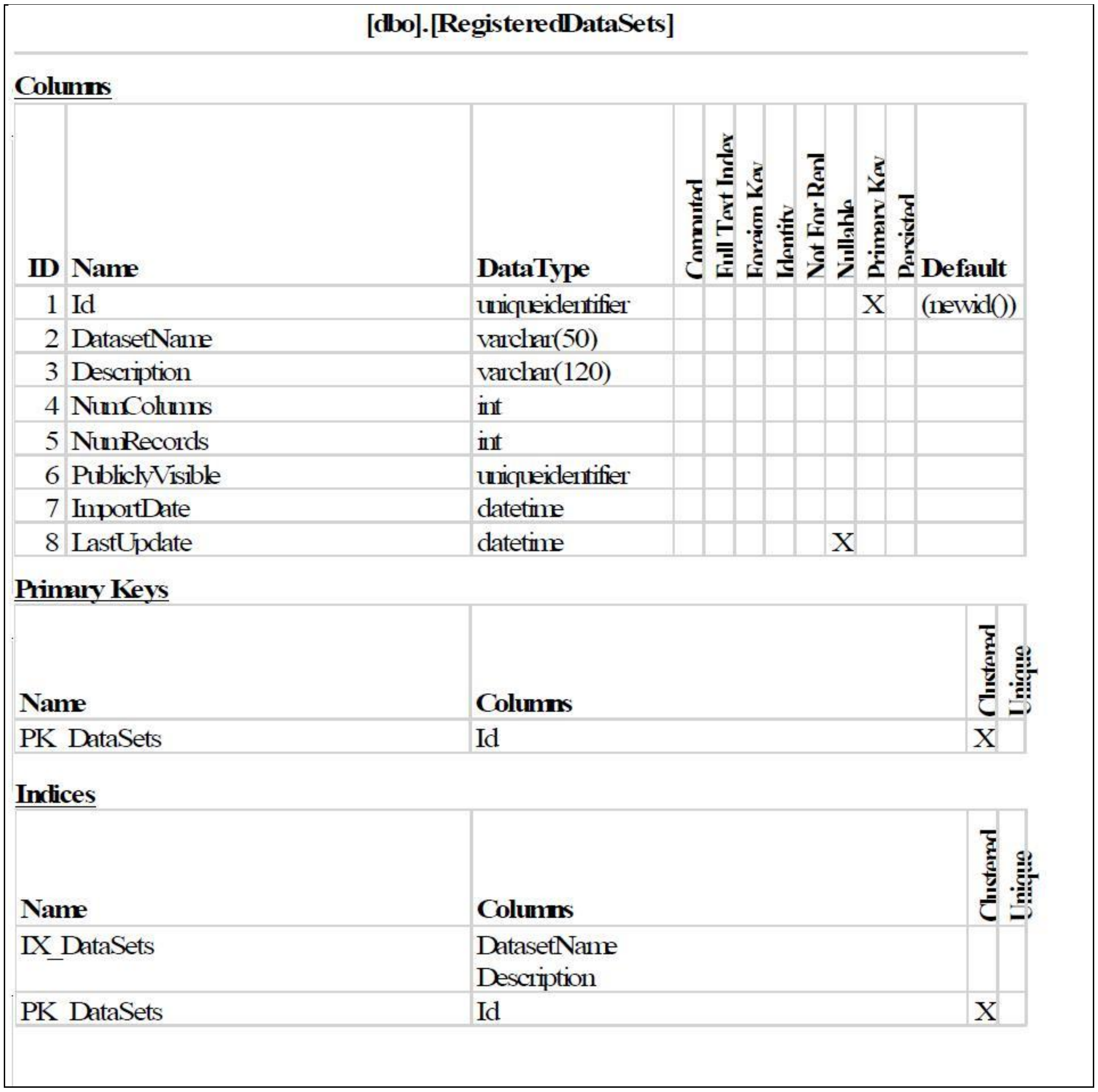
Primary Keys

Name	Columns	Clustered	Unique
PK_RegisteredDataExtenders	Id	X	

Indices

Name	Columns	Clustered	Unique
PK_RegisteredDataExtenders	Id	X	

Table 35 - RegisteredDataSets Table



[dbo].[RegisteredDataSets]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	DatasetName	varchar(50)									
3	Description	varchar(120)									
4	NumColumns	int									
5	NumRecords	int									
6	PubliclyVisible	uniqueidentifier									
7	ImportDate	datetime									
8	LastUpdate	datetime						X			

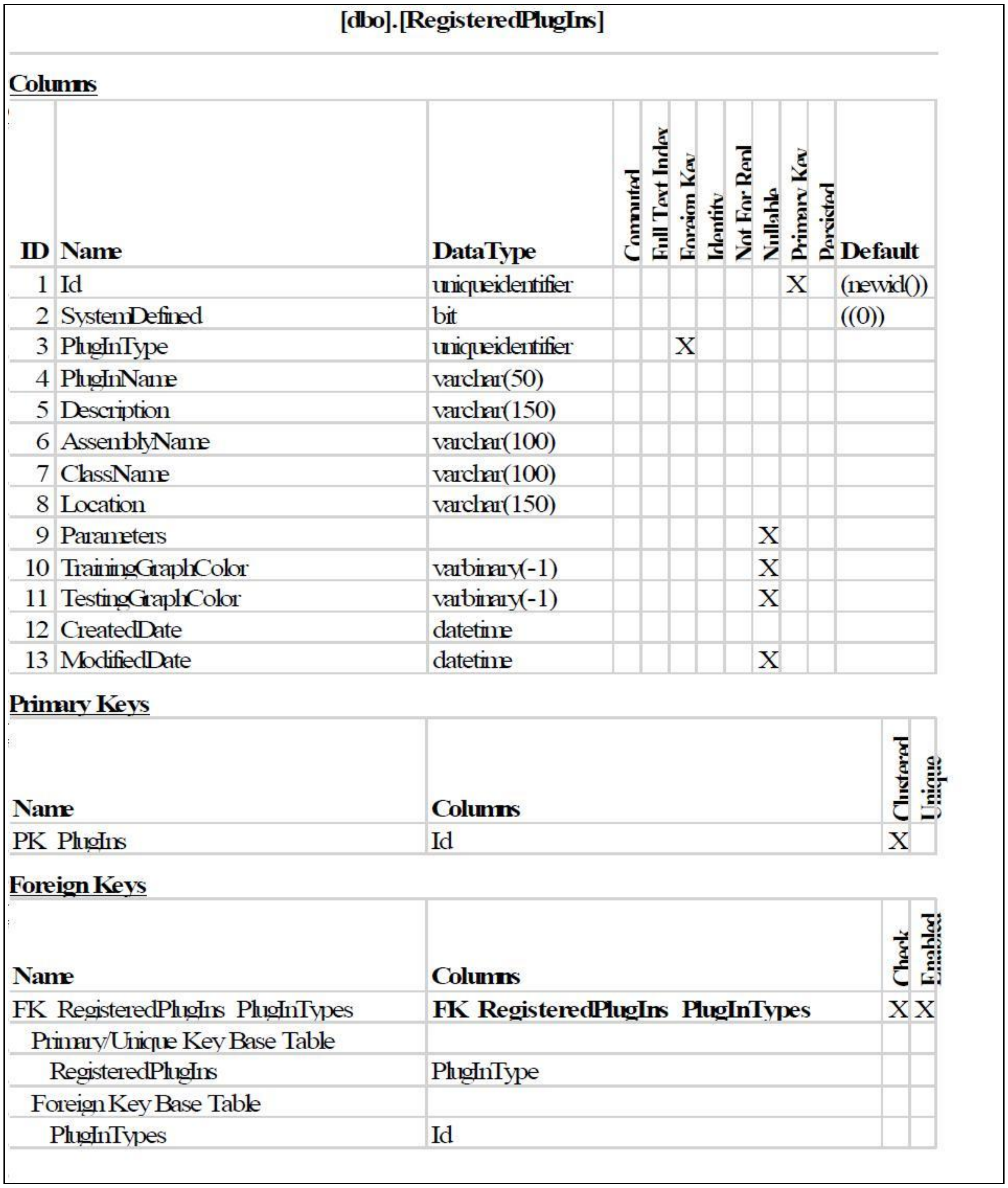
Primary Keys

Name	Columns	Clustered	Unique
PK_DataSets	Id	X	

Indices

Name	Columns	Clustered	Unique
IX_DataSets	DatasetName Description		
PK_DataSets	Id	X	

Table 36 - RegisteredPlugIns Table



[dbo].[RegisteredPlugIns]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	SystemDefined	bit									((0))
3	PlugInType	uniqueidentifier			X						
4	PlugInName	varchar(50)									
5	Description	varchar(150)									
6	AssemblyName	varchar(100)									
7	ClassName	varchar(100)									
8	Location	varchar(150)									
9	Parameters							X			
10	TrainingGraphColor	varbinary(-1)						X			
11	TestingGraphColor	varbinary(-1)						X			
12	CreatedDate	datetime									
13	ModifiedDate	datetime						X			

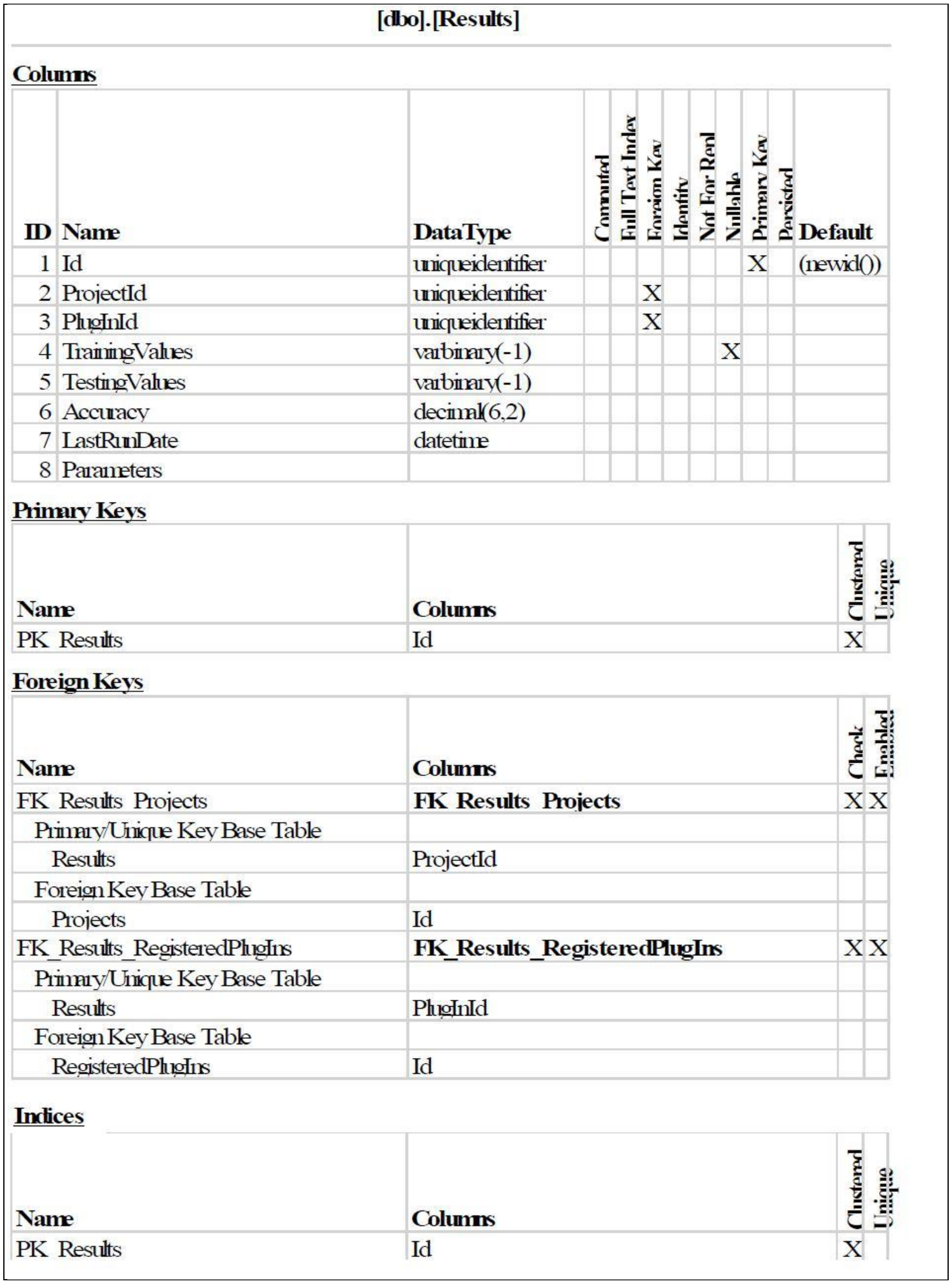
Primary Keys

Name	Columns	Clustered	Unique
PK PlugIns	Id	X	

Foreign Keys

Name	Columns	Check	Enabled
FK RegisteredPlugIns PlugInTypes	**FK RegisteredPlugIns PlugInTypes**	X	X
Primary/Unique Key Base Table			
RegisteredPlugIns	PlugInType		
Foreign Key Base Table			
PlugInTypes	Id		

Table 37 - Results Table



[dbo].[Results]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ProjectId	uniqueidentifier			X						
3	PlugInId	uniqueidentifier			X						
4	TrainingValues	varbinary(-1)						X			
5	TestingValues	varbinary(-1)									
6	Accuracy	decimal(6,2)									
7	LastRunDate	datetime									
8	Parameters										

Primary Keys

Name	Columns	Clustered	Unique
PK_Results	Id	X	

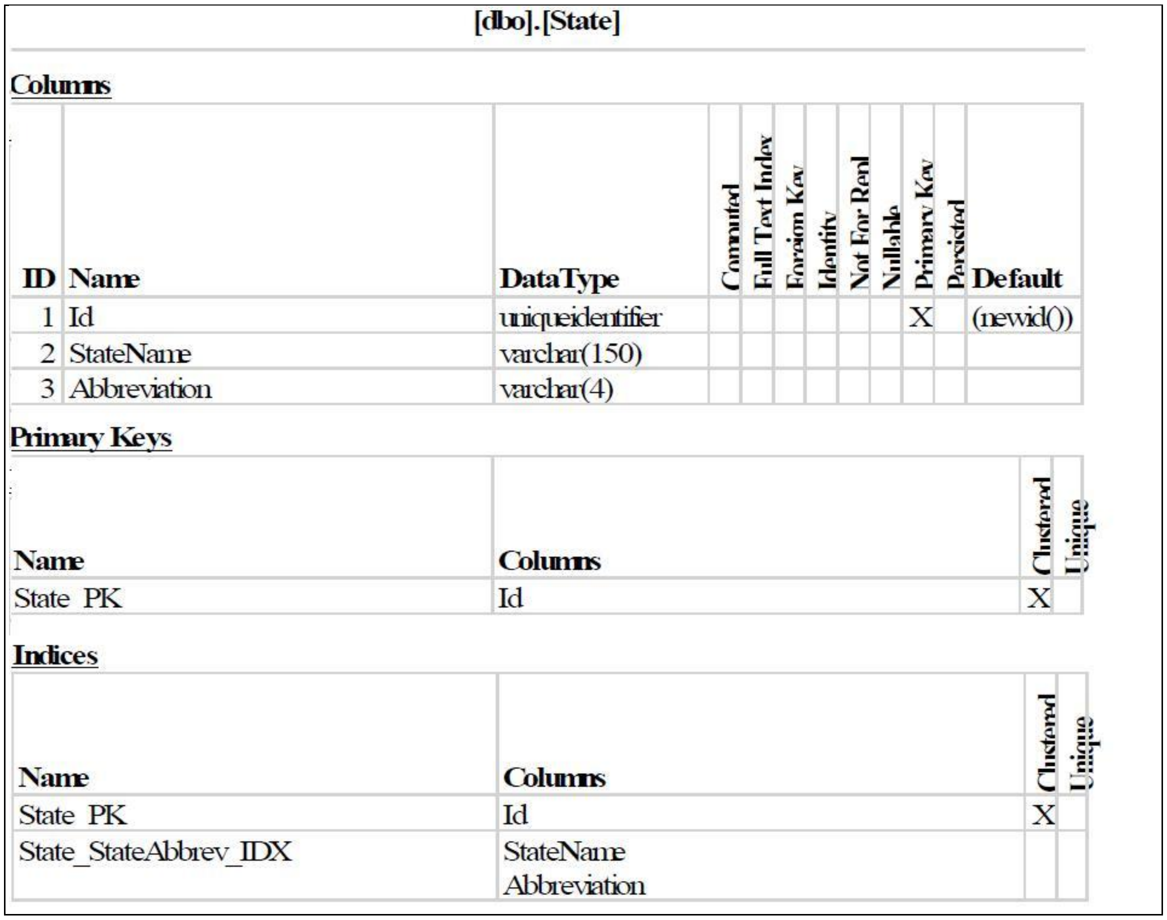
Foreign Keys

Name	Columns	Check	Enabled
FK_Results_Projects	**FK_Results_Projects**	X	X
Primary/Unique Key Base Table			
Results	ProjectId		
Foreign Key Base Table			
Projects	Id		
FK_Results_RegisteredPlugIns	**FK_Results_RegisteredPlugIns**	X	X
Primary/Unique Key Base Table			
Results	PlugInId		
Foreign Key Base Table			
RegisteredPlugIns	Id		

Indices

Name	Columns	Clustered	Unique
PK_Results	Id	X	

Table 38 - State Table



[dbo].[State]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	StateName	varchar(150)									
3	Abbreviation	varchar(4)									

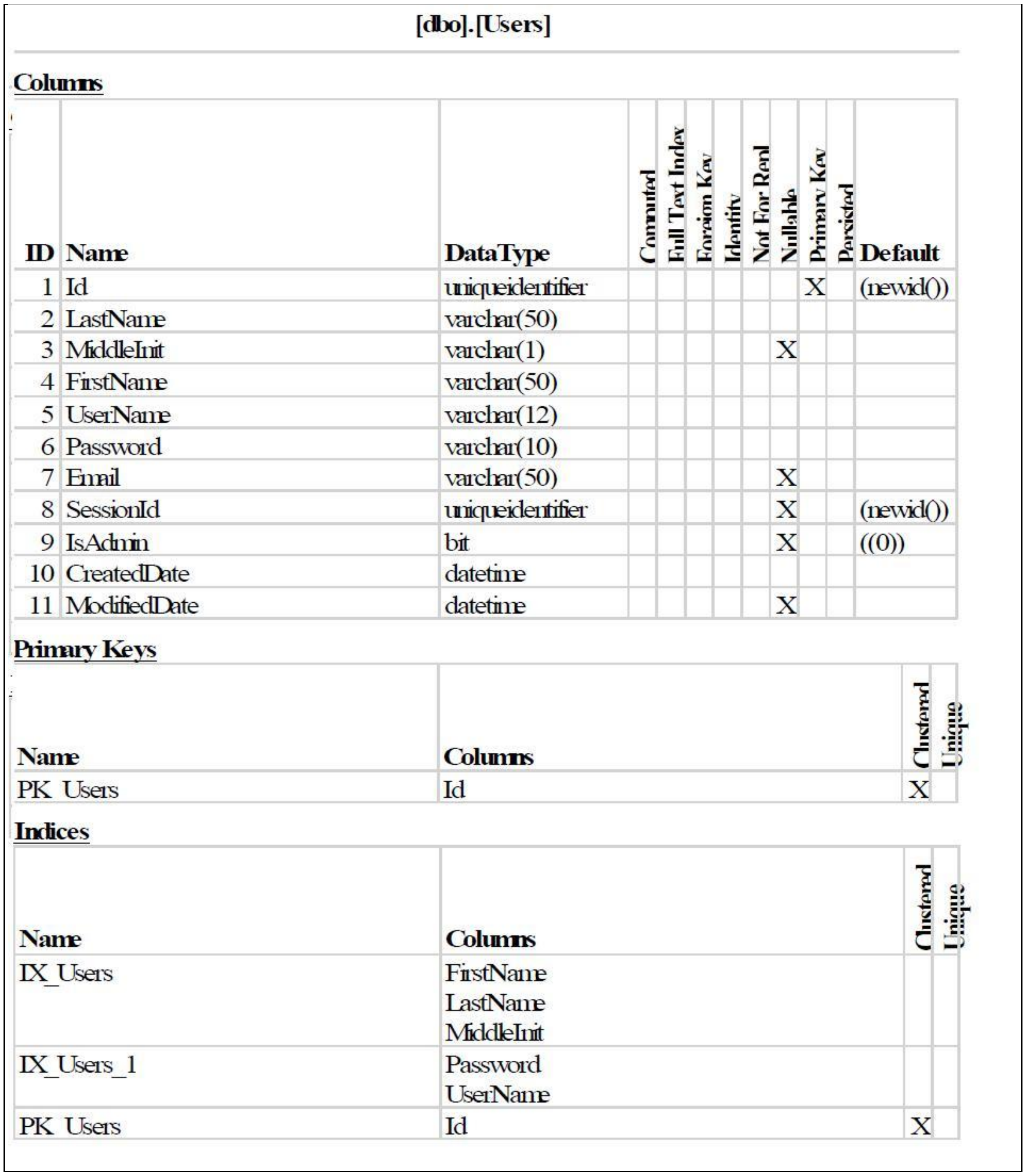
Primary Keys

Name	Columns	Clustered	Unique
State PK	Id	X	

Indices

Name	Columns	Clustered	Unique
State PK	Id	X	
State_StateAbbrev_IDX	StateName Abbreviation		

Table 39 - Users Table



[dbo].[Users]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	LastName	varchar(50)									
3	MiddleInit	varchar(1)						X			
4	FirstName	varchar(50)									
5	UserName	varchar(12)									
6	Password	varchar(10)									
7	Email	varchar(50)						X			
8	SessionId	uniqueidentifier						X			(newid())
9	IsAdmin	bit						X			((0))
10	CreatedDate	datetime									
11	ModifiedDate	datetime						X			

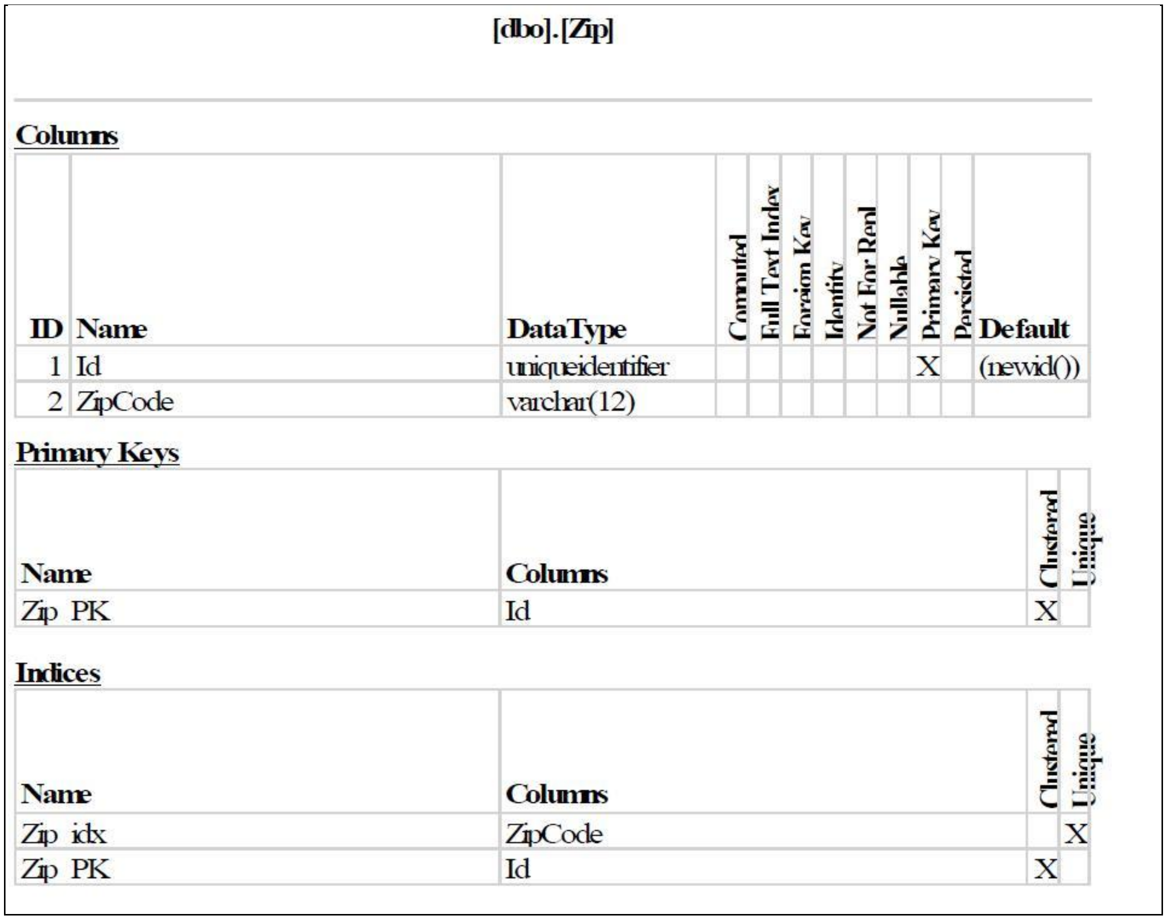
Primary Keys

Name	Columns	Clustered	Unique
PK Users	Id	X	

Indices

Name	Columns	Clustered	Unique
IX_Users	FirstName LastName MiddleInit		
IX_Users_1	Password UserName		
PK Users	Id	X	

Table 40 - Zip Table



[dbo].[Zip]

Columns

ID	Name	DataType	Computed	Full Text Index	Foreign Key	Identity	Not For Repl	Nullable	Primary Key	Persisted	Default
1	Id	uniqueidentifier							X		(newid())
2	ZipCode	varchar(12)									

Primary Keys

Name	Columns	Clustered	Unique
Zip PK	Id	X	

Indices

Name	Columns	Clustered	Unique
Zip idx	ZipCode		X
Zip PK	Id	X	

F. Raptor ERD

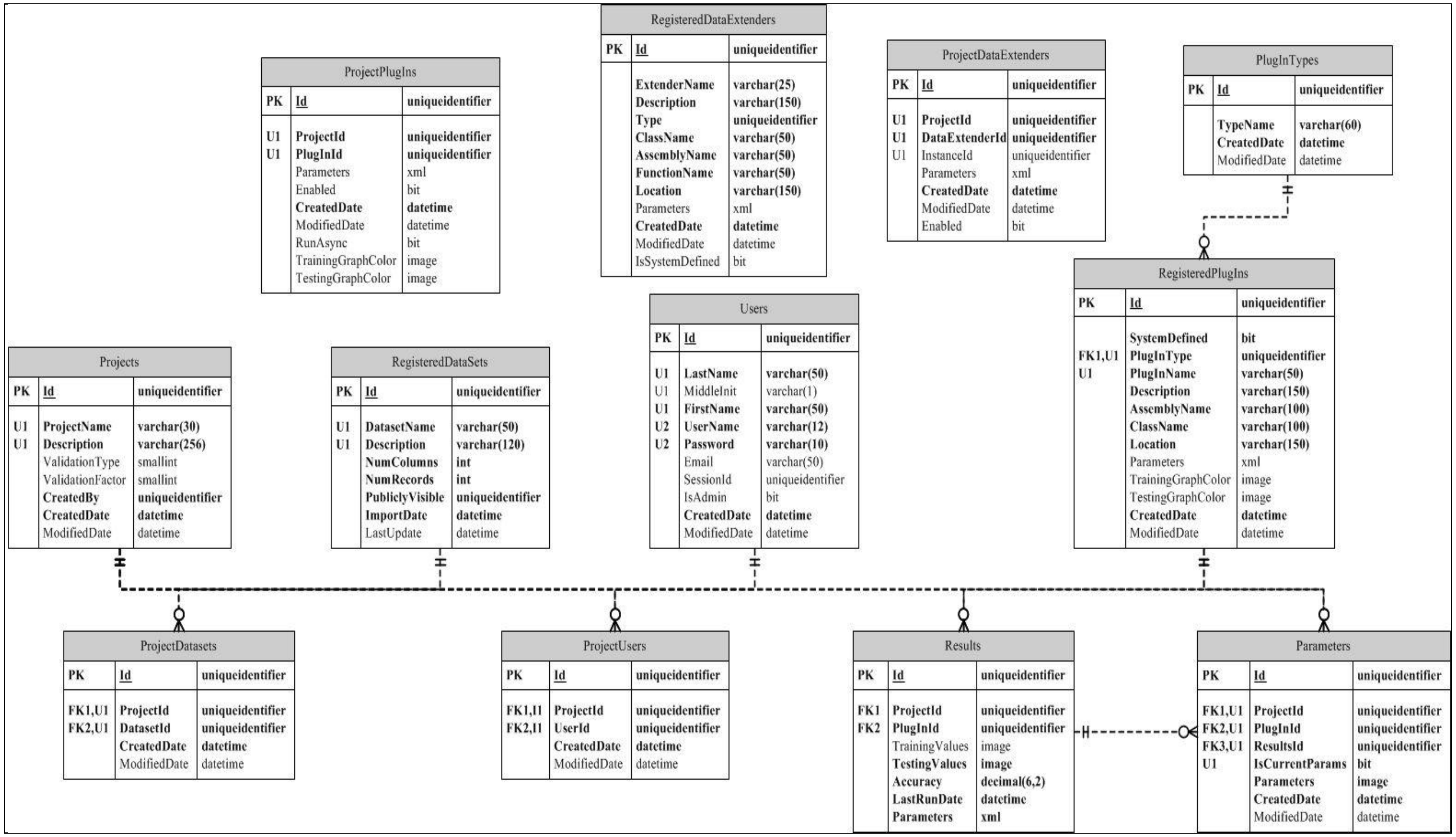


Figure 29 - Entity Relationship Diagram of Raptor Database (a)

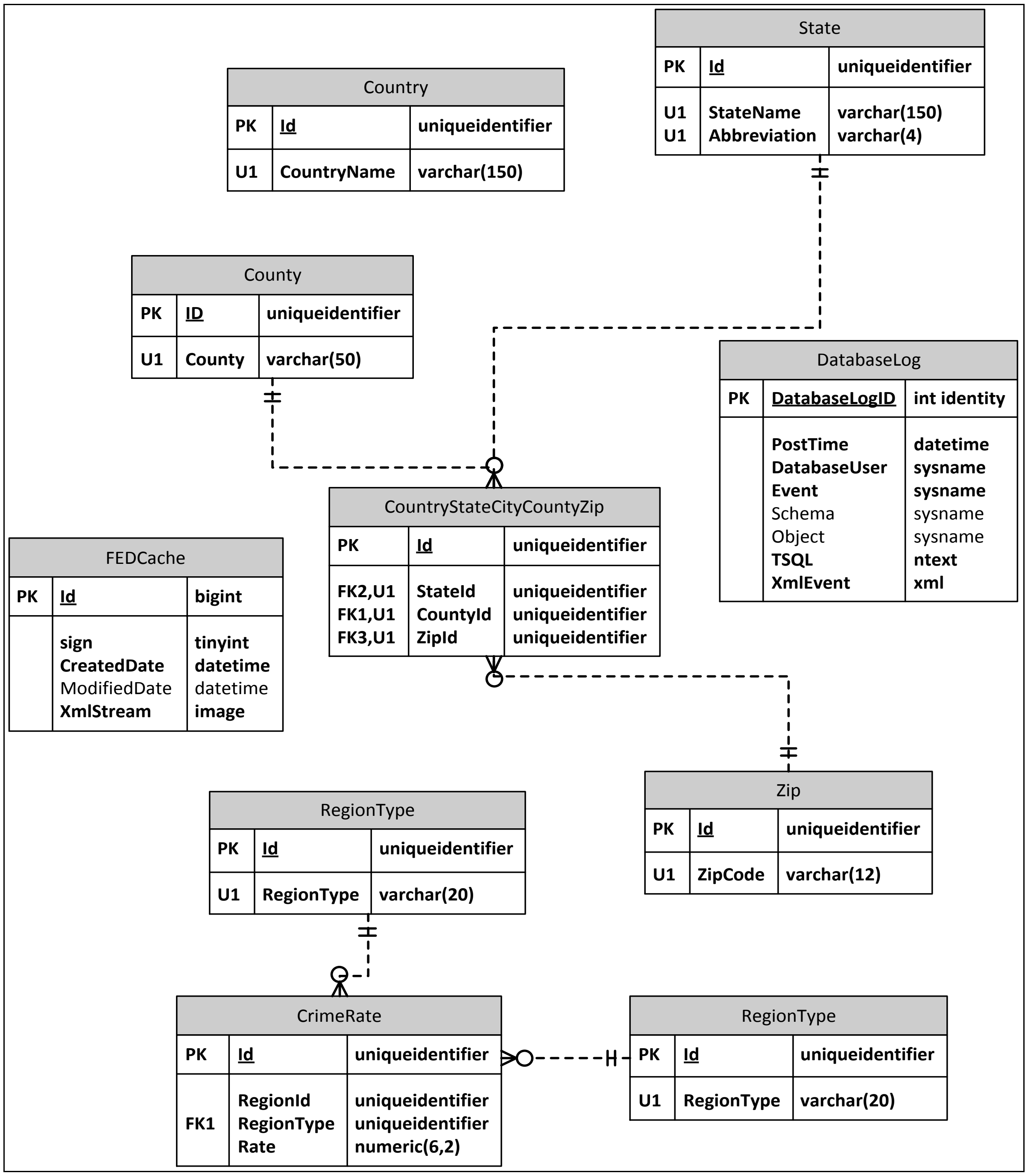


Figure 30 - Entity Relationship Diagram of Raptor Database (b)

G. Project Creation Screen Shots

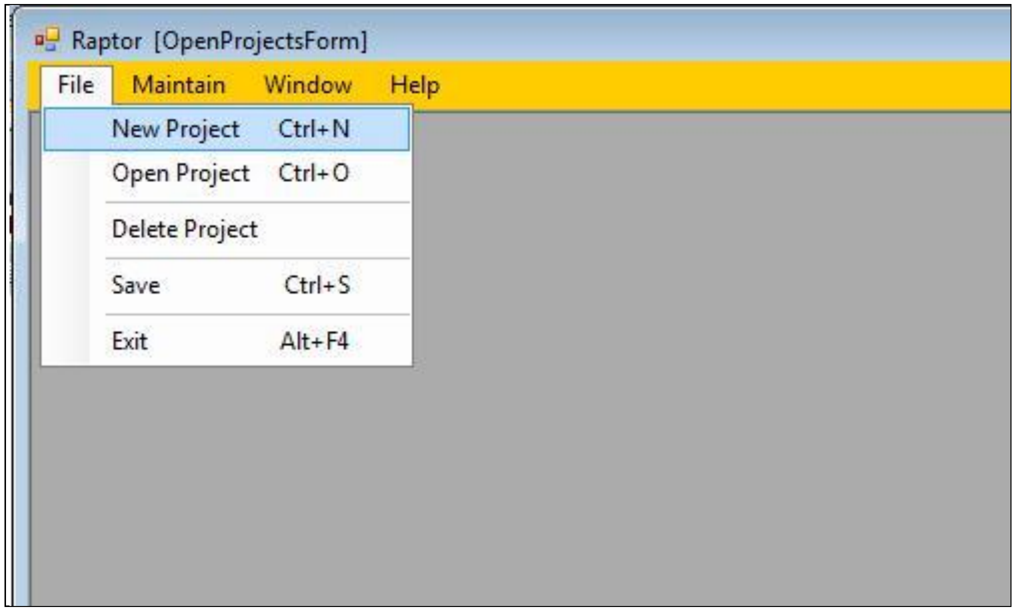
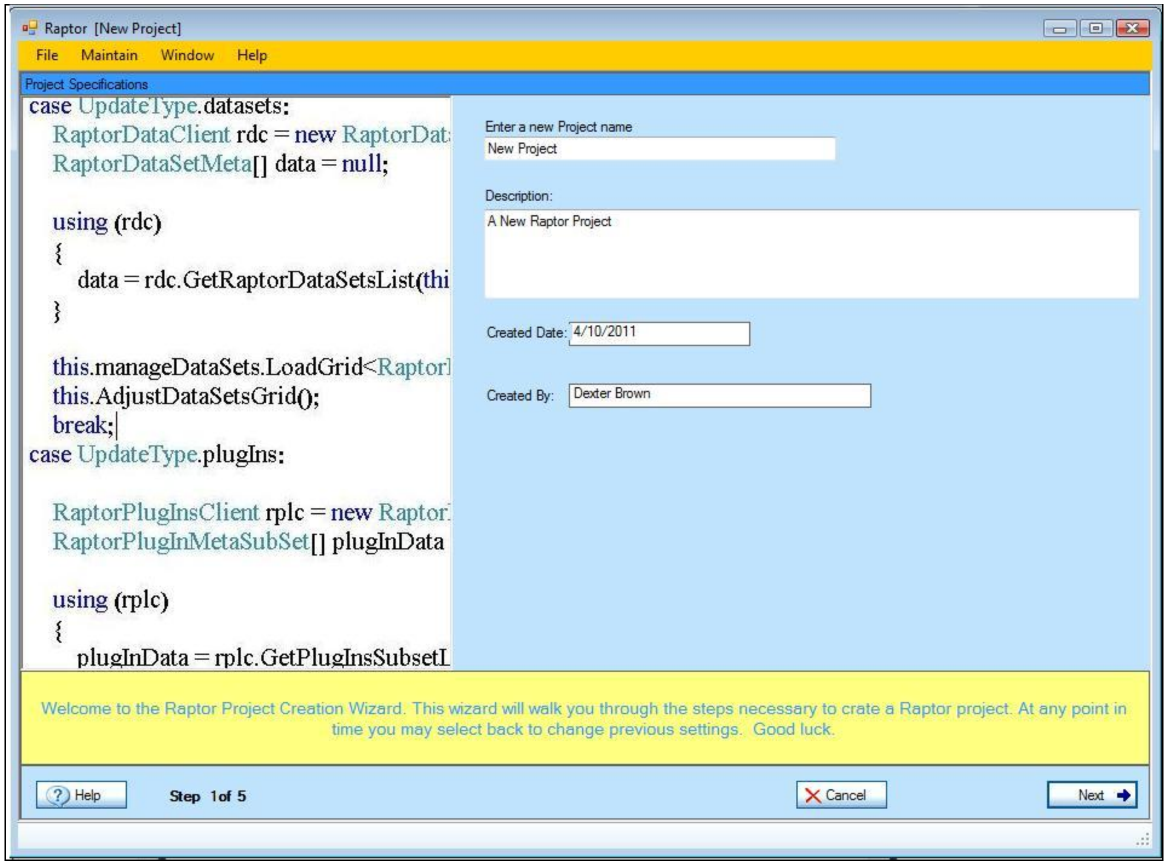


Figure 31 - Start New Project Wizard Screen.



```
case UpdateType.datasets:
    RaptorDataClient rdc = new RaptorDat
    RaptorDataSetMeta[] data = null;

    using (rdc)
    {
        data = rdc.GetRaptorDataSetsList(thi
    }

    this.manageDataSets.LoadGrid<Raptor
    this.AdjustDataSetsGrid();
    break;
case UpdateType.plugIns:

    RaptorPlugInsClient rplc = new Raptor
    RaptorPlugInMetaSubSet[] plugInData

    using (rplc)
    {
        plugInData = rplc.GetPlugInsSubsetL
```

Enter a new Project name: New Project

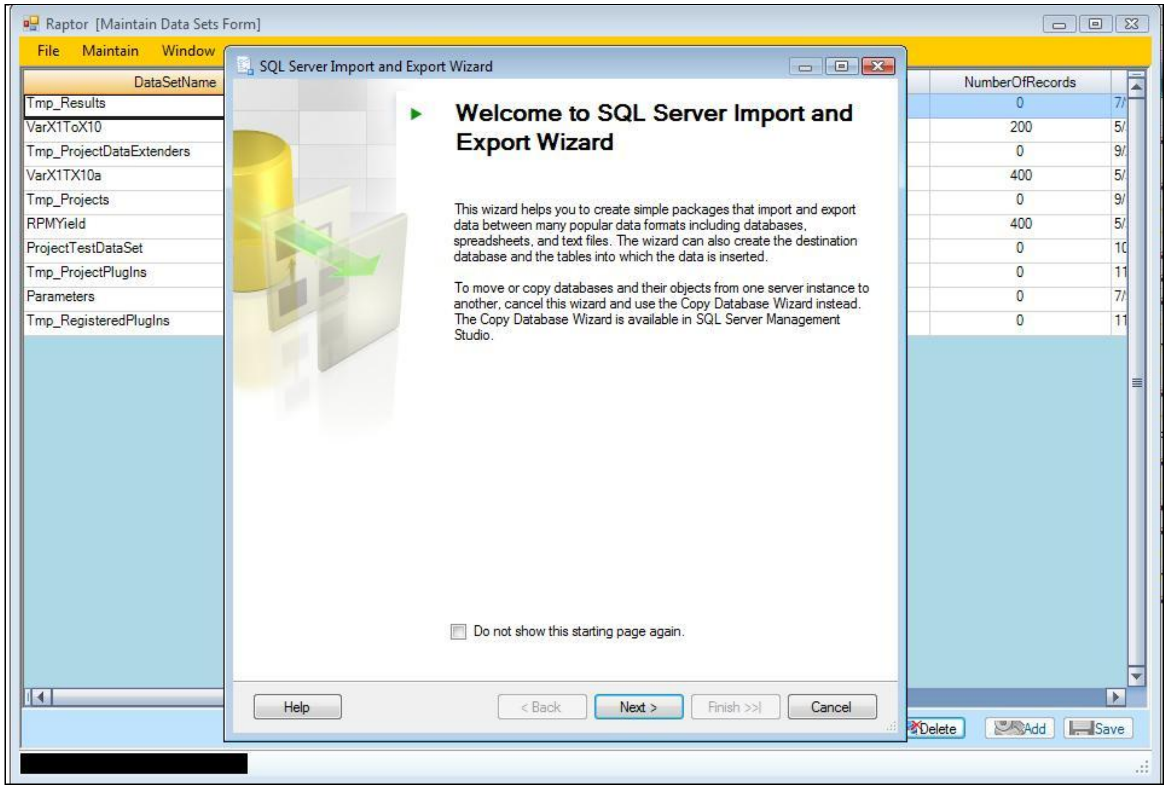
Description: A New Raptor Project

Created Date: 4/10/2011

Created By: Dexter Brown

Welcome to the Raptor Project Creation Wizard. This wizard will walk you through the steps necessary to crate a Raptor project. At any point in time you may select back to change previous settings. Good luck.

Figure 32 - New Project Screen


Raptor [Maintain Data Sets Form]
File
Maintain
Window
SQL Server Import and Export Wizard
Help
< Back
Next >
Finish >>|
Cancel
Delete
Add
Save

DataSetName	NumberOfRecords
Tmp_Results	0
VarX1ToX10	200
Tmp_ProjectDataExtenders	0
VarX1TX10a	400
Tmp_Projects	0
RPMYield	400
ProjectTestDataSet	0
Tmp_ProjectPlugIns	0
Parameters	0
Tmp_RegisteredPlugIns	0

Welcome to SQL Server Import and Export Wizard

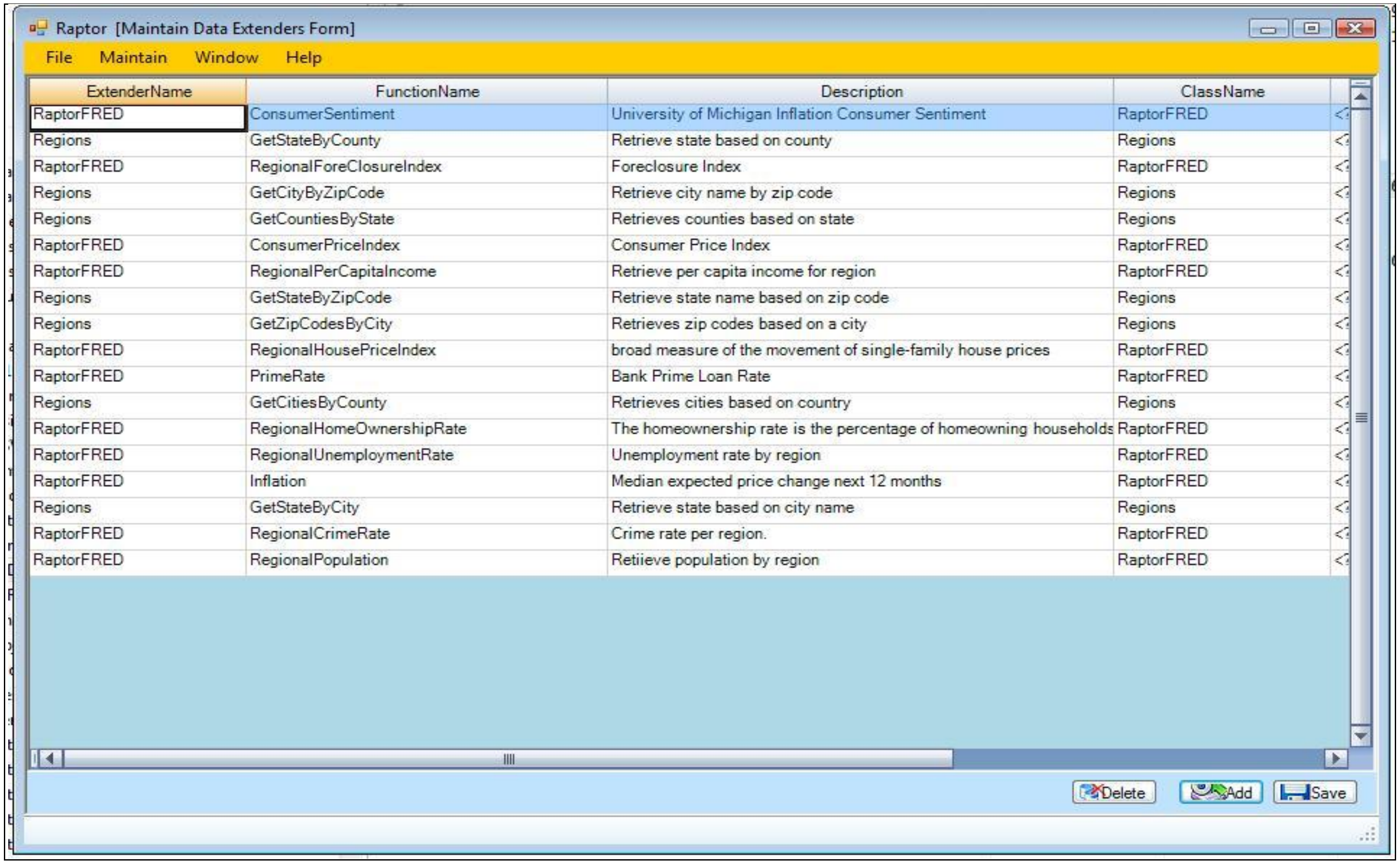
This wizard helps you to create simple packages that import and export data between many popular data formats including databases, spreadsheets, and text files. The wizard can also create the destination database and the tables into which the data is inserted.

To move or copy databases and their objects from one server instance to another, cancel this wizard and use the Copy Database Wizard instead. The Copy Database Wizard is available in SQL Server Management Studio.

☐ Do not show this starting page again.

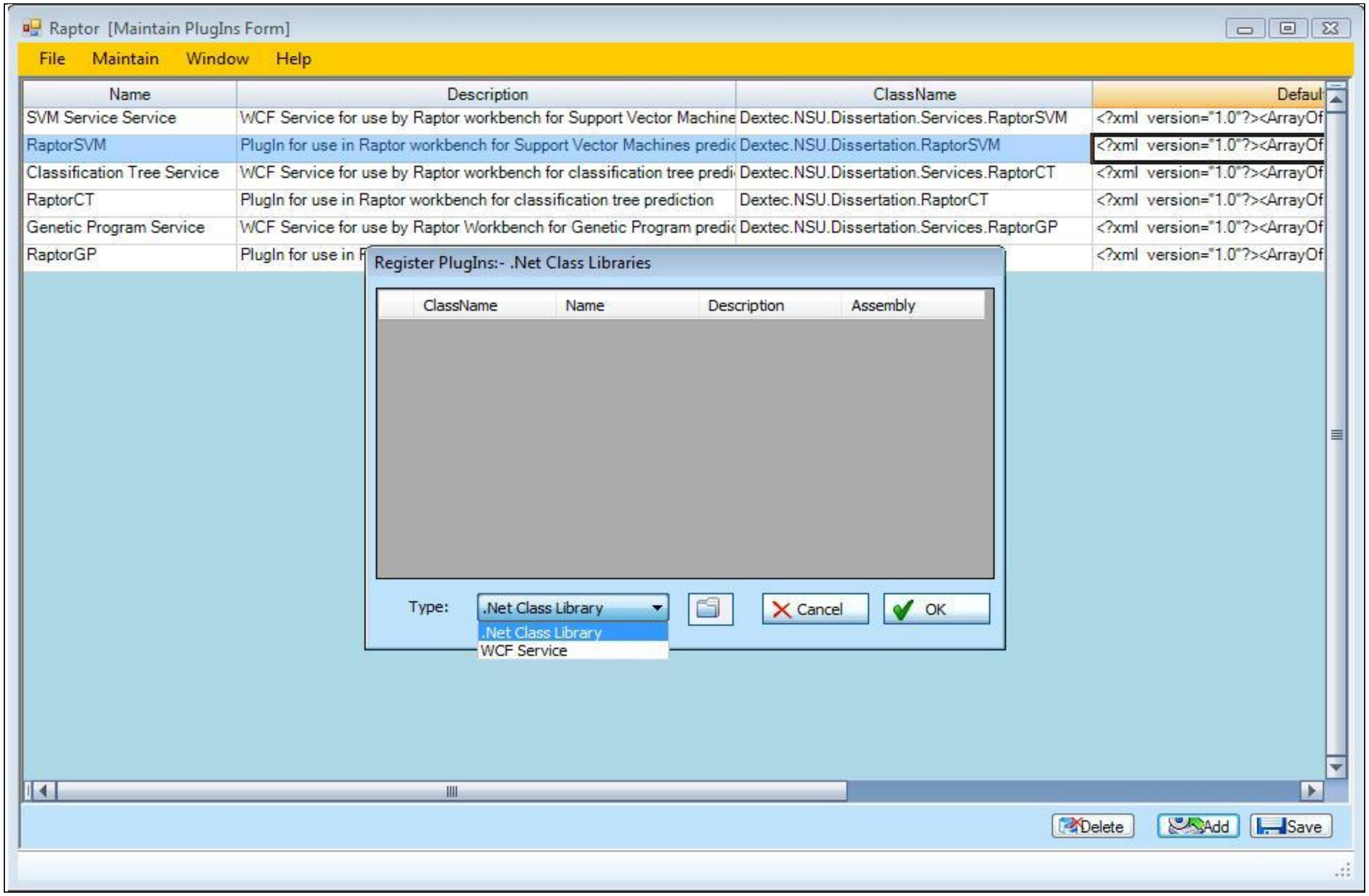
[redacted]

Figure 33 - Register Dataset Screen



ExtenderName	FunctionName	Description	ClassName
RaptorFRED	ConsumerSentiment	University of Michigan Inflation Consumer Sentiment	RaptorFRED
Regions	GetStateByCounty	Retrieve state based on county	Regions
RaptorFRED	RegionalForeClosureIndex	Foreclosure Index	RaptorFRED
Regions	GetCityByZipCode	Retrieve city name by zip code	Regions
Regions	GetCountiesByState	Retrieves counties based on state	Regions
RaptorFRED	ConsumerPriceIndex	Consumer Price Index	RaptorFRED
RaptorFRED	RegionalPerCapitaIncome	Retrieve per capita income for region	RaptorFRED
Regions	GetStateByZipCode	Retrieve state name based on zip code	Regions
Regions	GetZipCodesByCity	Retrieves zip codes based on a city	Regions
RaptorFRED	RegionalHousePriceIndex	broad measure of the movement of single-family house prices	RaptorFRED
RaptorFRED	PrimeRate	Bank Prime Loan Rate	RaptorFRED
Regions	GetCitiesByCounty	Retrieves cities based on country	Regions
RaptorFRED	RegionalHomeOwnershipRate	The homeownership rate is the percentage of homeowning households	RaptorFRED
RaptorFRED	RegionalUnemploymentRate	Unemployment rate by region	RaptorFRED
RaptorFRED	Inflation	Median expected price change next 12 months	RaptorFRED
Regions	GetStateByCity	Retrieve state based on city name	Regions
RaptorFRED	RegionalCrimeRate	Crime rate per region.	RaptorFRED
RaptorFRED	RegionalPopulation	Retiieve population by region	RaptorFRED

Figure 34 – Register Data Extenders Screen



Raptor [Maintain PlugIns Form]

File Maintain Window Help

Name	Description	ClassName	Defaul
SVM Service Service	WCF Service for use by Raptor workbench for Support Vector Machine	Dextec.NSU.Dissertation.Services.RaptorSVM	<?xml version="1.0"?><ArrayOf
RaptorSVM	PlugIn for use in Raptor workbench for Support Vector Machines predic	Dextec.NSU.Dissertation.RaptorSVM	<?xml version="1.0"?><ArrayOf
Classification Tree Service	WCF Service for use by Raptor workbench for classification tree predi	Dextec.NSU.Dissertation.Services.RaptorCT	<?xml version="1.0"?><ArrayOf
RaptorCT	PlugIn for use in Raptor workbench for classification tree prediction	Dextec.NSU.Dissertation.RaptorCT	<?xml version="1.0"?><ArrayOf
Genetic Program Service	WCF Service for use by Raptor Workbench for Genetic Program predic	Dextec.NSU.Dissertation.Services.RaptorGP	<?xml version="1.0"?><ArrayOf
RaptorGP	PlugIn for use in F		<?xml version="1.0"?><ArrayOf

Register PlugIns:- .Net Class Libraries

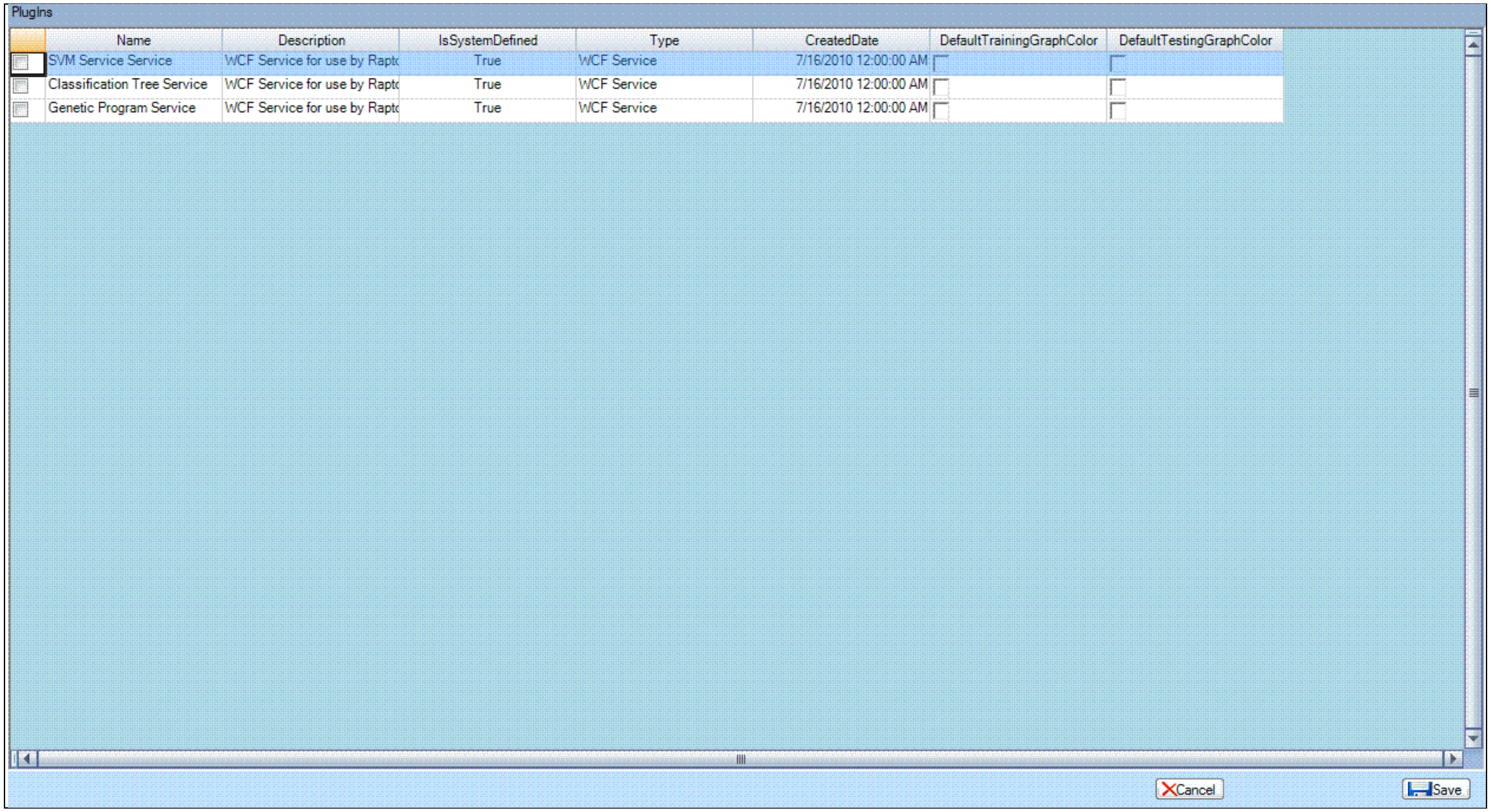
ClassName	Name	Description	Assembly

Type: .Net Class Library

.Net Class Library

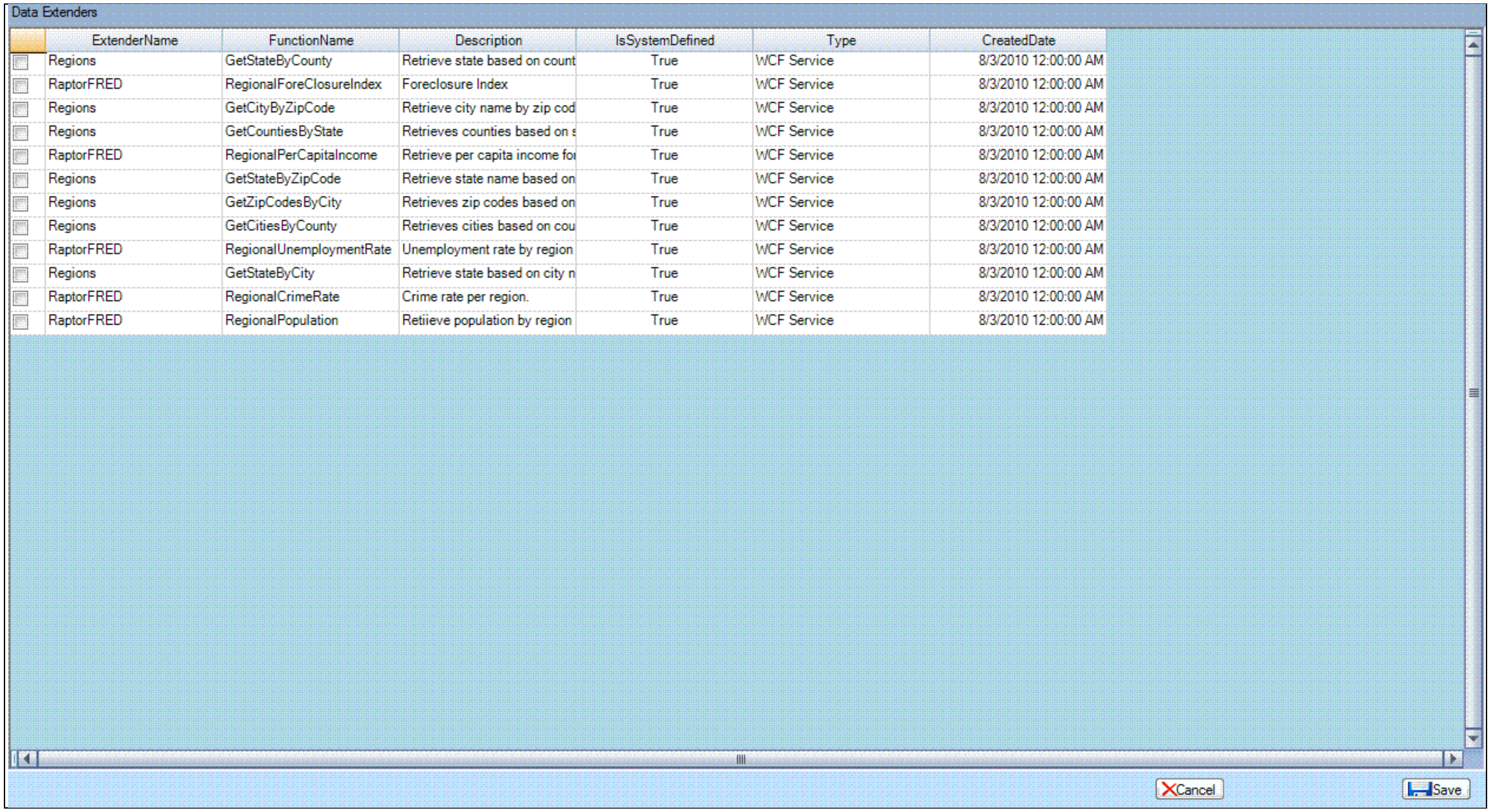
WCF Service

Figure 35 – Register Plug-Ins Screen.



	Name	Description	IsSystemDefined	Type	CreatedDate	DefaultTrainingGraphColor	DefaultTestingGraphColor
☐	SVM Service Service	WCF Service for use by Raptc	True	WCF Service	7/16/2010 12:00:00 AM	☐	☐
☐	Classification Tree Service	WCF Service for use by Raptc	True	WCF Service	7/16/2010 12:00:00 AM	☐	☐
☐	Genetic Program Service	WCF Service for use by Raptc	True	WCF Service	7/16/2010 12:00:00 AM	☐	☐

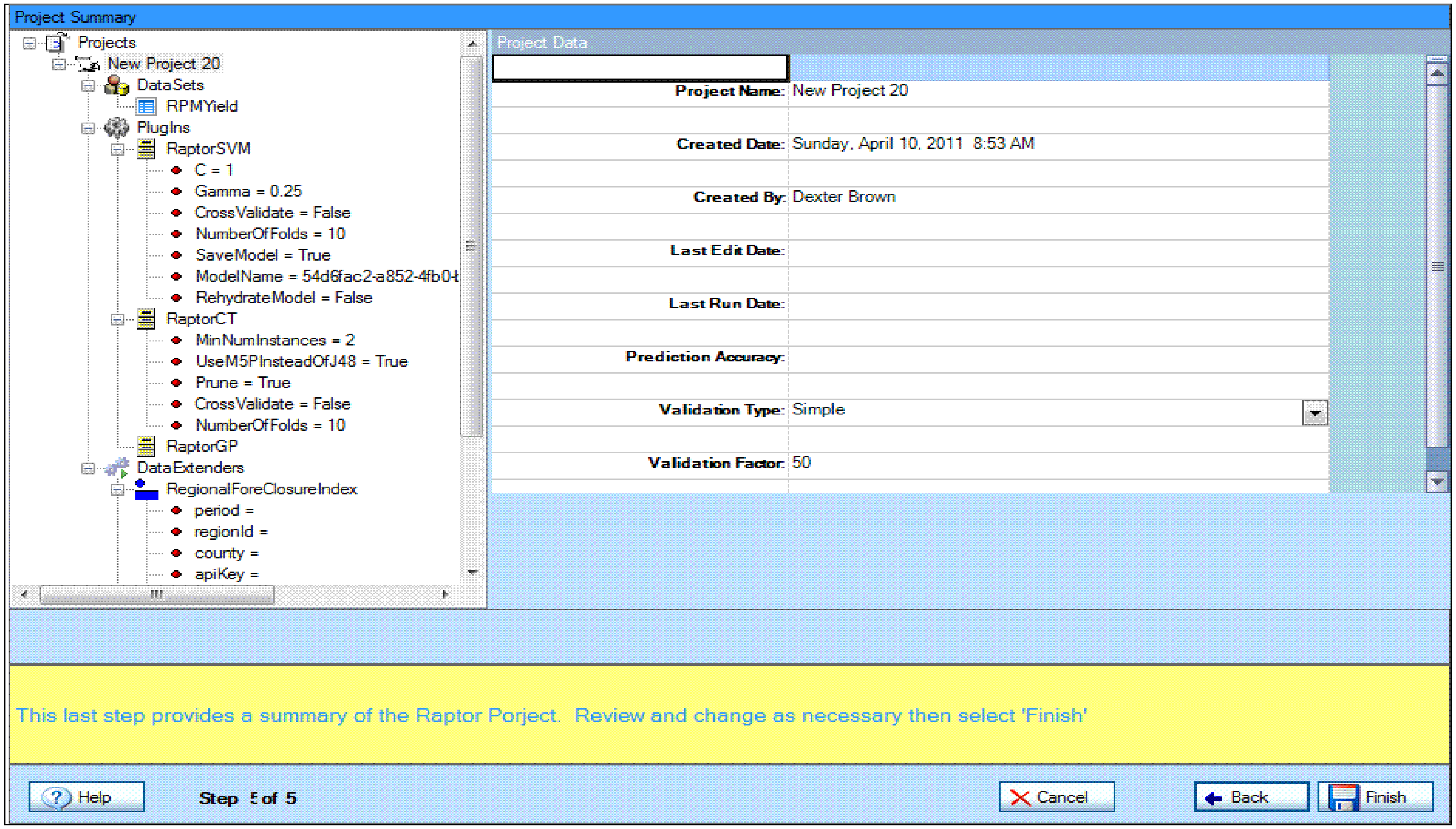
Figure 36 – Add Plug-In Screen



Data Extenders

	ExtenderName	FunctionName	Description	IsSystemDefined	Type	CreatedDate
☐	Regions	GetStateByCounty	Retrieve state based on count	True	WCF Service	8/3/2010 12:00:00 AM
☐	RaptorFRED	RegionalForeClosureIndex	Foreclosure Index	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetCityByZipCode	Retrieve city name by zip cod	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetCountiesByState	Retrieves counties based on s	True	WCF Service	8/3/2010 12:00:00 AM
☐	RaptorFRED	RegionalPerCapitaIncome	Retrieve per capita income fo	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetStateByZipCode	Retrieve state name based on	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetZipCodesByCity	Retrieves zip codes based on	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetCitiesByCounty	Retrieves cities based on cou	True	WCF Service	8/3/2010 12:00:00 AM
☐	RaptorFRED	RegionalUnemploymentRate	Unemployment rate by region	True	WCF Service	8/3/2010 12:00:00 AM
☐	Regions	GetStateByCity	Retrieve state based on city n	True	WCF Service	8/3/2010 12:00:00 AM
☐	RaptorFRED	RegionalCrimeRate	Crime rate per region.	True	WCF Service	8/3/2010 12:00:00 AM
☐	RaptorFRED	RegionalPopulation	Retiieve population by region	True	WCF Service	8/3/2010 12:00:00 AM

Figure 37 – Add Data Extenders


Project Summary

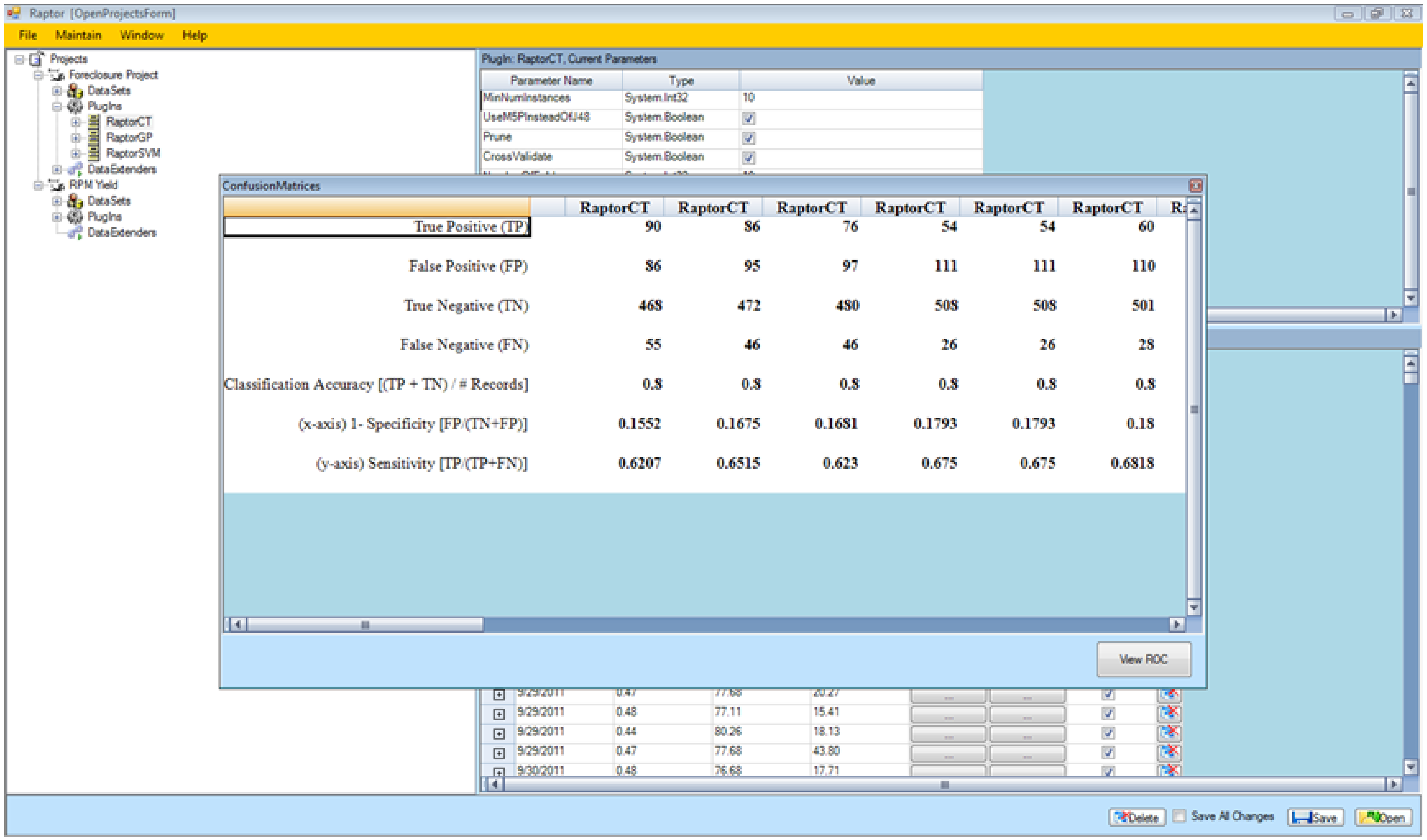
- Projects
  - New Project 20
    - DataSets
      - RPMYield
    - PlugIns
      - RaptorSVM
        - C = 1
        - Gamma = 0.25
        - CrossValidate = False
        - NumberOfFolds = 10
        - SaveModel = True
        - ModelName = 54d6fac2-a852-4fb0-
        - RehydrateModel = False
      - RaptorCT
        - MinNumInstances = 2
        - UseM5PInsteadOfJ48 = True
        - Prune = True
        - CrossValidate = False
        - NumberOfFolds = 10
      - RaptorGP
    - DataExtenders
      - RegionalForeClosureIndex
        - period =
        - regionId =
        - county =
        - apiKey =

Project Data

Project Name:	New Project 20
Created Date:	Sunday, April 10, 2011 8:53 AM
Created By:	Dexter Brown
Last Edit Date:	
Last Run Date:	
Prediction Accuracy:	
Validation Type:	Simple
Validation Factor:	50

This last step provides a summary of the Raptor Porject. Review and change as necessary then select 'Finish'

Figure 38 - New Project Screen



PlugIn: RaptorCT, Current Parameters

Parameter Name	Type	Value
MinNumInstances	System.Int32	10
UseM5PInsteadOfJ48	System.Boolean	✓
Prune	System.Boolean	✓
CrossValidate	System.Boolean	✓

	RaptorCT	RaptorCT	RaptorCT	RaptorCT	RaptorCT	RaptorCT	R
True Positive (TP)	90	86	76	54	54	60	
False Positive (FP)	86	95	97	111	111	110	
True Negative (TN)	468	472	480	508	508	501	
False Negative (FN)	55	46	46	26	26	28	
Classification Accuracy [(TP + TN) / # Records]	0.8	0.8	0.8	0.8	0.8	0.8	
(x-axis) 1- Specificity [FP/(TN+FP)]	0.1552	0.1675	0.1681	0.1793	0.1793	0.18	
(y-axis) Sensitivity [TP/(TP+FN)]	0.6207	0.6515	0.623	0.675	0.675	0.6818	

9/29/2011	0.47	77.68	20.27	✓
9/29/2011	0.48	77.11	15.41	✓
9/29/2011	0.44	80.26	18.13	✓
9/29/2011	0.47	77.68	43.80	✓
9/30/2011	0.48	76.68	17.71	✓

Figure 39 – Raptor Confusion Matrix View

H. Hardware and Software Requirements

Table 41 - Software Resource Requirements

Resource	Purpose	Note
Address Database	US address database	US Postal Service. http://www.usps.com/
ALFRED® License	Consume web service.	http://alfred.stlouisfed.org
Dudley MPSR	Augment ECJ19	Available upon request from msndex@msn.com
ECJ19	GP library	Available at http://cs.gmu.edu/~eclab/projects/ecj/
Graphing & Grid Libraries.	UI components	FarPoint Grid & XYGraph Components.
IKVM.Net	Java to .Net Converter	Available at http://www.ikvm.net/.
Microsoft Excel.	Statistical Analysis	Obtained through MSDN Academic Alliance.
Microsoft Visio Enterprise Architect.	UML artifacts and code generation.	Obtained through MSDN Academic Alliance.
Smart Draw	Diagramming	http://www.smartdraw.com
SQL Server 2008 Developer Edition.	Database Server.	Obtained through MSDN Academic Alliance.
LibSVM.	SVM library	Available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/
Visual Studio 2010 Premium.	IDE for C#.Net	Obtained through MSDN Academic Alliance.
WEKA Workbench	Classification tree library.	Available at http://www.cs.waikato.ac.nz/ml/weka

Table 42 - Hardware Resource Requirements

Resource	Purpose	Note
Desktop PC with OS >= Windows Vista.	Workbench client	
MSDN Account.	Azure development account.	

References

Abdelwahed, T. & Amir, E. M. (2005). New evolutionary bankruptcy forecasting model based on genetic algorithms and neural networks. *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*. IEEE Computer Society.

Abdou, H. A. (2009). Genetic programming for credit scoring: The case of Egyptian public sector banks. *Expert Systems with Applications*, In Press, Uncorrected Proof.

Abu-Nimeh, S., Nappa, D., Wang, X. & Nair, S. (2007). A comparison of machine learning techniques for phishing detection. *Proceedings Of The Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit*. Pittsburgh, Pennsylvania, ACM.

Arafah, A. & Haider, H. (2009). Closer look at loan defaults ahead of prison terms. *McClatchy - Tribune Business News*.

ALFRED®. (2009). *ArchivaL Federal Reserve Economic Data*. Retrieved March 27th, 2009, from http://alfred.stlouisfed.org.

Altman, E. I. (1984). The success of business failure prediction models : An international survey. *Journal of Banking, Accounting & Finance*, **8**(2),171-198.

Anonymous. (2009a). Freddie Mac stops foreclosure sales on loans eligible for new Obama home affordable modification program. *PR Newswire*. March 4th 2009.

Anonymous. (2009b). Open forum: Staving off foreclosures. *National Mortgage News*, **33**(22), 4.

Baesens, B., Van Gestel, T., Stepanova, M., Suykens, J. & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, **54**, 627-635.

Beaver, W. (1966). Financial ratios as predictors of failures. Empirical research in accounting: Selected studies. *Journal of Accounting Research*, **5**, 71–111.

Bellotti, T. & Crook, J. (2009). Support vector machines for credit scoring and discovery of significant features. *Expert Systems with Applications*, **36**(2, Part 2), 3302-3308.

Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, **5**, 1089-1105.

Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings Of The Fifth Annual Workshop On Computational Learning Theory*. Pittsburgh, Pennsylvania, United States, ACM.

Bowden, E. V. (1992). *Economics: The Science of Common Sense. (7th ed.).* Cincinnati, Ohio: Southwestern Publishing.

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

Brown, C. (2009). Mortgage foreclosure predictions for 2010. *The Mortgage Professional*. Retrieved February 5th, 2010, from http://nationalmortgageprofessional.com/news15276/mortgage-foreclosure-predictions-2010.

Brown, S. (2009). Brief: Dallas-fort worth sees jump in foreclosures of $1 million-plus homes. *McClatchy - Tribune Business News*.

Bruce, W. S. (1995). The application of genetic programming to the automatic generation of object-oriented programs. PhD Dissertation. *Graduate School of Computer and Information Sciences*. Nova Southeastern University.

Calhoun, M. (2010). Center for Responsible Lending. *Mergers & Acquisitions Business*, 405. Retrieved February 7th, 2010, from http://proquest.umi.com.ezproxylocal.library.nova.edu/pqdweb?index=2&did=1954476531&SrchMode=2&sid=1&Fmt=3&VInst=PROD&VType=PQD&RQT=309&VName=PQD&TS=1265581363&clientId=17038.

Chang, C. & Lin, C. (2009). LIBSVM: A Library for support vector machines. Retrieved May 14th, 2009, from http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Charalambous, C., Charitou, A. & Kaourou, F. (2000). Comparative analysis of artificial neural network models: Application in bankruptcy prediction. *Annals of Operations Research*, **99**(1-4), 403-425.

Cho, I., & Kim, Y. (2001). Critical factors for assimilation of object-oriented programming languages. *Journal of Management Information Systems,* **18**(3): 125.

Cielen, A., Peeters, L. & Vanhoof, K. (2004). Bankruptcy prediction using a data envelopment analysis. *European Journal of Operational Research*, **154**(2), 526-532.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. *Proceedings of the 23rd international conference on machine learning*. Pittsburgh, Pennsylvania, ACM.

DeGroat, B. (2009). Subprime mortgage crisis: Failure to predict failure. *US Fed News Service*, Including US State News.

Dudley, D. S. (2011). The design and development of a dynamic object-oriented genetic program for symbolic regression. Retrieved April 4th, 2011, from http://www.CodePlex.aspx?Dudley.

Durbin, R. (2010). $98 million in recovery act funding will be used to put Chicagoans back to work, mitigate foreclosure crisis. *US Fed News Service*. Retrieved February 7th, 2010, from http://proquest.umi.com.ezproxylocal.library.nova.edu/pqdweb?index=3&did=1947652281&SrchMode=2&sid=3&Fmt=3&VInst=PROD&VType=PQD&RQT=309&VName=PQD&TS=1265582093&clientId=17038.

Esmeir, S. & Markovitch, S. (2004). Lookahead-based algorithms for anytime induction of decision trees. *Proceedings Of The Twenty-First International Conference On Machine Learning*. Banff, Alberta, Canada, ACM.

Ellen, Y. (2009). Renters being forced out by foreclosure on landlords. *McClatchy - Tribune Business News*.

Fan, A. & Palaniswami, M. (2000). Selecting bankruptcy predictors using a support vector machine approach. *Proceedings of the International Joint Conference on Neural Networks (IJCNN'00)*. Australia, IEEE Press.

Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures. Ph.D. dissertation, University of California, Irvine, United States. California. Retrieved January 4th, 2010, from Dissertations & Theses.

Finlay, S. (2009). Are we modeling the right thing? The impact of incorrect problem specification in credit scoring. *Expert Systems with Applications*, **36**(5), 9065-9071.

Foote, C. L., Gerardi, K. & Willen, P. S. (2008). Negative equity and foreclosure: Theory and evidence. *Journal of Urban Economics*, **64**(2), 234-245.

ForeclosureU.com. (2009). ForeclosureU introduces new web based software to help troubled property owners obtain mortgage modifications. *Computer Weekly News*, Sept. 10th, 63.

Frydman, H., Altman, E. I., Kao, D. (1985). Introducing recursive partitioning for financial classification: the case of financial distress. *The Journal of Finance*, **40**(1), 269.

Gao, Z., Cui, M. & Po, L. M. (2008). Enterprise bankruptcy prediction using noisy-tolerant support vector machine. *Proceedings of the 2008 International Seminar on Future Information Technology and Management Engineering, FITME 08*, China. IEEE Computer Society.

Gores, P. (2009a). Brief: Foreclosure filings up 5% in February. *McClatchy - Tribune Business News*.

Gores, P. (2009b). Home foreclosures increase again in Wisconsin. *McClatchy – Tribune Business News*.

Ghahramani, Z. (2004). Unsupervised Learning. *Gatsby Computational Neuroscience Unit*. University College London. Retrieved May 10th, 2009, from http://www.gatsby.ucl.ac.uk/~zoubin/course05/ul.pdf.

Grover, M., Smith, L. & Todd, R. M. (2008). Targeting foreclosure interventions: An analysis of neighborhood characteristics associated with high foreclosure rates in two Minnesota counties. *Journal of Economics and Business*, **60**(1-2), 91-109.

Gutierrez, S. (2009a). U.S. Foreclosure index: U.S. Foreclosures about 1 million in 2008; Fourth quarter shows decline over third-quarter peak. *Business Wire*.

Gutierrez, S. (2009b). U.S. Foreclosure index: Foreclosures slow dramatically, down more than 25% in January; California foreclosures at lowest level since December 2007. *Business Wire*. February 11th 2009.

Herschtal, A., & Raskutti, B. (2004). Optimizing area under the roc curve using gradient descent. *Proceedings of the twenty-first international conference on Machine learning*. Banff, Alberta, Canada, ACM.

Hevner, A. R., March, S., Park, J., & Ram, S. (2004). Design science in information research. *MIS Quarterly*, **28**(1), 75-105.

Holmes, G., Donkin, A. & Witten, I. H. (1997). WEKA: A machine learning workbench. *Department of Computer Science: University of Waikato*. Retrieved May 16th, 2009, from http://www.cs.waikato.ac.nz/~ml/publications/1994/Holmes-ANZIIS-WEKA.pdf

Johnson, B. (2009). The tsunami effect. *National Real Estate Investor*. Retrieved February 5th, 2010, from http://nreionline.com/finance/real-estate-foreclosure-predictions-0501/.

Kay, A. C. (1996). The early history of Smalltalk. *History of programming languages II*, ACM Press: 511-598.

Kiviluoto, K. (1998). Predicting bankruptcies with the self-organizing map. *Neurocomputing*, **21**(1-3), 191-201.

Koza, J. (2008). Introduction to genetic programming: Tutorial. *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*. Atlanta, GA, USA, ACM.

Koza, J., Andre, D., Bennett III, F. & Keane, M. (1999). *Genetic Programming 3: Darwinian Invention and Problem Solving*. Morgan Kaufman.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection (1st ed.)*. MIT Press.

Laitinen, E. K. & Laitinen, T. (2000). Bankruptcy prediction: Application of the Taylor's expansion in logistic regression. *International Review of Financial Analysis*, **9**(4), 327-349.

Langdon, W. B. & Poli, R. (2002). *Foundations of genetic programming*. Springer Press.

Langley, P. & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, **38**(11), 54-64.

Lee, Y. C. (2007). Application of support vector machines to corporate credit rating prediction. *Expert Systems with Applications*, **33**, 67-74.

Lee, K. C., Han, I. & Kwon, Y. (1996). Hybrid neural network models for bankruptcy predictions. *Decision Support Systems*, **18**(1), 63-72.

Lensberg, T., Eilifsen, A. & McKee, T. E. (2006). Bankruptcy theory development and classification via genetic programming. *European Journal of Operational Research*, **169**(2), 677-697.

Li, J. (2006). Comparing cart with back propagation neural networks in vegetation greenness classification. *Proceedings of the 44th annual Southeast regional conference*. Melbourne, Florida, ACM.

Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Popovici, E., & et al. (2008). ECJ 18 - A Java-based Evolutionary Computation Research System. *George Mason University, Evolutionary Computation Laboratory*. Retrieved December 14th, 2008, from http://cs.gmu.edu/~eclab/projects/ecj/.

Maurer, H., & Linblad, C. (2009). Stepping up the foreclosure fight. *Business Week*, (4121), 4.

Marais, M. L., Patell, J. M. & Wolfson, M. A. (1984). The experimental design of classification models: An application of recursive partitioning and bootstrapping to commercial bank loan classifications. *Journal of Accounting Research*, **22**, 84- 113.

MBA. (2008). Research Data Notes:Sources of Foreclosure Data. Retrieved May 4th, 2008, from http://www.mortgagebankers.org/~/SourcesofForeclosureData.pdf

McKee, T. E. (2000). Developing a bankruptcy prediction model via rough sets theory. *International Journal of Intelligent Systems in Accounting Finance & Management*, **9**(3), 159-173.

McKee, T. E. & Lensberg, T. (2002). Genetic programming and rough sets: A hybrid approach to bankruptcy classification. *European Journal of Operational Research*, **138**(2), 436-451.

Min, S. H., Lee, J. & Han, I. (2006). Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert Systems with Applications*, **31**(3), 652-660.

Min, J. H. & Lee, Y. C. (2005). Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert Systems with Applications*, **28**(4), 603-614.

Mora, A. M., Castillo, P. A., Merelo, J. J., Alfaro-Cid, E., et al. (2008). Discovering causes of financial distress by combining evolutionary algorithms and artificial neural networks. *Proceedings Of The 10th Annual Conference On Genetic And Evolutionary Computation*. Atlanta, GA, USA. ACM Press.

Morasca, S. (2002). A proposal for using continuous attributes in classification trees. *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*. Ischia, Italy, ACM.

Moschitti, A. (2008). Kernel methods, syntax and semantics for relational text categorization. *Proceeding Of The 17th ACM Conference On Information And Knowledge Management*. Napa Valley, California, USA, ACM.

Moore, A. W. (2003). Support Vector Machines. *School of Computer Science: Carnegie Mellon University*. Retrieved May 22nd, 2009, from http://www.autonlab.org/tutorials/svm15.pdf

MSDN (2008). Microsoft Neural Network Algorithm (SSAS). *SQL Server 2005 Books Online*. Retrieved April 19th, 2009, from http://msdn.microsoft.com/en-us/library/ms174941(SQL.90).aspx

Murphy, P. M. & Pazzani, M. J. (1994). Exploring the decision forest: An empirical investigation of Occam's razor in decision tree induction. *Journal of Artificial Intelligence Research*, **1**, 257-275.

myFICO (2009). The score that matters. *Fair Isaac Corporation*. Retrieved March 27th, 2009, from http://www.myfico.com/Default.aspx

Nordin, P. (1997). *Evolutionary program induction of binary machine code and its application*. Krehl Verlag, Munster, Germany.

Odeh, O., Koduru, P., Das, S., Featherstone, A. M. & Welch, S. M. (2007). A multi-objective approach for the prediction of loan defaults. *Proceedings Of The 9th Annual Conference On Genetic And Evolutionary Computation*. London, England, ACM Press.

Olick, D. (2010). Predictions 2010: Real estate. *Realty Check*. Retrieved February 5th, 2010, from http://www.cnbc.com/id/34110130

Paul, G. (2009a). Brief: Foreclosure filings up 5% in February. McClatchy - Tribune Business News.

Paul, G. (2009b). Home foreclosures increase again in Wisconsin. *McClatchy - Tribune Business News*.

Perez, M. (2006). Artificial neural networks and bankruptcy forecasting: A state of the art. *Neural Computing & Applications*, **15**(2), 154-163.

Perlich, C., Provost, F. & Simonoff, J. (2003). Tree induction vs. logistic regression: A learning-curve analysis. *Journal of Machine Learning Research*, **4**, 211-255.

Pittman, K. (2008). Comparison of data mining techniques used to predict student retention. *Ph.D. dissertation, Nova Southeastern University*, United States, Florida. Retrieved January 9th, 2010, from Dissertations & Theses @ Nova Southeastern University.

Poli, R., McPhee, N. F. & Vanneschi, L. (2008). The impact of population size on code growth in gp: Analysis and empirical validation. *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. Atlanta, GA, USA, ACM.

Pritchard, J. (2009). Mortgage. *About.com*. Retrieved March 22nd, 2009, from http://banking.about.com/od/mortgages/g/mortgage.htm.

Qi, M. (2001). Predicting US recessions with leading indicators via neural network models. *International Journal of Forecasting*, **17**, 383-401.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81-106.

Ravi Kumar, P. & Ravi, V. (2007). Bankruptcy prediction in banks and firms via statistical and intelligent techniques - A review. *European Journal of Operational Research*, **180**(1), 1-28.

Riolo, R., Worzel, B. & Soule, T. (2009). Genetic programming theory and practice vi. *Proceedings of the 6th Annual Genetic Programming Theory and Practice Workshops*. New York, NY. Springer Press.

Rose, M. J. (2008). Predatory lending practices and subprime foreclosures: Distinguishing impacts by loan category. *Journal of Economics and Business,* **60**(1-2), 13-32.

Russel, S., & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach. (2nd ed.*). New Jersey: Pearson Education.

Sai, Y. & Zhong, C. (2007). A hybrid GA-BP model for bankruptcy prediction. *Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07).* IEEE Computer Society.

Sai, Y., Zhong, C. & Nie, P. (2007). A hybrid RST and GA-BP model for Chinese listed company bankruptcy prediction. *Proceedings of the Third International Conference on Natural Computation (ICNC 2007).* IEEE Computer Society.

Sanders, D. H. (1990). *Statistics: A Fresh Approach (4th ed.).* McGraw-Hill.

Schebesch, K. B. & Stecking, R. (2005). Support vector machines for classifying and describing credit applicants: Detecting typical and critical regions. *Journal of the Operational Research Society,* **56**, 1082-1088.

Schuetz, J., Been, V. & Ellen, I. G. (2008). Neighborhood effects of concentrated mortgage foreclosures. *Journal of Housing Economics*, **17**(4), 306-319.

Shin, K. S., Lee, T. S. & Kim, H. J. (2005). An application of support vector machines in bankruptcy prediction model. *Expert Systems with Applications*, **28**(1), 127-135.

Shin, K. S. & Lee, Y. J. (2002). A genetic algorithm application in bankruptcy prediction modeling. *Expert Systems with Applications*, **23**(3), 321-328.

Silva, J. (2009). Foreclosures: Predictions for 2010. *Foreclosure Dump Blog*. Retrieved January 10th, 2010, from http://www.foreclosuredump.com.

Silva, S. & Costa, E. (2005). Resource-limited genetic programming: The dynamic approach. *Proceedings of the 2005 conference on Genetic and evolutionary computation*. Washington DC, USA, ACM.

Steinwart, I. & Christmann, A. (2008). *Support Vector Machines (1st ed.).* Springer Press.

Steve, B. (2009). Brief: Dallas-fort worth sees jump in foreclosures of $1 million-plus homes. *McClatchy - Tribune Business News*.

Steve, T. (2009). Peoria area foreclosures on the rise. *McClatchy - Tribune Business News*.

Strupp, R. J. (2009). Fraud hits FHA loans. *The Sun*.

Tan, A. C. & Gilbert, D. (2003). An empirical comparison of supervised machine learning techniques in bioinformatics. *Proceedings of the First Asia-Pacific bioinformatics conference on Bioinformatics 2003 - Volume 19*. Adelaide, Australia, Australian Computer Society, Inc.

Train, K. (2003). *Discrete Choice Methods with Simulation*. Cambridge University Press.

Tsai, C. F. (2009). Feature selection in bankruptcy prediction. *Knowledge-Based Systems*, **22**(2), 120-127.

Tsai, C. F. & Wu, J. W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, **34**(4), 2639-2649.

Vapnik, V.N., & Chervonenkis, A.Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, **16**(2), 264–280.

Wilson, R. L. & Sharda, R. (1994). Bankruptcy prediction using neural networks. *Decision Support Systems*, **11**(5), 545-557.

Wu, C. H., Tzeng, G. H., Goo, Y. J. & Fang, W. C. (2007). A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. *Expert Systems with Applications*, **32**(2), 397-408.

Yu, L., Wang, S. & Lai, K. K. (2009). An intelligent-agent-based fuzzy group decision making model for financial multi-criteria decision support: The case of credit scoring. *European Journal of Operational Research*, **195**(3), 942-959.

Zhang, G., Hu, M. Y., Patuwo, B. E. & Indro, D. C. (1999). Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis. *European Journal of Operational Research*, **116**(1), 16-32.

NOVA SOUTHEASTERN UNIVERSITY

The Graduate School of Computer and Information Sciences

Certification of Authorship

Submitted to (Advisor's Name): Dr. Sumitra Mukherjee

Student's Name: Dexter R. Brown

Date of Submission: 01/03/2012

Purpose and Title of Submission: Dissertation Report: **A Comparative Analysis Of Machine Learning Techniques For Foreclosure Prediction**

Certification of Authorship: I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas, or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for this purpose.

[illegible]

Student's Signature: